**MINISTÉRIO DA EDUCAÇÃO**
**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**
**DEPARTAMENTO ACADEMICO DE INFORMÁTICA – CAMPUS CURITIBA**
**VII CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA**

JOEDE BABY FADEL FILHO

# SISTEMA PARA CONTROLE DE CONSORCIO INTERMUNICIPAL DE SAÚDE

CURITIBA
2012

JOEDE BABY FADEL FILHO

# SISTEMA PARA CONTROLE DE CONSORCIO INTERMUNICIPAL DE SAÚDE

Trabalho de Conclusão de Curso de Especialização em Tecnologia Java pela Universidade Tecnológica Federal do Paraná – Curitiba.

Orientador: Prof. Robson Ribeiro Linhares

CURITIBA
2012

# FOLHA DE APROVAÇÃO

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção de Especialista em Tecnologia Java pela Universidade Tecnológica Federal do Paraná – Curitiba.

_____

Professor Robson Ribeiro Linhares
Orientador

_____

Professor Nelson Kashima
Membro da banca examinadora

_____

Professor Paulo Bordin
Membro da banca examinadora

# DEDICATÓRIA

A Deus por tudo que me proporciona na vida.
À meus pais, os quais amo muito, pelo exemplo de vida e família.
À minha esposa Josiane pelo carinho, compreensão e companheirismo.

# AGRADECIMENTOS

A Deus e a Nossa Senhora pela proteção e pela oportunidade de tonar-me uma pessoa mais conhecedora em busca de uma maior eficiência que permitirá a concretização de todos os meus sonhos.

Aos meus pais Joéde e Edinéia que me apoiaram e me incentivaram desde o começo e as orações que sei que nunca faltará, obrigado pai e mãe, amo vocês.

Ao amor da minha vida Josiane, agradeço por confiar em mim, estando sempre ao meu lado me concedendo conselhos sábios e me apoiando em todas as decisões, amo você.

Ao meu orientador Robson pelos conhecimentos transmitidos e pela sua paciência em corrigir o meu trabalho.

Ao meu colega Ademir que colaborou para o desenvolvimento desta monografia.

# EPÍGRAFE

“Tudo posso naquele que me fortalece”.

(Filipenses 4:13)

# RESUMO

Este trabalho apresenta um sistema de gerência de Consórcio Intermunicipal de Saúde (CIS), em que seu desenvolvimento foi baseado por estudos do atual sistema. O objetivo deste sistema é fornecer um auxílio aos administradores do consórcio e as Secretarias Municipais de Saúde, proporcionando uma maior integração. Para orientar o desenvolvimento do sistema, foi utilizada uma adequação do processo *Extreme Programming*. A linguagem de programação escolhida foi Java, juntamente com os frameworks *Java Server Faces* e *PrimeFaces.* O Sistema Gerenciador de Banco de Dados foi o MySQL.
Palavras chave: Java, Consórcio Intermunicipal de Saúde e *Java Server Faces.*

# ABSTRACT

This work shows a management system of Intermunicipal Health Consortium, whose development was based in studies of the current system. The purpose of this system is to provide aid to the consortium administrators and Municipal Health Departments, providing greater integration. In order to guide the systems it was used an adequation of Extreme Programming process. The language of programming used was JAVA associating with Frameworks Java Server Faces and PrimeFaces. The manager database was MySQL.


Key words: Java, Intermunicipal Health Consortium and Java Server Faces.

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AJAX | *Asynchronous Javascript and XML* |
| API | *Application Program Interface* |
| CIS | Consórcio Intermunicipal de Saúde |
| CLT | Consolidação da Leis Trabalhistas |
| CSS | *Cascading Style Sheets* |
| DOM | *Document Object Model* |
| EJB | *Enterprise JavaBeans* |
| HTML | *Hypertext Markup Language* |
| IDE | *Integrated Development Environment* |
| JDBC | *Java Database Connectivity* |
| JEE | *Java Enterprise Edition* |
| JME | *Java Mobile Edition* |
| JSE | *Java Standart Edition* |
| JSF | *Java Server Faces* |
| JSON | *Java Script Object Notation* |
| JSP | *Java Server Pages* |
| JSTL | JavaServer Pages Standard Tag Library |
| JVM | *Java Virtual Machine* |
| MER | Modelo Entidade-Relacionamento |
| MVC | *Model-View Controller* |
| PHP | *Personal Home Page* |
| POJOs | *Plain Old Java Object* |
| RTI | *Runtime Type Information* |
| SQL | *Structure Query Language* |
| SGBD | Sistema Gerenciador de Banco de Dados |
| UML | *Unifeld Modeling Language* |
| XHTML | *Extensible Hypertext Markup Language* |

| | |
|---|---|
| XML | *Extensible Markup Language* |
| XSTL | *Extensible Stylesheet Language Transformations* |
| XP | *Extreme Programming* |

# LISTA DE FIGURAS

# LISTA DE TABELAS

# Sumário

# 1 INTRODUÇÃO

Consórcios Intermunicipais de Saúde (CIS) são associações entre municípios que visam garantir serviços de saúde. Segundo o ex-Ministro da Saúde Carlos César Albuquerque, os consórcios estabelecem uma estratégia para melhorar qualidade dos serviços. Essas associações tem como base a igualdade entres os participantes, assim permitem aos gestores compartilharem os recursos (humanos e infraestrutura), produzindo melhores resultados, podendo ser uma importante forma de consolidação do SUS.

Um consórcio é estruturado de forma simplificada e sem burocracias, praticando a igualdade entre seus participantes. A administração das atividades busca resolver de forma ágil e sem complicação, buscando um melhor atendimento à população. De acordo com o documento “o Consórcio e a Gestão Municipal (1997, 24)” do Ministério da Saúde, o CIS pode ser estruturado administrativamente da seguinte forma:

- Com um Conselho de Municípios, em geral composto pelos Secretários de Saúde, representando os municípios, que é o nível máximo de deliberação, responsável pela condução da política do consórcio;
- Com um Conselho Fiscal, responsável pelo controle da gestão financeira do consórcio;
- Com uma Secretaria Executiva ou de Coordenação, responsável pela implementação das ações, cujo coordenador é indicado pelo Conselho de Municípios.

As equipes técnicas dos Consórcios Municipais de Saúde podem conter recursos humanos provenientes dos municípios ou contratados através de concurso público sobre regime CLT. A comunidade poderá participar das atividades administrativas, através do intermédio dos Conselhos, ficando sobre responsabilidade dos Conselhos de Saúde a fiscalização das atividades realizadas pelos CIS.

## 1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver um Sistema para Controle de Consorcio Intermunicipal de Saúde. Este sistema será desenvolvido para possibilitar um melhor gerenciamento das requisições de consultas, exames e medicamentos, tendo as seguintes funcionalidades: cadastro de fornecedores, cadastro de medicamentos, cadastro de municípios, cadastro de procedimentos e gerenciamento de requisições. O sistema desenvolvido visa facilitar as atividades administrativas.

Para desenvolver a aplicação serão estudadas as características de sistemas similares já existentes, adequando-as para interface *Web.* O processo de desenvolvimento *Extreme Programming* (XP), este foi escolhido devido a sua flexibilidade e por ser voltado para projetos pequenos e médios, o processo será adaptado para sua aplicação no desenvolvimento, também será utilizada a tecnologia Java 7, os frameworks *Java Server Faces* (JSF) e *PrimeFaces*. O servidor web para hospedar a aplicação será o *Apache Tomcat* 7.

## 1.2 OBJETIVOS ESPECÍFICOS

- Estudar os sistemas existentes.
- Implantar o sistema em um Consócio Intermunicipal de Saúde.
- Aperfeiçoar as funcionalidades dos sistemas existentes.
- Apresentar relatórios dos dados do sistema.

## 1.3 LIMITES E RESTRIÇÕES

- O sistema será on-line acessado via web, tendo como padrão os navegadores que suportam XHTML.

- O sistema não gerenciará o financeiro.
- O sistema não gerenciará as consultas médicas.

## 1.4 JUSTIFICATIVA

Devido a necessidades dos usuários do CIS, justifica-se o desenvolvimento de um sistema que gerencie as informações dos consórcios ajudando os participantes a terem um controle simplificado sobre as informações.

O sistema proposto será desenvolvido para ser utilizado na *Web*, tornando o usuário independente do sistema operacional. Além disso, irá contar com a vantagem de manutenção em tempo real, ou seja, o sistema é atualizado apenas no servidor e os usuários terão acesso ao sistema atualizado instantaneamente.

## 1.5 ORGANIZAÇÃO DO TRABALHO

O trabalho está estruturado da seguinte forma:

No Capítulo 2 estão descritos os conceitos e ferramentas utilizados para o desenvolvimento do trabalho e descreve a metodologia adotada para implementação do sistema.

No Capítulo 3 estão descritas as atividades realizadas para o desenvolvimento do sistema.

No Capítulo 4 apresenta as conclusões obtidas com o desenvolvimento do projeto.

# 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata da fundamentação teórica das tecnologias e metodologias utilizadas para implementação deste trabalho.

## 2.1 JAVA

Java é uma plataforma de programação Orientada Objetos, que é compilada para um *bytecode* interpretado pela *Java Virtual Machine* (JVM). Uma das principais características é a independência de plataforma. Com o Java é possível executar o mesmo código fonte em computadores que estão com diferentes sistemas operacionais, sem a necessidade de reescrevê-lo (DEITEL; DEITEL, 2003). A plataforma é divida em três edições de desenvolvimento *Java Standard Edition (*JSE), *Java Mobile Edition (*JME) e *Java Enterprise Edition* (JEE) que foi utilizada neste projeto.

Uma característica da plataforma aplicada no projeto, foram as técnicas de Reflexão, também conhecida como *Runtime Type Information* (RTI), que permite examinar e realizar introspecção de códigos dentro da própria classe. Em *Java* a reflexão está implementada na *Java Reflection API,* que possibilita realizar uma série de funcionalidades entre elas:

- Invocar métodos de um determinado objeto;
- Obter e definir métodos e atributos;
- Criar uma nova instancia de uma classe que não sabemos o nome até o tempo da execução;

O código abaixo exibe um código fonte utilizado no sistema que mostra a utilização de reflexão.

```
try {
    if (m.getName().indexOf("get") > -1) {
        if (m.getReturnType().toString().indexOf("Class") == -1) {
            Field field = r1.getClass().
                    getDeclaredField(getFildNameToFunction(m.getName()));
            field.setAccessible(true);
            if (field.get(r1) != null) {
                if (!field.get(r1).equals(field.get(r2))) {
                    return false;
                }
            } else if (field.get(r2) != null) {
                return false;
            }
        }
    }
} catch (NoSuchFieldException | SecurityException
        | IllegalArgumentException | IllegalAccessException ex) {
    return false;
}
```

Figura 2.1 Exemplo de aplicação da reflexão no sistema (autoria própria)

A plataforma para desenvolvimento de aplicações voltadas para execução desktop é o JSE, que permite desenvolver aplicações para desktops e servidores, oferecendo uma interface de usuário rica em versatilidade, portabilidade e segurança. O JSE possui bibliotecas de acesso a banco de dados: o *Java Database Connectivity* (JDBC), que é um conjunto de classes e interfaces que possibilita a execução de comandos *Structure Query Language* (SQL).

*Java Enterprise Edition* (JEE), que é voltada para o desenvolvimento de aplicações de grande porte, que são distribuídas em redes e disponibilizadas Web. O JEE possui todas as características do JSE, além de funcionalidades para internet, redes, intranet, entre outros. O JEE está atualmente na versão 6. Os desenvolvedores irão se beneficiar das melhoras de produtividade com *Annotations e Plain Old Java Objects* (POJOs), que diminuem a quantidade de configurações XML (Oracle, 2012). Existe uma série de especificações para o desenvolvimento de aplicações Web entre elas o *Java Server Pages* (JSP), uma tecnologia que oferece uma maneira simplificada de criar conteúdos dinâmicos para Web; e os *Servlets,* um componente que gera dados *Hypertext Markup Language* (HTML) e *Extensible Markup Language* (XML) para a camada de apresentação de um aplicativo Web.

A plataforma Java apresenta uma série de *frameworks* que facilitam o desenvolvimento de aplicações web entre eles o *Java Server Faces* (JSF) e *Prime Faces*.

## 2.2 JAVA SERVER FACES (JSF)

O JSF contém todos os códigos necessários para manipulação de eventos e organização de componentes, desenvolvendo interfaces rápidas. Pelo pouco tempo voltado para o desenvolvimento das interfaces, o programador concentra-se o seu esforço no desenvolvimento da regra de negócio de sistema (GEARY; HORSTMANN, 2005).

Na figura 2.2 é apresentado um código com a utilização deste framework, podendo destacar as tags<h:inputText> (campo para entrada de texto), <h:inputSecret> (campo para entrada de senha) e o <h:command Button> (botão para execução de comandos).

```
<h:form id="frmUsuarioSenha">
    <h:panelGrid columns="2" cellpadding="5" width="330" >
        <h:outputLabel for="nome" value="Nome:" />
        <h:outputLabel id="nome" value="#{logonUsuario.usuario.usNome}" />
        <h:outputLabel for="senha" value="Senha Atual:" />
        <h:inputSecret id="senha"  value="#{logonUsuario.usuario.usSenha}"
                       required="true"/>
        <f:facet name="footer">
            <h:commandButton value="Logon"
                             actionListener="#{alterarSenha.alterarSenha()}" />
            <h:commandButton value="Fechar"
                             action="../default.xhtml"  ajax="false"
                             immediate="true" />
        </f:facet>
    </h:panelGrid>
</h:form>
```

Figura 2.2 Exemplo de aplicação do código JSF(autoria própria)

## 2.3 PRIMEFACES E AJAX

O *PrimeFaces* é um *framework* de código aberto que possui um conjunto de componentes JSF com várias extensões entre eles: HtmlEditor, caixas de dialogo, auto complete, gráficos; construído baseado nas normas AJAX do JSF 2.0 e também possui componentes *Mobile.* O *PrimeFaces* incorpora 30 temas em sua estrutura. Ele é mantido pelo Prime Teknoloji, empresa de desenvolvimento de software turca, especializada em desenvolvimento ágil e consultoria JEE (ÇAĞATAY ÇIVICI, 2011).

O código abaixo mostra a utilização do *PrimeFaces* no sistema, podendo destacar as *tags* <p:inputText> (campo para entrada de texto) , <p:commandButton> (botão para execução de comandos) e <p:ajax> (tag que implementa o Ajax).

```
<h:form id="pes">
    <h:panelGrid columns="4" cellpadding="5"
                 style="background: #ECECEE">
        <h:outputLabel for="mun" value="Codigo Municipio:" />
        <p:inputText id="mun"
                     value="#{municipiosBean.municipios.muCodigo}" size="10">
            <p:ajax event="blur" listener="#{municipiosBean.editDados()}"
                    update="frmMunicipios" />
        </p:inputText>
        <p:commandLink immediate="true" oncomplete="modelDialog.show()">
            <p:graphicImage value="../../img/pequisa.png" width="30px" />
        </p:commandLink>
        <p:commandButton value="Novo"
                         actionListener="#{municipiosBean.novo()}"
                         update="frmMunicipios,pes" immediate="true"
                         oncomplete="requestFocus();" />
    </h:panelGrid>
</h:form>
```

Figura 2.3: Exemplo de aplicação de códigos PrimeFaces (autoria própria)

A execução do *PrimeFaces,* requer o Java na versão 5 ou superior e JSF 2.X. Abaixo é exibida uma tabela de dependências para utilização do *framework.*

| Dependência | Versão | Tipo | Descrição |
|---|---|---|---|
| JSF *runtime* | 2.0 ou 2.1 | Requerido | Apache MyFaces ou Oracle Mojarra |
| itext | 2.1.7 | Opcional | Exportar dados (PDF) |
| apache poi | 3.7 | Opcional | Exportar dados (Excel) |
| rome | 1.0 | Opcional | Leitor de Feeds |
| *commons-fileupload* | 1.2.1 | Opcional | Upload de arquivo |
| *commons-io* | 1.4 | Opcional | Upload de arquivo |

Tabela 2.1: Tabela de dependências do PrimeFaces
Fonte: ÇAĞATAY ÇIVICI, 2012 p. 13

O AJAX (*Asynchronous Javascript and XML*) não é apenas uma tecnologia, mas sim um conjunto de tecnologias trabalhando juntas, oferecendo novas funcionalidades. É responsável por unir as tecnologias XHTML, CSS, DOM, XML, XSTL, XMLHttpRequest e Java Script. Apesar do nome a utilização XML não é obrigatória, pode ser utilizado *JavaScript Object Notation* (JSON). O AJAX tem quatro princípios:

- O navegador hospeda uma aplicação, e não conteúdo: o código *JavaScript* é carregado no navegador do usuário quando acessa o site (aplicativo);
- O servidor fornece dados, e não conteúdo: o servidor fornece as informações de forma de acordo com a necessidade do cliente;
- A intenção do usuário com a aplicação pode ser flexível e continua: na utilização da aplicação, o usuário não necessita ficar submetendo informações a um formulário (interrompe a interação), as interações ficam do lado do cliente assim não há interrupção da interação com o usuário;
- Real codificação requer disciplina: se a codificação não for correta poderá sobrecarregar o servidor.

## 2.4 MYSQL

O MySQL é um dos Sistemas Gerenciadores de Banco de Dados (SGBD) mais populares, com mais de 10 milhões de instancias pelo mundo, possuindo versões para Windows, Linux, Solaris, Unix, FreeDSB. Seu principal uso é para aplicações WEB, como servidores de dados para comercio eletrônico. Ele passou a ter suporte a transações a partir da versão 5, assim se consolidando como ferramenta de armazenamento de dados (MySQL, 2012).

O Mysql é um SGBD robusto que utiliza SQL (*Structure Query Language* – Linguagem de Consulta Estuturada) muito rápida, muiti-tarefas e multi-usuários. O banco de dados oferece os padrões ANSI/ISO SQL, para utiliza-lo os usuários podem escolher entre dois tipos de licença de produto *Open Source*: utiliza a licença GNU (*General Public License*), ou comprar uma licença comercial padrão.

Seu planejamento inicial foi desenvolvido para trabalhar com dados de tamanho médio de 10 a 100 milhões de registros em sistemas de pequeno porte (MySQL, 2012).

## 2.5 VISUAL PARADIGM

O Visual Paradigm é uma ferramenta *case* voltada para o desenvolvimento visual da UML (*Unified Modeling Language*) ou Linguagem de Modelagem de Dados, podendo construir os diagramas de maneira rápida, possuindo suporte para engenharia reversa com Java, C++, PHP, entre outras. Foi projetada para um vasto leque de operadores, incluindo Engenheiros de Software, Analistas de Negócio, Analistas de Sistema, Arquiteto de Sistema. O *Visual Paradigm* tem suporte a UML 2.0, porém, quando

se vai desenvolver uma aplicação para web, a ferramenta não tem os diagramas navegacionais.

## 2.6 TOMCAT

O Tomcat é um servidor web, sendo um software livre desenvolvido pela *Apache Software Foundation,* o servidor suporta parte da especificação JEE, com as tecnologias *Servlet* e JSP. A configuração do servidor é realizada através de arquivos XML, essas informações são lidas na inicialização do contêiner, assim toda alteração das configurações representa a necessidade de reinicialização. O Tomcat não pode ser considerado um servidor de aplicações, apesar de ter algumas características, por não suportar tecnologias como EJB (*Enterprise JavaBeans*).

O servidor também foi a implementação oficial para JSP e tecnologias *Servlets* da Sun. O *Tomcat* pode desempenhar a função de servidor web, ou pode trabalhar integrado com os servidores dedicados, como o *Apache,* da fundação *Apache*, ou IIS, da Microsoft (ORACLE, 2012).

## 2.7 NETBEANS

O Netbeans é um IDE multiplataforma, que fornece uma base sólida para o desenvolvimento de aplicações, possuindo um conjunto de APIs (*Application Programming Interface*). Ele é um editor rico em recursos para desenvolvimento *Swing*, Web com recursos para *Servlet,* JSP, JSTL e EJBs, também fornece soluções para escrever sistemas em dispositivo móvel, além de ter *plugins* para UML. Por ser escrito em Java, a ferramenta é independente de plataforma, pois funciona em qualquer sistema operacional que tenha suporte a JVM. Além de *Java,* a IDE suporta C, C++, Ruby e PHP.

## 2.8 EXTREME PROGRAMMING (XP)

O *Extreme Programming* (XP) é um processo de desenvolvimento que utiliza metodologia ágil.

Segundo (SOMMERVILLE, 2006) um processo de desenvolvimento de software é um conjunto de atividades e resultados associados, que geram um produto de software.

As Metodologias Ágeis se tornaram conhecidas em 2001 quando especialistas em processo de desenvolvimento de software representaram os métodos de desenvolvimento *Extreme Programming* XP (Beck, (1999) e Scrum (Schwaber e Beedle, 2002). Deste movimento foi criada a Aliança Ágil que estabeleceu o Manifesto Ágil (*Agile Manifesto*, 2012), exibindo quatro conceitos chaves que são:

- Indivíduos e interações ao invés de processos e ferramentas de desenvolvimento;
- *Software* executável ao invés de documentação;
- Colaboração do cliente ao invés de negociações de contrato;
- Respostas rápidas a mudanças ao invés de seguir planos.

O XP por exigir uma comunicação constante entre todos indivíduos da equipe de desenvolvimento é voltado para desenvolvimento de sistemas com equipes pequenas de até 12 desenvolvedores. A metodologia possui os seguintes valores:

- *Feedback* rápido: a comunicação deve ser continua entre os participantes do projeto;
- Presumir simplicidade: o XP sugere que o programador adote a solução mais simples para resolução do problema;
- Mudanças incrementais: são realizadas pequenas mudanças no sistema;
- Abraçar mudanças: facilita a inclusão e altera as funcionalidades do sistema;

- Trabalho de qualidade: o termo qualidade para o XP é desenvolver um sistema que atenda aos requisitos do cliente.

O XP baseia-se em 12 práticas (BECK, 1999) descritas abaixo:

- Planejamento: consiste em decidir o que é necessário ser feito e o que pode ser adiado no projeto, levando em consideração os requisitos atuais.
- Entregas frequentes: visa entregar pequenas partes do sistema, atualizando-o conforme os requisitos vão surgindo.
- Metáfora: são as descrições de um software sem a utilização de termos técnicos, com o intuito de guiar o desenvolvimento do software.
- Projeto simples: o sistema desenvolvido deverá ser o mais simples possível e satisfazer os requisitos atuais.
- Testes: os testes são criados inicialmente, pois XP foca na validação durante o processo de desenvolvimento.
- Programação em pares: a codificação do sistema é realizada em duplas, enquanto um programador implementa o código e o outro observa o trabalho para identificar erros semânticos e sintáticos.
- Refatoração: foca no aperfeiçoamento do projeto de software, devendo ser realizada somente quando necessária.
- Propriedade coletiva: qualquer membro da equipe pode alterar o código-fonte, adicionando ou retirando partes do código.
- Integração contínua: essa pratica visa integrar e construir o sistema várias vezes ao dia, deixando os códigos sempre atualizados.
- 40 horas de trabalho semanal: o processo XP assume que não se deve fazer horas extras constantes, caso seja necessário por mais de uma semana, irá mostrar que existe um grave problema no projeto que deverá ser resolvido.
- Cliente presente: o cliente tem uma grande participação durante o desenvolvimento do sistema, devendo estar disponível para tirar as duvidas de requisitos.

- Código padrão: códigos com arquitetura padronizados, para que esses possam ser facilmente alterados por todos os programadores.

O XP possui papéis para demonstrar quais tarefas as pessoas irão desempenhar, onde um indivíduo envolvido no projeto poderá assumir mais de um papel no decorrer do desenvolvimento do sistema, que são:

- Treinador: Preocupa-se com a execução técnica e evolução do processo, possui conhecimentos de XP e orienta a equipe.
- Rastreador: Coleta dados sobre o andamento do projeto e a sua conformidade com o planejamento feito para as iterações e release.
- Programador: o desenvolvimento é sempre em pares, escrevendo o código e os casos de teste unitários. Também é responsável por estimar o tempo para a implementação das estórias.
- Cliente: Responsável por escrever as estórias (têm a finalidade de criar estimativas de tempo para a reunião de planejamento na qual o software será entregue) junto com os programadores e priorizá-las, ajudando na escrita dos casos de teste funcionais.
- Testador: Ajuda o cliente na definição e escrita dos testes funcionais. O responsável não precisa ter apenas essa função, podendo desempenhar também outros papéis.

O ciclo de vida do XP é relativamente pequeno em comparação aos modelos de processos convencionais, sendo dividido em quatro fases: planejamento, testes, codificação e projeto.

- Fase 1 – Planejamento: consiste em realizar a coleta dos requisitos do sistema, nesta fase podem ser realizadas as atividades estimativas de custo, escopo do projeto, dentre outras atividades que serão desenvolvidas de acordo com a necessidade do projeto, pois o XP não define um especificação formal de requisitos.

- Fase 2 – Testes: foram realizados os planejamento testes de aceitação pelo cliente e de caixa-branca pelos programadores, as atividades de teste são realizadas com o proposito de verificar se o sistema irá satisfazer as necessidades, cada funcionalidade deverá ser testada para evitar impactos negativos.
- Fase 3 – Codificação: é a fase do processo em que o sistema é implementado. No XP a qualidade do código é muito importante, ele proporciona meios para garanti-la, como a integração contínua, programação em duplas e a criação dos testes antes de ser criado o código. Além dos métodos atribuídos pelo XP é importante que seja seguido padrões de código.
- Fase 4 – Projeto: esta fase no XP é responsabilidade de toda equipe, todos os membros podem ajudar para elaboração de um projeto, uma das características do processo é a realização de pequenos projetos, porém e são realizado várias vezes durante o desenvolvimento do sistema.

Nas Fases 1 (diagrama de caso de uso) e 4 (diagrama de classe) foram utilizados diagramas da UML para auxiliar na documentação do sistema.

A metodologia XP possui restrições para sua utilização, sendo as quais:

- Cultura: a organização da empresa é inserida dentro de uma cultura tradicional que gera muita documentação, gastando muito tempo com análise e projeto antecipado. Essa cultura impossibilita a utilização do XP.
- Tamanho da equipe: quando o tamanho da equipe for grande (as equipes XP possuem geralmente 12 pessoas) fica difícil a aplicação de alguns conceitos do XP, como a comunicação em uma grande equipe.
- Tecnologia: em tecnologias para quais seja complexa a escrita dos casos de teste, e quando o *feedback* ocorre em um tempo longo ou realização de mudanças são muito complicadas, fica inviável a utilização do XP.
- Espaço físico: quando o espaço físico dificulta a comunicação entre a equipe.

- Cliente: o XP exige uma grande participação do cliente, quando ele não tem condições de participar impossibilita a sua utilização.

# 3 APLICAÇÃO DA METODOLOGIA

Este capítulo visa mostrar as atividades desenvolvidas na aplicação da metodologia XP, adequada de acordo com a necessidade do projeto.

Uma das necessidades do projeto foi recolher os requisitos do sistema e das atividades. Segundo (SOMMERVILLE, 2006) o sistema precisa ser modelado com um conjunto de componentes e de relações entre esses componentes.

Para modelagem do sistema foi utilizada a UML (*Unified Modeling Language*) ou Linguagem de Modelagem Unificada que é uma linguagem visual utilizada para modelar sistemas computacionais por meio do paradigma de Orientação a Objetos (GUEDES, 2005).

No desenvolvimento do sistema foram utilizadas 2 iterações, na primeira o foco foram os cadastros e na segunda foram as requisições e os relatórios.

## 3.1 PLANEJAMENTO

### 3.1.1 Metáfora (Requisitos do sistema)

A analise de requisitos foi desenvolvida basicamente na primeira iteração, onde foram levantadas as necessidades do sistema CIS e estudado o atual sistema.

As funcionalidades do sistema foram dividas em categorias de acordo com as suas necessidades, podendo ser:

- Essencial: funcionalidade imprescindível ao sistema.

- Importante: funcionalidade que é importante ao sistema, porém sua ausência não implicará em prejuízo.
- Desejável: funcionalidade que irá melhorar a usabilidade do sistema.

| Necessidades | Categoria |
|---|---|
| 1. Configuração do Sistema | *Importante* |
| 2. Cadastro de Munícipios | *Essencial* |
| 3. Cadastro de Fornecedor | *Essencial* |
| 4. Cadastro de Medicamentos | *Essencial* |
| 5. Cadastro de Procedimentos | *Essencial* |
| 6. Cadastro de Usuários | *Essencial* |
| 7. Gerenciamento de Requisições | *Essencial* |
| 8. Gerenciamento de Permissões | *Desejável* |
| 9. Relatórios | *Importante* |

Tabela 3.1: Funcionalidades do Sistema

1. Configuração do Sistema: Possibilitará ao usuário cadastrar imagens, textos que serão utilizados no sistema.
2. Cadastro de Municípios: Possibilitará ao usuário cadastrar, alterar os municípios participantes do consórcio.
3. Cadastro de Fornecedores: Possibilitará ao usuário cadastrar os fornecedores, são considerados fornecedores médicos, consultórios, fornecedor de medicamentos entre outros.
4. Cadastro de Medicamentos: Possibilitará ao usuário cadastrar os medicamentos utilizados no consórcio, sendo que este poderão ter vários fornecedores.

5. Cadastro de Usuários: Permitirá o cadastro de usuários do sistema, podendo ser vinculado ao município.
6. Gerenciamento de Requisições: Permitirá aos usuários gerenciar as requisições (consultas e exames), gerindo informações de exames, fornecedores e informações sobre o paciente.
7. Gerenciamento de Permissões: Permitirá o gerenciamento aos acessos dos usuários, adicionando e removendo acessos de acordo com a necessidade, o sistema possibilitará criar vários perfis de acesso.
8. Relatórios: Permitirá ao usuário visualizar dados de requisições, medicamentos e municípios.

Na tabela 3.2 exibe a comparação de funcionalidades do sistema atual com as funcionalidades do novo sistema.

| Tarefa | Sistema Atual | Sistema para Controle de Consórcio Intermunipal de Saúde |
|---|---|---|
| 1. Configuração do Sistema | | |
| 2. Cadastro de Munícipios | | |
| 3. Cadastro de Fornecedor | | |
| 4. Cadastro de Medicamentos | | |
| 5. Cadastro de Procedimentos | | |
| 6. Cadastro de Usuários | | |
| 7. Gerenciamento de Requisições | | |
| 8. Gerenciamento de Permissões | | |
| 9. Relatórios | | |

Tabela: 3.2: Comparativos entre sistemas

### 3.1.2 Análise de Requisitos

O XP não define uma maneira formal para realizar a análise de requisitos, portanto foi utilizado o diagrama de caso de uso da UML para auxiliar na documentação do sistema. Nesta fase foram formalizados os dados coletados na subfase de coleção de requisitos, onde foram realizadas as especificações dos grupos de usuários, que identifica os usuários e suas devidas categorias, formalizadas em casos de uso e nas suas especificações.

O diagrama de caso de uso procura, por meio de uma linguagem simples, possibilitar a compreensão do comportamento externo do sistema por qualquer pessoa, tentando apresenta-lo através de uma perspectiva dos usuários (GUEDES, 2006). Na tabela 3.3 demonstra os casos de uso identificados para o desenvolvimento do sistema.

| Identificador | Caso de Uso |
|---|---|
| UC1 | Gerenciar Medicamentos |
| UC2 | Gerenciar Procedimentos |
| UC3 | Gerenciar Usuários |
| UC4 | Gerenciar Fornecedores |
| UC5 | Gerenciar Requisições |
| UC6 | Imprimir Requisições |

Tabela 3.3: Casos de uso Identificados

A Figura 3.1 mostra o diagrama de caso de uso do Sistema, o ator Administrador é responsável pelos cadastros e gerenciar requisições, o ator Município são usuários da Secretaria Municipal de Saúde. O caso de uso Gerenciar Requisições é o principal caso do uso do sistema.

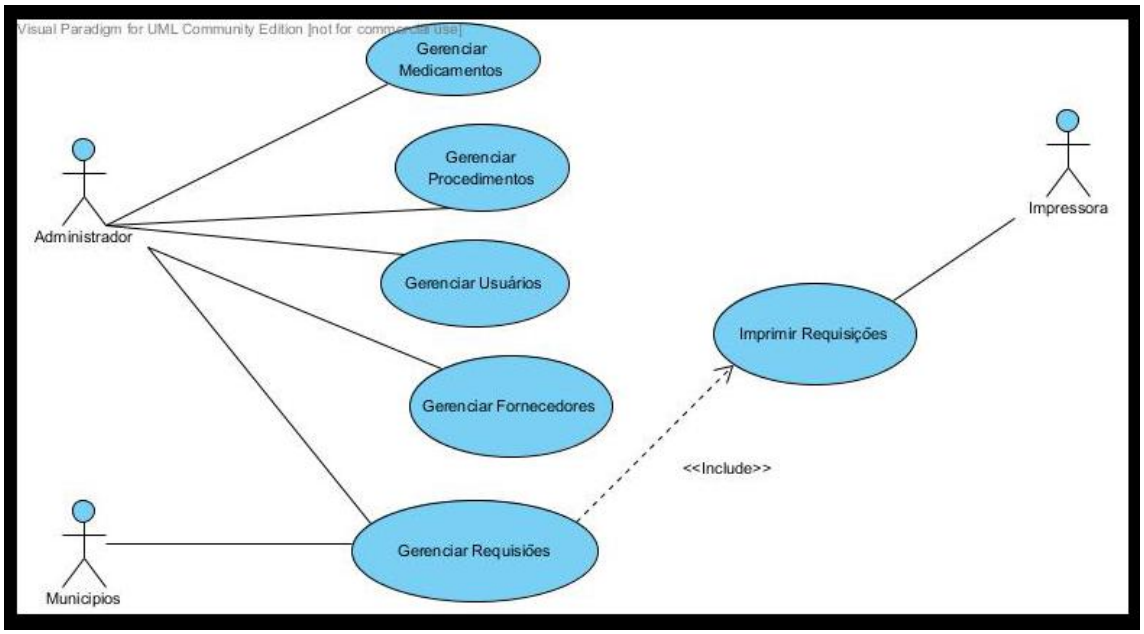

Figura 3.1: Diagrama de caso de uso

A especificação ou documentação dos Casos de Uso descreve, por meio de uma linguagem bastante simples, as funções em linhas gerais do caso de uso. Ele destaca quais atores interagem com o mesmo, quais as etapas que devem ser executadas pelo Ator e pelo sistema para que o caso de uso execute sua função, também quais parâmetros devem ser fornecidos e quais restrições e validações ele deve possuir (GUEDES, 2005).

As tabelas abaixo exibem a descrição dos casos de uso do sistema:

| Nome do Caso de Uso | Gerenciar Procedimentos |
|---|---|
| Caso de Uso Geral | |
| Ator Principal | Administrador |
| Resumo | Gerencia os cadastros de procedimentos do sistema. |
| Pré-Condição | Estar logado no sistema e ter nível de acesso. |
| Pós-Condição | |
| Ações do Ator | Ações do Sistema |
| 1. Seleciona a opção de cadastro | |
| | 2. Exibe a tela de cadastro |
| 3. Insere os dados na tela | |
| | 4. Persiste os dados |
| | 5. Exibe uma mensagem ao usuário |
| 6. Seleciona o ícone de busca | |
| | 7. Mostra os dados cadastrados na tela |
| Restrições/Validações | |

Tabela 3.4 – Exibe o caso de uso Gerenciar Procedimentos

| Nome do Caso de Uso | Gerenciar Fornecedores |
|---|---|
| Caso de Uso Geral | |
| Ator Principal | Administrador |
| Resumo | Gerencia os cadastros de fornecedores do sistema. |
| Pré-Condição | Estar logado no sistema e ter nível de acesso. |
| Pós-Condição | |
| Ações do Ator | Ações do Sistema |
| 1. Seleciona a opção de cadastro | |
| | 2. Exibe a tela de cadastro |
| 3. Insere os dados na tela | |
| | 4. Persiste os dados |
| | 5. Exibe uma mensagem ao usuário |
| 6. Seleciona o ícone de busca | |
| | 7. Mostra os dados cadastrados na tela |
| Restrições/Validações | |

Tabela 3.5 – Exibe o caso de uso Gerenciar Fornecedores

| | |
|---|---|
| Nome do Caso de Uso | Gerenciar Medicamentos |
| Caso de Uso Geral | |
| Ator Principal | Administrador |
| Resumo | Gerencia os cadastros de grupo de medicamentos e medicamentos |
| Pré-Condição | Estar logado no sistema e ter nível de acesso. |
| Pós-Condição | |
| Ações do Ator | Ações do Sistema |
| 1. Seleciona a opção de cadastro | |
| | 2. Exibe a tela de cadastro |
| 3. Insere os dados na tela | |
| | 4. Persiste os dados |
| | 5. Exibe uma mensagem ao usuário |
| 6. Seleciona o ícone de busca | |
| | 7. Mostra os dados cadastrados na tela |
| Restrições/Validações | |

Tabela 3.6 – Exibe o caso de uso Gerenciar Medicamentos

| Nome do Caso de Uso | Gerenciar Usuário |
|---|---|
| Caso de Uso Geral | |
| Ator Principal | Administrador |
| Resumo | Gerencia os usuários do sistema |
| Pré-Condição | Estar logado no sistema e ter nível de acesso. |
| Pós-Condição | |
| Ações do Ator | Ações do Sistema |
| 1. Seleciona a opção de cadastro | |
| | 2. Exibe a tela de cadastro |
| 3. Insere os dados na tela | |
| | 4. Calcula a idade do usuário do CIS |
| | 5. Persiste os dados |
| | 6. Exibe uma mensagem ao usuário |
| 7. Seleciona o ícone de busca | |
| | 8. Mostra os dados cadastrados na tela |
| Restrições/Validações | Valida se já tem um usuário cadastrado com o mesmo login |

Tabela 3.7 – Exibe o caso de uso Gerenciar Usuário

| Nome do Caso de Uso | Gerenciar Requisições |
|---|---|
| Caso de Uso Geral | |
| Ator Principal | Municípios |
| Resumo | Gerencia as requisições de requerimento de consultas do sistema |
| Pré-Condição | Estar logado no sistema e ter nível de acesso. |
| Pós-Condição | |
| Ações do Ator | Ações do Sistema |
| 1. Seleciona a opção de cadastro | |
| | 2. Exibe a tela de cadastro |
| 3. Insere os dados na tela | |
| | 4. Valida usuário |
| | 5. Persiste os dados |
| | 6. Exibe uma mensagem ao usuário |
| 7. Seleciona o ícone de busca | |
| | 8. Mostra os dados cadastrados na tela |
| Restrições/Validações | Se o usuário tiver ligado a um município não poderá altera-lo |

Tabela 3.8 – Exibe o caso de uso Gerenciar Requisições

| Nome do Caso de Uso | Imprimir Requisições |
|---|---|
| Caso de Uso Geral | Gerenciar Requisições |
| Ator Principal | Municípios |
| Resumo | Gerencia as requisições de requerimento de consultas do sistema |
| Pré-Condição | Estar logado no sistema e ter nível de acesso. |
| Pós-Condição | Ter uma impressora instalada no cliente. |
| Ações do Ator | Ações do Sistema |
| 1. Clica no botão imprimir | |
| | 2. Gera um pdf com os dados da requisição |
| 3. Coloca o comando para imprimir | |
| | 4. Imprime a requisição |
| Restrições/Validações | |

Tabela 3.9 – Exibe o caso de uso Gerenciar Usuário

## 3.2 TESTE

Nesta fase devido ao curto prazo de desenvolvimento foram alterados os testes sugeridos pelo processo, ao invés de realizar os testes de caixa-branca foram realizados os testes de caixa-preta.

Os testes de *software* servem para fornecer dados de qualidade do sistema no contexto funcional, a técnica escolhida para realização dos testes do sistema foi o teste de caixa-preta também chamado de teste comportamental. Sendo uma abordagem onde os testes são derivados da especificação de programa ou de componente, cujo comportamento somente pode ser determinada estudando-se suas entradas e saídas relacionadas. (SOMMERVILLE, 2006). Os testes tentam encontrar erros de funções, de interface, de estrutura de dados e erros na inicialização e termino.

Na primeira iteração foram desenvolvidos os testes dos cadastros. Na tabela abaixo mostra os passos para o teste do Cadastro de Usuário, modelo que foi aplicado para todas as outras funcionalidades de cadastro.

| | |
|---|---|
| **Identificador** | Cadastro de Usuário |
| **Abordagem** | Caixa Preta |
| **Técnicas** | Manual |
| **Passos** | 1. Efetuar o login no sistema<br>2. Selecionar a opção para cadastrar o usuário<br>3. Tentar gravar os dados sem os campos necessários<br>4. Inserir os dados do usuário<br>5. Gravar os dados<br>6. Consultar os usuários na lista. |
| **Observações** | Caso esteja faltando algum dado aparecerá uma mensagem de erro, apontando o dado que está faltando.<br>Caso esteja tudo correto aparecerá uma mensagem de sucesso para o usuário. |

Tabela 3.10 Testes Funcionais – Cadastrar Usuário

Na segunda iteração foram desenvolvidos os testes da requisição e dos relatórios.

Na tabela abaixo mostra os passos para teste das Requisições.

| | |
|---|---|
| **Identificador** | Cadastro de Requisição |
| **Abordagem** | Caixa Preta |
| **Técnicas** | Manual |
| **Passos** | 1. Efetuar o login no sistema com um usuário que esteja vinculado a um município<br>2. Selecionar a opção para cadastrar a requisição<br>3. Tentar gravar os dados sem os campos necessários<br>4. Tente alterar o município<br>5. Inserir os dados da requisição<br>6. Gravar os dados<br>7. Consultar os usuários na lista.<br>8. Imprima a requisição |
| **Observações** | Caso esteja faltando algum dado aparecerá uma mensagem de erro, apontando o dado que está faltando.<br>Caso esteja tudo correto aparecerá uma mensagem de sucesso para o usuário.<br>Se o município for alterado será considerado um erro.<br>Na lista das requisições só poderá ter as requisições pertencentes ao município vinculado do usuário |

Tabela 3.11 Testes Funcionais – Cadastro de Requisição

Na tabela abaixo mostra os passos para teste do Relatório de Requisição.

| | |
|---|---|
| **Identificador** | Relatórios |
| **Abordagem** | Caixa Preta |
| **Técnicas** | Manual |
| **Passos** | 1. Efetuar o login no sistema<br>2. Selecionar a opção para Relatório de Requisição<br>3. Execute os dados sem nenhum filtro<br>4. Verifique se os dados estão corretos<br>5. Volte a tela de cadastro e teste com todos os filtros<br>6. Verifique se não retorna dados não filtrados |
| **Observações** | Verificar a somatória dos valores das requisições. |

Tabela 3.12 Testes Funcionais – Relatórios

## 3.3 CODIFICAÇÃO

Para a implementação do sistema foi utilizada a plataforma Java. Esta foi escolhida por ser de uso livre e por ter vários frameworks que facilitam o desenvolvimento do sistema. Nesta fase foram realizadas atividades referentes ao desenvolvimento do sistema, no qual teve como base os recursos produzidos nas fases anteriores.

O sistema foi desenvolvido utilizando o modelo MVC (*Model-View Controller*) mostrado na Figura 3.2, a primeira camada é a de apresentação, sendo esta a interface de comunicação com o usuário, a segunda camada onde ficam todas as regras de negócio do sistema e a terceira é a que faz as operações com o banco de dados, onde os dados são persistidos, consultados ou excluídos.

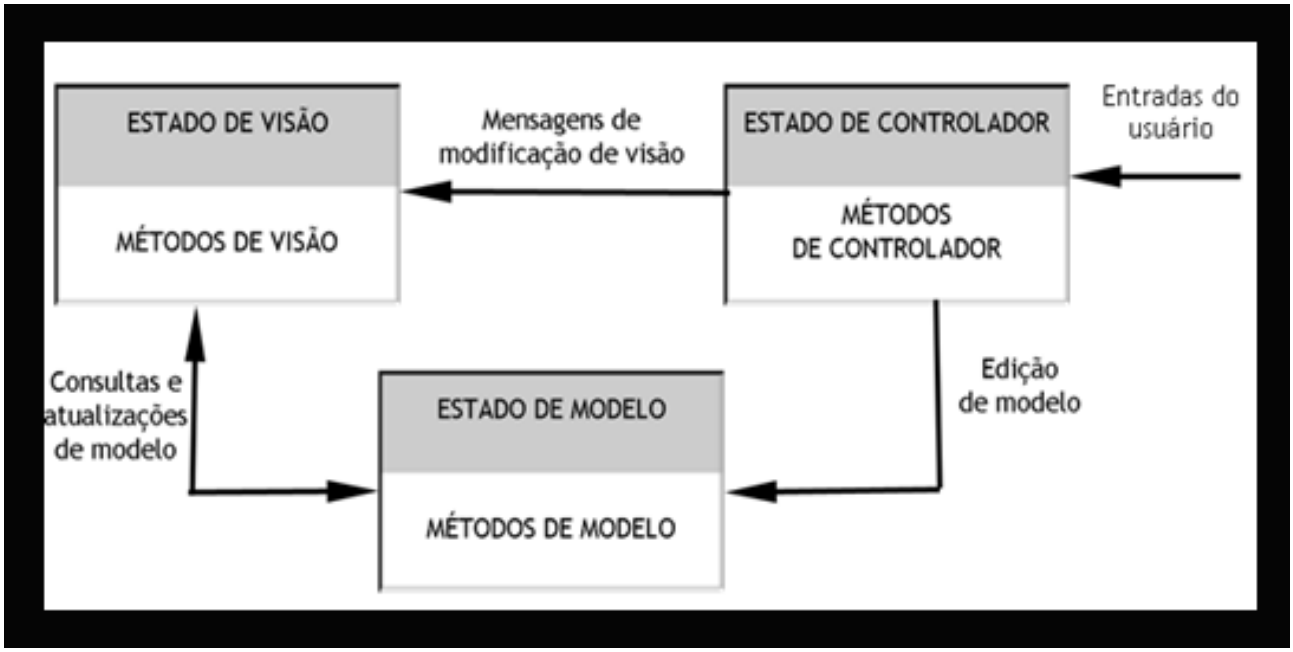


Figura 3.2: Modelo *Model-View Controller* (MVC)
Fonte: SOMMERVILLE, 2006 p. 266

O MVC atualmente é o modelo padrão de desenvolvimento utilizado na Engenharia de Software, mantendo de forma separada a lógica de negocio, classes de banco de dados e as interfaces. Na figura 3.3 mostra a divisão, utilizando o MVC, no diretório bean estão as classes que mostram a regra de negocio, a parte do modelo está dividida em dois diretórios model (classes que fazem o modelo do banco de dados) e dao (classes que fazem operações com o banco de dados), e a visão está no diretório de páginas web e *servlet*.

.


Figura 3.3: Estrutura MVC no projeto

Na primeira iteração foram desenvolvidos os cadastros do sistema dentre elas:

- Fornecedores;
- Procedimentos;
- Municípios;
- Medicamentos;
- Usuário;
- Grupo de Usuário;
- Grupo de Medicamentos;

Na segunda iteração foram desenvolvidas:

- Permissão;
- Requisições;
- Relatórios;

## 3.4 PROJETO

Na fase de projeto foi desenvolvido Modelo Entidade-Relacionamento (MER), diagrama de classe e documentado os resultados dos testes.

O Modelo Entidade-Relacionamento (MER) tem por base a percepção de que o mundo real é formado por um conjunto de objetos chamados entidades e pelo conjunto dos relacionamentos entre esses objetos. Foi desenvolvido para facilitar o projeto do banco de dados, permitindo a especificação do esquema da empresa, que representa toda a estrutura lógica do banco de dados. O MER é um dos modelos com maior capacidade semântica; os aspectos semânticos do modelo se referem à tentativa de representar o significado dos dados. O MER é extremamente útil para mapear, sobre um esquema conceitual, o significado e interações das empresas reais (SILBERSCHATZ, 1999).

A Imagem 3.5 exibe o MER do sistema, pode-se destacar nesta imagem a tabela de requisições, onde são armazenados dos dados dos usuários do Consócio Intermunicipal de Saúde, nesta tabela se relaciona com as tabelas de municípios, procedimentos, fornecedores.

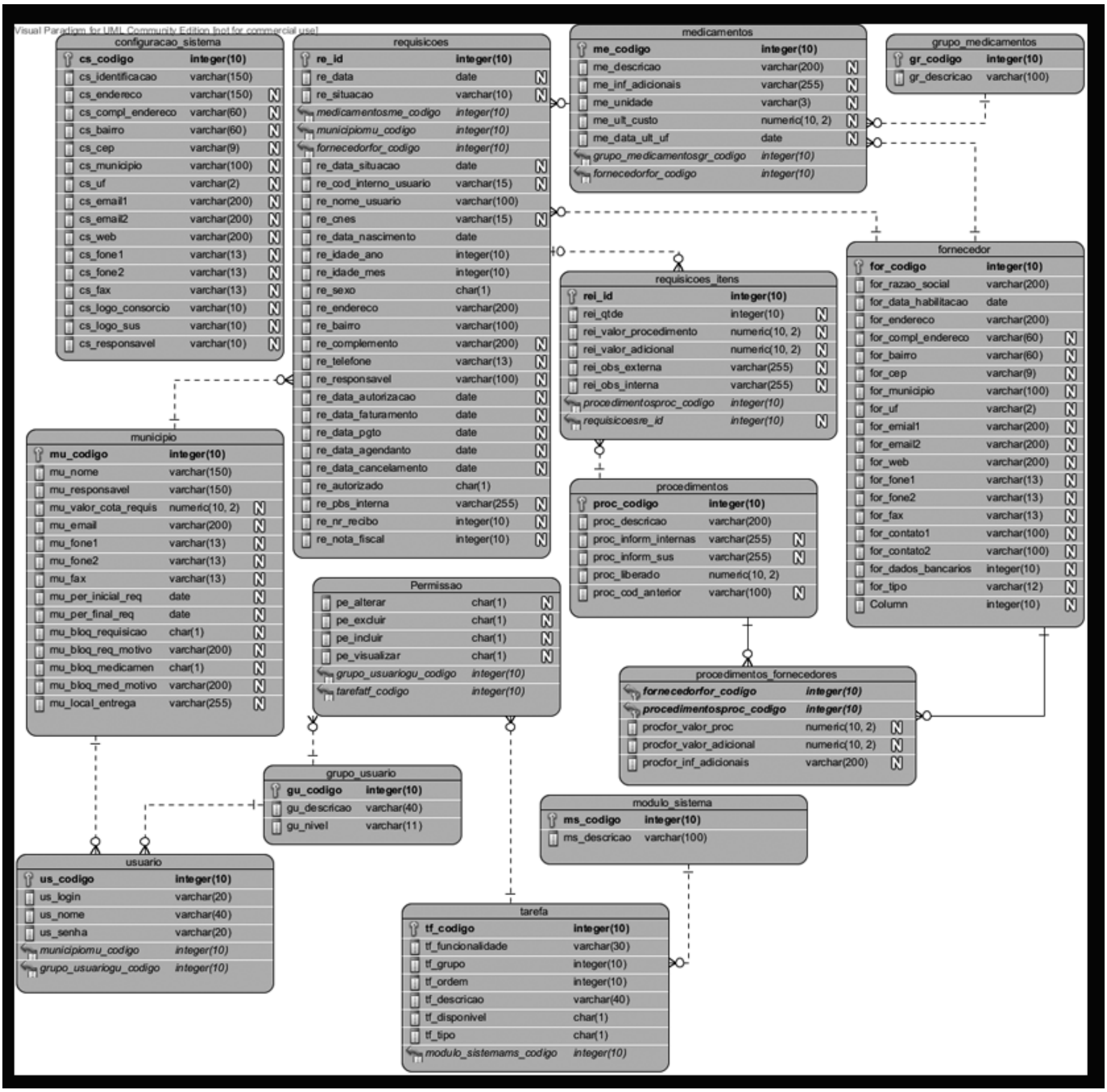


Figura 3.4: Modelo Entidade-Relacionamento

Para um melhor entendimento do relacionamento das classes do sistema foi desenvolvido um diagrama de classe mostrando como as classes se relacionam. Segundo BOOCH; RUMBAUGH; JACOBSON (2005) o Diagrama de Classes é o diagrama mais comumente encontrado em modelagem de sistemas Orientado a Objetos, é composto por classes, interfaces e seus relacionamentos. A Figura 3.5 mostra um trecho do diagrama de classe gerado, pode destacar as classes Requisicoes e RequisicoesBean, as quais interagem utilizando o método de reflexão.

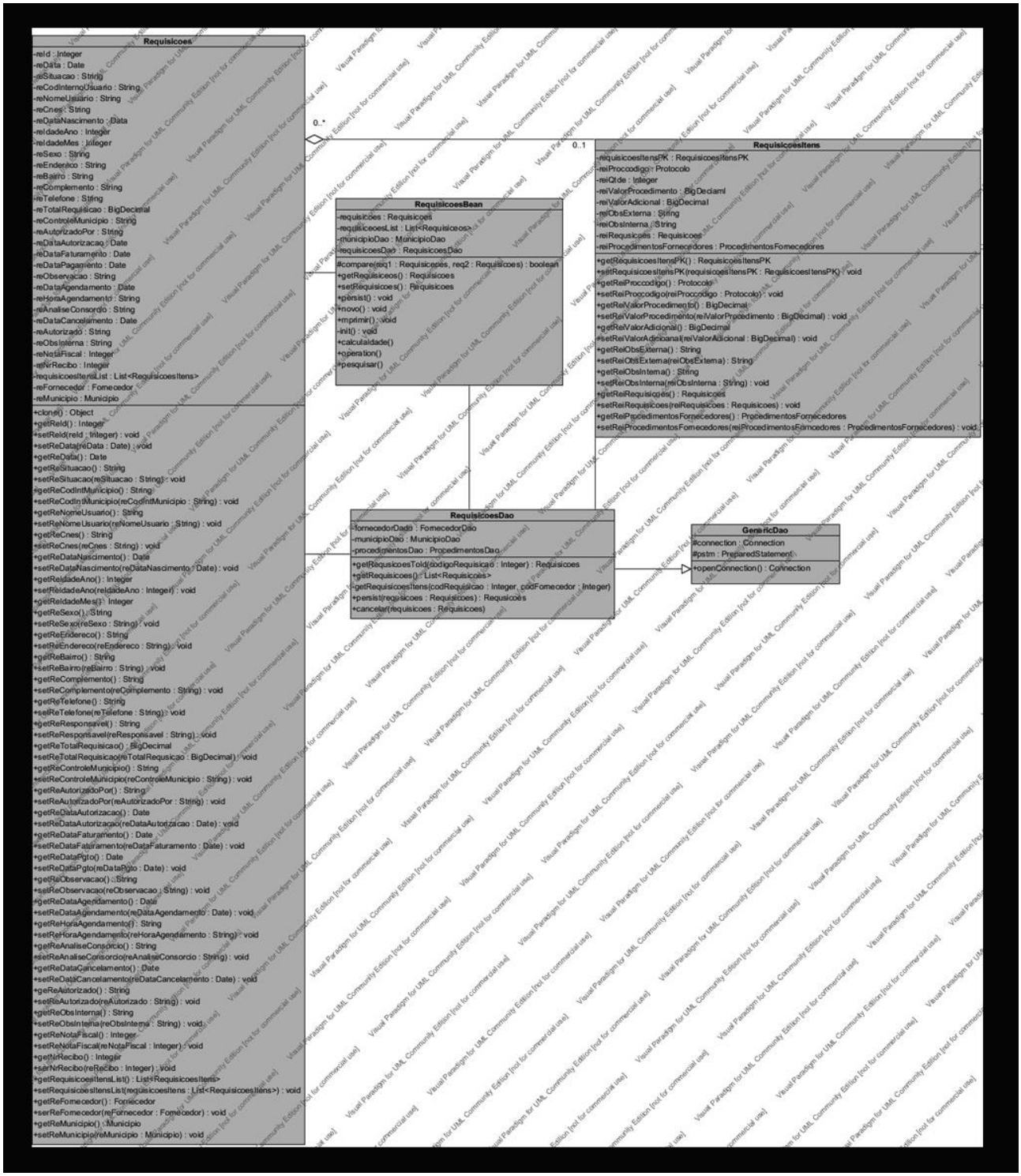


Figura 3.5: Diagrama de Classe

Na primeira iteração foi realizado teste com os cadastros do sistema, nas imagens abaixo são exibidos os resultados dos testes. A Figura 3.6 exibe a execução do teste mostrando a tentativa de salvar dos dados do procedimento sem os atributos necessários.

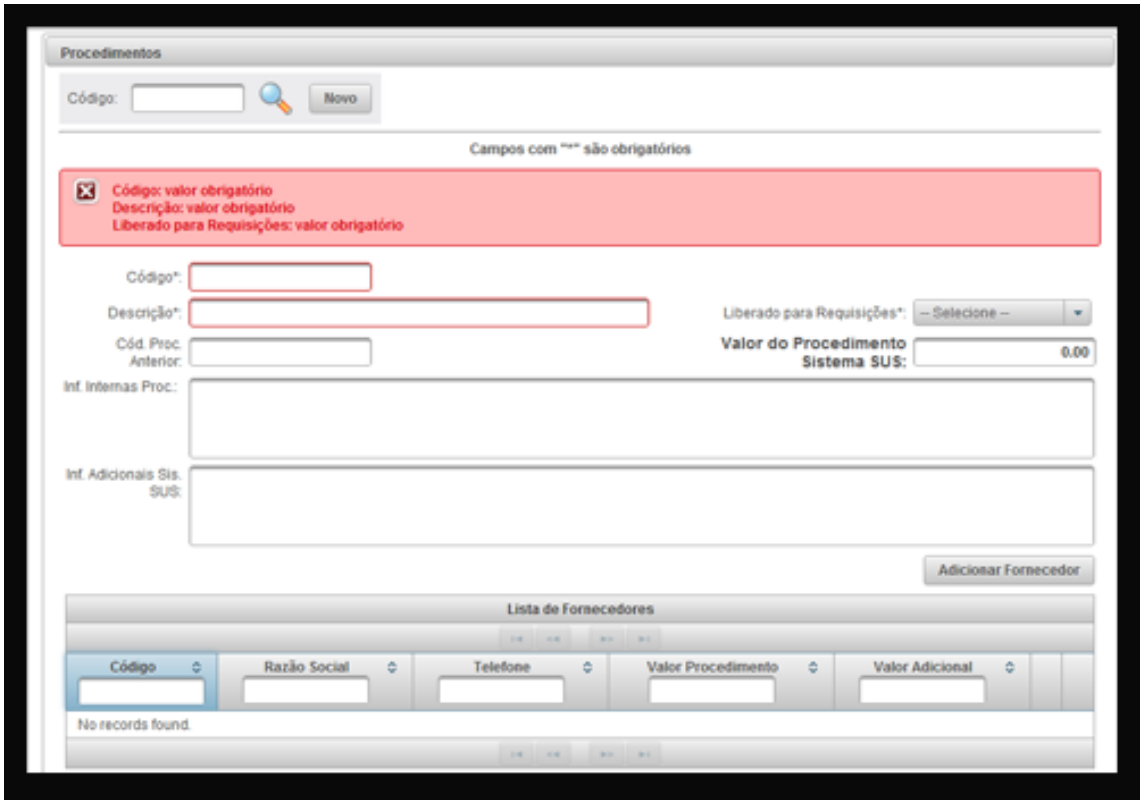


Figura 3.6: Resultado teste caixa-preta – validação de dados

A Figura 3.7 mostra a realização do teste com colocando todos os atributos necessário para realizar o procedimento, e a Figura 3.8 exibe os procedimentos cadastrados.

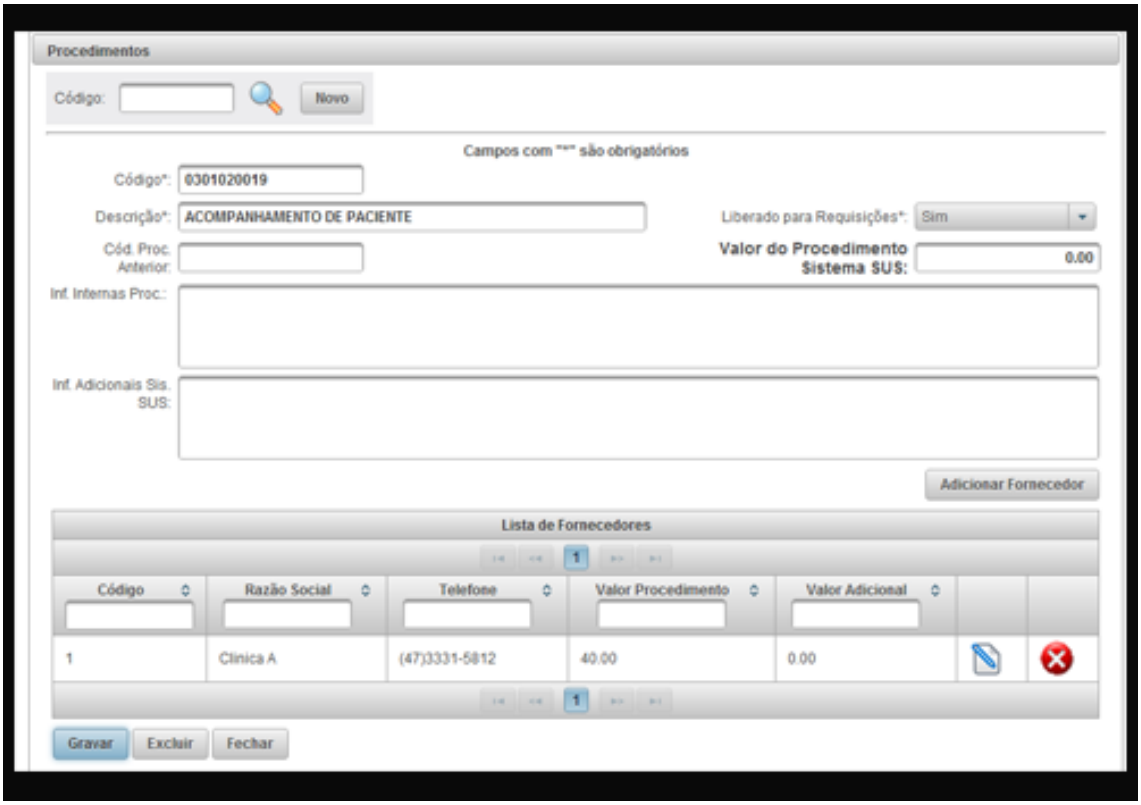

Figura 3.7: Resultado teste caixa-preta – dados completos

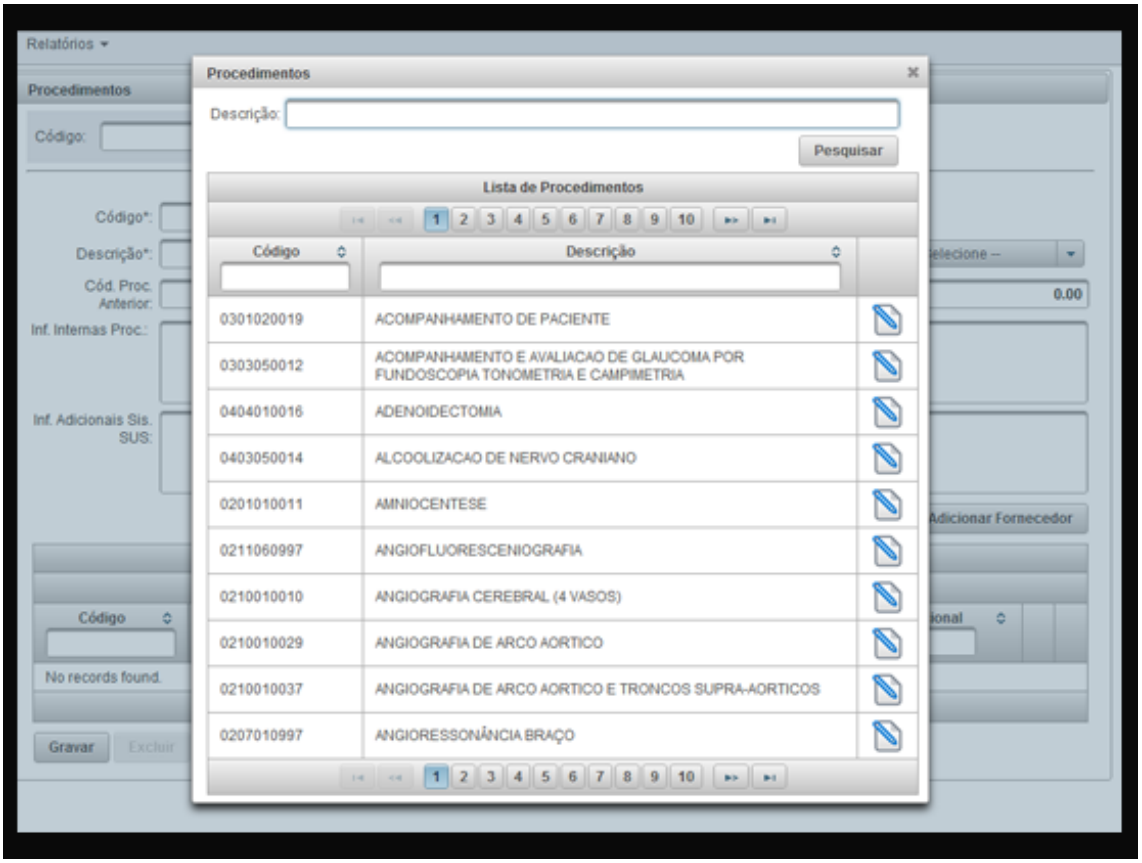

Figura 3.8: Resultado teste caixa-preta – lista de procedimentos após cadastro

# 4 CONCLUSÃO

O desenvolvimento do Sistema para Controle de Consorcio Intermunicipal de Saúde proporcionou uma oportunidade de aplicar os conhecimentos adquiridos durante a especialização.

O sistema possibilita uma maior integração dos municípios com o Consórcio Intermunicipal de Saúde, proporcionando fácil acesso aos procedimentos, medicamentos e requisições.

A utilização dos frameworks *Java Server Faces* e *PrimeFaces* possibilitaram desenvolver as interfaces ao usuário de forma mais simples. O uso processo *Extreme Programming (*XP) tornou-se um desafio, pois foi adaptado de acordo com as necessidades do projeto, dificuldades que foram superadas com consultas a livros, artigos e Internet.

A ferramenta utilizada para fazer a codificação do sistema foi o Netbeans 7.1, já a modelagem do sistema foram utilizadas as ferramenta Visual Paradigm e o SGBD escolhido foi o MySQL.

Os testes realizados para encontrar falhas foi o de caixa-preta, e essas falhas foram corrigidas.

Após a implantação do sistema, o consórcio posicionou-se afirmando que o desenvolvimento do projeto alcançou seus objetivos, pois atendeu os requisitos do Consorcio Intermunicipal de Saúde, facilitando a gestão das requisições.

## 4.1 TRABALHOS FUTUROS

O sistema poderá ser incrementado, com novas funcionalidades que podem ser adicionadas ao sistema:

- Gerenciamento do financeiro;
- Gerenciamento de consultas;
- Histórico de médico;
- Modulo de gerenciamento de avaliação dos serviços pela população.

# REFERÊNCIAS BIBLIOGRÁFICAS


BECK, K. **Programação Extrema Explicada**. Bookman, 1999.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **Unified Modeling Language User Guide. 2. ed**. Boston: Addison-Wesley Professional, 2005

ÇAĞATAY ÇIVICI. **Prime Faces Guide** [online] Disponível na Internet via: <http://primefaces.googlecode.com/files/primefaces_users_guide_3_0.pdf>. Acesso em 02 jul. 2012.

DEITEL H. M.; DEITEL P. J. **Java Como Programar 4ª Edição** Lang Lisboa – 4 Ed. Porto Alegre: Bookman, 2003.

GEARY D.; HORSTMANN C **Core Java Server Faces -** São Paulo Alta Books, 2005

GUEDES, G. T. A.; **UML Uma Abordagem Pratica**. São Paulo: Novatec, 2005.

MySQL Disponível em:
< http://dev.mysql.com/doc/> Acesso em: 20 jun. 2012

SILBERSCHATZ, A.; **Sistema de Banco de Dados**. São Paulo: Makron Books, 1999.

SOMMERVILLE, I. **Engenharia de Software**. 7. ed. [S.l]: São Paulo: Addison-Wesley, 7 ed. 2006

ORACLE: Disponível em:
http://www.oracle.com/technetwork/java/index-141671.html> Acesso em: 02 jul. 2012