UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

SISTEMA DE GERENCIAMENTO DE LOCADORA - SISLOC

Autor: _____
Ramon de Menezes Pedrosa Andrade

Orientador: _____
Prof. Antônio Cláudio Gómez de Sousa, M.Sc.

Examinador: _____
Prof. Aloysio de Castro P. Pedroza, D.Sc.

Examinador: _____
Prof. Marcelo Luiz Drumond Lanza, M.Sc.

DEL
Dezembro de 2006

Dedicatória

Dedico este trabalho primeiramente aos meus pais, que sempre me apoiaram em todas as decisões tomadas na minha vida, ao apoio familiar e financeiro. Dedico também às pessoas que de alguma forma contribuíram para que este longo caminho fosse concluído, entre elas os meus grandes amigos de faculdade, minha namorada, minha irmã e meu irmão.

Agradecimento

Agradeço primeiramente a Deus que me deu forças para nunca desistir diante dos grandes desafios impostos pela faculdade.

Aos meus familiares e minha namorada que sempre estiveram do meu lado me dando apoio e incentivo.

Aos meus amigos da faculdade que serão lembrados pela união que sempre tivemos ao longo da faculdade.

Aos professores e funcionários do DEL, pelo trabalho de ensino realizado.

Resumo

No mundo atual, percebemos a grande expansão da indústria cinematográfica nas últimas décadas, com o desenvolvimento de grandes produções. Visto a este fato e analisando a expectativa da entrada da era digital na TV brasileira, com a venda de televisores digitais, podemos perceber que o ramo de locadoras pode sofrer uma grande expansão no mercado brasileiro, já que assistiremos no conforto de nossas casas filmes com qualidade digital de som e imagem.

Pensando nesta expansão propus uma solução que cria um novo modelo de negócio no mercado de locadoras. Esse modelo de negócio envolve justamente uma maior demanda pelas pessoas por locações de filme e com o custo benefício mais apropriado para a quantidade de filmes locados. A solução apresentada será um sistema que se baseia neste novo modelo de negócio e que envolve o gerenciamento da locadora e a divulgação dos filmes para seus clientes.

Palavras-chaves

Web, locadora de filmes, modelo de negócio, Hibernate, WebWork, MySQL, Java, J2EE.

Índice

Índice de figuras

Siglas

API	*Application Programming Interface*
AJAX	*Asynchronous JavaScript and XML*
BO	*Bussiness Object*
CORBA	*Common Object Request Broker Architecture*
CSS	*Cascading Style Sheet*
DAO	*Data Access Object*
DBCP	*Data Base Connection Pool*
ERS	Especificações de Requisitos de Software
GUI	*Graphical User Interface*
HD	*Hard Disk*

HQL	*Hibernate Query Language*
HTTP	*Hyper Text Transfer Protocol*
IDE	*Integrated Development Environment*
IoC	*Inversion of Control*
J2EE	*Java 2 Enterprise Edition*
JDBC	*Java Database Connectivity*
JDK	*Java Development Kit*
JDT	*Java Development Tools*
JSP	*Java Serves Pages*
JTA	*Java Transaction API*
JVM	*Java Virtual Machine*
MVC	*Model View Controller*
OGNL	*Object-Graph Navigation Language*
RPC	*Remote Procedure Call*
SisLoc	Sistema de Gerenciamento de Locadoras
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	*Structured Query Language*
VO	*Value Object*
XML	*Extensible Markup Language*

1 Introdução

O projeto denominado 'Sistema de Gerenciamento de Locadora (SisLoc)' tem como finalidade criar uma nova oportunidade de negócio no ramo de locadoras. O SisLoc se baseia em um modelo de negócio diferenciado e possui funcionalidades que tornam simples o gerenciamento e divulgação da locadora.

O SisLoc além de prover todas as funcionalidades que uma locadora necessita, tais como cadastro de associados, cadastro de filmes, etc. ainda permitirá que sejam cadastrados funcionários, lojas, atores, diretores e geração de diversos tipos de relatórios tais como de mídias alugadas, relatório de caixa mensal.

O SisLoc funcionará a partir de um modelo de negócio novo em relação ao modelo de negócio tradicional das locadoras atuais. O modelo de negócio que o SisLoc se baseia foi desenvolvido a partir da crescente expansão da indústria cinematográfica e a procura cada vez maior das pessoas pelo lazer dentro de casa.

Tendo em vista essa procura pela população e a necessidade de um novo modelo de locadora para atender essa demanda, o SisLoc adotará um modelo de negócio baseado em planos mensais. Assim cada associado titular possuirá um plano mensal, e esse plano lhe dará direito a sempre ter consigo certa quantidade de filmes, por tempo indeterminado.

Assim sempre que o associado desejar alugar filmes, o sistema fará a verificação da quantidade de filmes que o associado possui em seu poder e verificará se será possível o associado alugar os filmes que deseja. Caso não seja possível o sistema não aceitará que o usuário alugue os filmes, a não ser que o usuário retorne os filmes que ele possui consigo.

O SisLoc será um sistema Web, desenvolvido de forma a ter uma interface limpa e fácil de usar e que permita uma automação dos processos mais comuns que ocorrem em uma locadora, que são as locações de filmes.

Para isso foram criados formas de pesquisas diferenciadas tais como pesquisa de associados por nome, telefone e código de barras que o sistema gerará. A pesquisa por código de barras é bastante importante para o processo de automação, pois com uma máquina leitora de código de barras o processo de digitar se torna obsoleto.

Além do sistema de gerenciamento o sistema permite também a divulgação dos filmes cadastrados na locadora através do site do sistema. O site também é baseado na plataforma J2EE [9] e utiliza os dados do cadastro do filme realizados pelo sistema de gerenciamento, para divulgar aos associados os filmes que estão disponíveis na loja.

Para apresentar o que foi implementado, este documento está dividido em capítulos da seguinte forma: no capítulo 2, é dada uma visão geral da solução, mostrando os principais componentes do sistema, explicando sucintamente o papel de cada um, e descrevendo todo o processo de desenvolvimento de cada um dos módulos que compõem esse projeto.

No capítulo 3 são explicados os conceitos de base para o desenvolvimento de um aplicativo utilizando J2EE e algumas tecnologias inerentes a essa filosofia de programação Java, como o Hibernate [7] e o WebWork [14].

No capítulo 4, o projeto é detalhado, sendo o sistema dividido em dois módulos, no qual o primeiro módulo é o módulo de gerenciamento da locadora, onde são especificadas as suas funcionalidades e posteriormente são apresentadas as funcionalidades do módulo de divulgação.

No capítulo 5, são apresentadas as conclusões deste trabalho e são feitas algumas propostas de trabalhos futuros em cima deste projeto final.

A arquitetura do sistema é mostrada na figura a seguir:

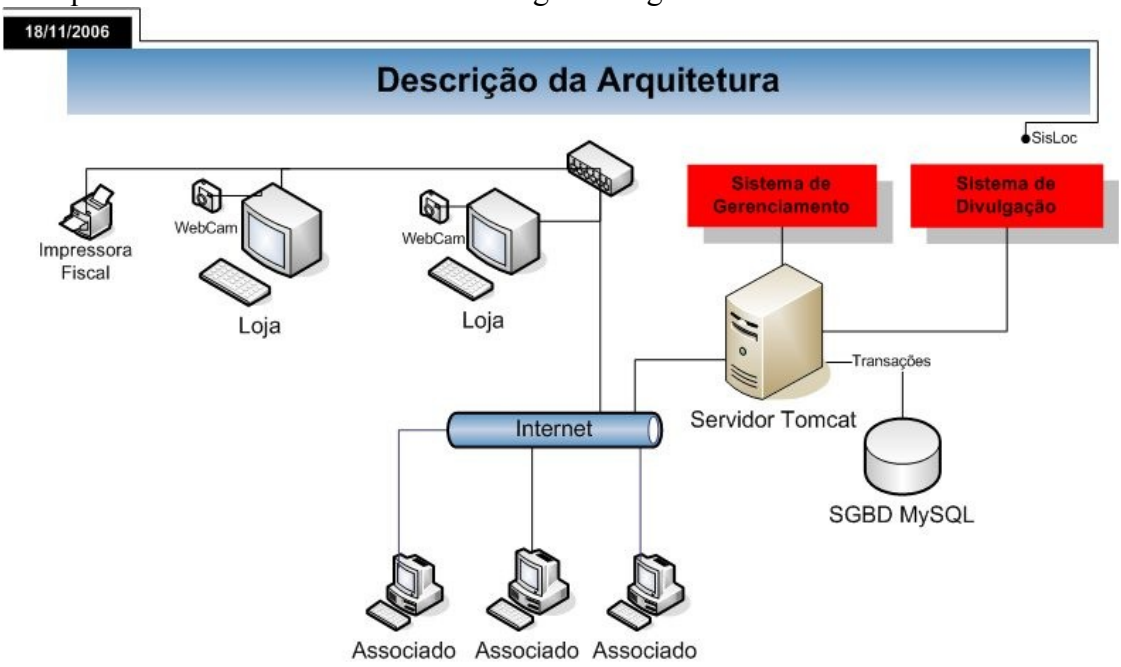

Figura 1: Descrição da Arquitetura do Sistema

Há ainda um apêndice, que compõe o anexo deste projeto. Ele é composto pelas Especificações de Requisitos de Software (ERS), cuja norma é baseada na norma ANSI/IEEE 830 simplificada [12].

A ERS está dividida em 3 seções e representa um maior detalhamento da solução proposta. Na primeira seção é feita uma breve introdução sobre o software desenvolvido, sendo apresentada sua finalidade e escopo. Na seção 2 é feita a descrição geral do produto, sendo uma complementação as descrições feitas nos capítulos da documentação anteriores ao apêndice. Na última seção, são detalhados os requisitos de software, mostrando-se o diagrama de entidades e relacionamentos (e seu dicionário de dados) e a análise através de Diagramas de Casos de Uso, Diagrama de Seqüência e Diagrama de Classes.

2 Desenvolvimento

Neste capítulo será apresentado todo o trabalho realizado durante o projeto. Serão explicadas as fases do desenvolvimento do projeto, bem como as ferramentas utilizadas para que o projeto fosse concluído. As ferramentas serão listadas desde o ambiente de desenvolvimento até as API's (*Application Programming Interface* - Interface de Programação de Aplicativos) utilizadas ao longo do desenvolvimento.

Como ambiente de desenvolvimento, foi escolhida a ferramenta Eclipse [6]. Trata-se de uma IDE (*Integrated Development Environment* - Ambiente Integrado de Desenvolvimento) de código aberto e de maior popularidade para o desenvolvimento de aplicações Java. O Eclipse é uma IDE baseada em *plugins*, onde cada *plugin* adicionado oferece uma contribuição ao aplicativo. Assim foi adicionado ao Eclipse o *plugin* do Tomcat que disponibiliza meios de inicializar e parar o Tomcat pelo Eclipse além de possibilitar o modo *debug*;

A partir da decisão de utilizar o Eclipse como ambiente de desenvolvimento foi escolhido o servidor onde a aplicação Java desenvolvida seria instalada e disponibilizada para acesso Web. Assim o servidor escolhido foi o servidor de aplicações Java da *Apache Foundation*, o Tomcat.

O Tomcat [15] é um servidor de aplicações Java para Web. É distribuído como software livre e desenvolvido como código aberto dentro do conceituado projeto *Apache Jakarta* e oficialmente endossado pela *Sun* como a implementação de referência para as tecnologias *Java Servlet* e *Java Server Pages* (JSP). O Tomcat é robusto e eficiente o suficiente para ser utilizado mesmo em um ambiente de produção.

Tecnicamente o Tomcat é um *container* Web, cobrindo parte da especificação J2EE com tecnologias como *Servlet* e JSP e tem a capacidade de atuar também como servidor Web/HTTP, ou pode funcionar integrado a um servidor Web dedicado como o Apache httpd ou o Microsoft IIS.

O desenvolvimento inicial se deu em cima do módulo principal: o módulo de gerenciamento de locadoras, cujo *front-end* foi desenvolvido totalmente usando JSP's. Como *back-end*, desenvolveram-se várias classes Java, seguindo o conceito de arquitetura MVC, que será explicado mais adiante. Seguindo este conceito, foi necessária a utilização do *framework* WebWork, sobre o qual foi necessário um estudo profundo sobre aplicabilidade e configurações específicas.

Assim, os primeiros passos deste projeto foram no sentido de se criar as configurações iniciais de modo que ele pudesse funcionar no *application server* escolhido: o Apache Tomcat. Essa configuração é feita através dos seguintes arquivos xml:

- *context.xml*: define todo o contexto no qual o projeto, incluindo o path da aplicação relativo ao Tomcat (/sisloc).
- *web.xml*: faz o mapeamento do arquivo de configuração do WebWork, das *taglibs* utilizadas (WebWork e DisplayTag [4]) e define algumas características como extensão das *actions* (*.action*, para o caso deste projeto) e o arquivo de "boas-vindas" (*login-form.jsp)*.
- *xwork.xml*: faz o mapeamento de todas as *actions* envolvidas no projeto. Ele mapeia o path relativo (`/sisloc/`*`login`*, por exemplo) em uma classe (`com.mj.sisloc.`*`action`*`.admin.LoginAction`, para o exemplo)

Um trecho do arquivo *xwork.xml* é mostrado a seguir:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
    <include file="WebWork-default.xml"/>

    <package name="default" extends="WebWork-default">

        <global-results>
            <result name="login"
type="dispatcher">loginForm.jsp</result>
            <result name="expired"
type="dispatcher">/admin/sessaoExpirada.jsp</result>
        </global-results>

        <action name="loginForm"
class="com.mj.sisloc.action.admin.LoginFormAction">
            <interceptor-ref name="defaultStack"/>
            <result name="success"
type="dispatcher">loginForm.jsp</result>
        </action>

        <action name="login"
class="com.mj.sisloc.action.admin.LoginAction">
            <interceptor-ref name="hibernate"/>
            <interceptor-ref name="defaultStack"/>
            <result name="success" type="chain">adminHome</result>
            <result name="error"
type="dispatcher">loginForm.jsp</result>
            <result name="trocarSenha"
type="chain">trocarSenhaForm</result>
        </action >

        <action name="logoff"
class="com.mj.sisloc.action.admin.LogoffAction">
            <interceptor-ref name="defaultStack"/>
            <result name="success" type="chain">loginForm</result>
            <result name="error"
type="dispatcher">loginForm.jsp</result>
        </action>

    </package>
    <!—Inclusao dos outros arquivos xwork-package.xml -->
    <include file="xwork-ator.xml"/>
    <include file="xwork-cobranca.xml"/>
    <include file="xwork-diretor.xml"/>
    <include file="xwork-midia.xml"/>
    <include file="xwork-associado.xml"/>
    <include file="xwork-plano.xml"/>
    <include file="xwork-query.xml"/>
    <include file="xwork-aluguel.xml"/>
    <include file="xwork-relatorio.xml"/>
    <include file="xwork-loja.xml"/>
    <include file="xwork-filme.xml"/>
    <include file="xwork-usuario.xml"/>
    <include file="xwork-listadesejo.xml"/>
</xwork>
```

A estrutura desse arquivo será explicada na seção *framework* WebWork *3.1.4*. Outra importante configuração que foi necessária adicionar ao projeto foi configuração de *logging*. *Logging* é uma técnica de depuração que exibe e/ou grava as mensagens que a aplicação envia para indicar o seu comportamento atual. Como essa funcionalidade é de suma importância tanto nas fases de desenvolvimento e testes quanto na fase de produção de um projeto, toda aplicação bem desenvolvida dispõe de tal funcionalidade. Para isso foi utilizada no projeto a API de *log* especificada e indicada pela Apache, a Log4J [11].

O Log4J é um projeto *open source* baseado no trabalho de diversos autores. Ele permite ao desenvolvedor controlar, de maneira flexível, cada saída de *log*. Com ele, pode-se ativar o *log* em tempo de execução sem modificar os binários da aplicação, sendo necessário apenas editar um arquivo de configuração.

Para se usar o Log4J deve-se ter um considerável planejamento e esforço, pois após colocarmos instruções de *log* dentro do código de um sistema, é importante que não seja preciso modificá-las manualmente.

Esse planejamento é feito na configuração, esta é, talvez, a parte mais importante ao utilizar o Log4J. A configuração pode ser feita por programação, entretanto, fica claro que é obtida uma flexibilidade muito maior ao usarmos outras duas formas possíveis de configuração: um arquivo XML ou um arquivo de propriedades (no formato chave=valor).

Configurar o Log4J é de fato ativar os *loggers* para mostrar as mensagens de *log* nos *appenders* escolhidos usando *layouts* determinados pelo desenvolvedor. Os *loggers* podem ser criados e configurados em qualquer ordem, em particular um *logger* irá encontrar e agrupar seus descendentes mesmo que ele seja instanciado depois deles.

Isto tudo pode ser feito em um arquivo que deve ser chamado de *log4j.properties*, desta forma a configuração é feita apenas uma vez e é carregada na inicialização da aplicação. Este arquivo deve ser colocado no *classpath* da aplicação.

```
log4j.rootLogger=info, stdout, WebWork
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%-5p: %d{HH:mm:ss} (%F:
%L) - %m%n

log4j.appender.WebWork=org.apache.log4j.DailyRollingFileAppender
log4j.appender.WebWork.DatePattern='.'yyyy-MM-dd
log4j.appender.WebWork.file=/usr/local/tomcat/files/sisloc/logs/sisloc
.log
log4j.appender.WebWork.layout=org.apache.log4j.PatternLayout
      log4j.appender.WebWork.layout.ConversionPattern=%5p:    %d{dd-MM-
yyyy HH:mm:ss} (%F:%L) - %m%n
```

O Log4j possui três componentes principais: *loggers*, *appenders* e *layouts*. Eles trabalham juntos para que os desenvolvedores façam o *log* de suas mensagens de acordo com o tipo e o nível delas, e controle em tempo de execução como estas mensagens são formatadas e onde são reportadas.

A primeira e principal vantagem de qualquer API de *log* sobre o *System.out.println* está em sua habilidade de desabilitar certos níveis de *log* (quando estes não são necessários) enquanto os demais níveis continuam funcionando normalmente. O Log4J oferece cinco níveis de *log* em sua hierarquia: *DEBUG, INFO, WARN, ERROR e FATAL*. O nível da hierarquia de *log* será informado quando da chamada ao método de *log*.

Um *appender* é uma saída onde são mostradas as mensagens, o console, algum arquivo, um servidor remoto de *sockets*, etc. Um *logger* pode ter mais de um *appender*. Por exemplo, podemos mandar as mensagens de *log* para o console e para um arquivo ao mesmo tempo. Além disso, podemos mostrar níveis diferentes de mensagens nos *appenders* diferentes, ou seja, mandar para o console todas as mensagens enquanto que para o arquivo podemos salvar apenas as mensagens de nível WARN, por exemplo.

Layout é o formato como a mensagem vai ser mostrada. Este formato pode incluir a data da mensagem, qual o método que contém o *log*, qual linha, classe, arquivo, etc.

Outra configuração necessária foi a configuração de *build* da aplicação que é utilizada para automatizar a construção e a instalação no servidor. Para utilizar este recurso foi utilizado a ferramenta Ant [2] para Java. O Ant é um projeto *opensource*, produzido pelo grupo Jakarta da Fundação Apache e é uma ferramenta para a automação de *builds* de projetos, por exemplo, atualiza o classpath, compila o código separando os arquivos *.java* e os *.class* em diretórios distintos, gera *javadoc* do projeto, configura e executa a aplicação.

O Ant trabalha com arquivos XML chamados de *buildfiles*, onde os dados desse arquivo são interpretados pelo Ant, para que possa executar as tarefas que estão descritas. O *buildfile* é um arquivo XML geralmente chamado de *build.xml*.

O Ant adiciona portabilidade na automação de *builds* e a partir das configurações do Ant podemos compilar todas a nossa aplicação, adicionar as dependências dos *.jar*'s necessários ao funcionamento da aplicação, criar os arquivos *.war* que serão ao final do *build* instalados na pasta do servidor *Java* utilizado.

Por fim, configuramos a configuração da conexão com o banco de dados. Para isso, como utilizaríamos o Hibernate como *framework* de conexão e persistência dos dados da aplicação, seria necessário configurar o arquivo *hibernate.cfg.xml* que possibilitaria a conexão com o banco MySQL [17] da aplicação.

Ao final da configuração e instalação de todos esses arquivos, foi completada a fase de instalação do ambiente de desenvolvimento e podemos assim começar a fase de desenvolvimento das funcionalidades oferecidas pelo sistema de gerenciamento de locadora e do sistema de divulgação da locadora. O primeiro sistema a ser desenvolvido foi o sistema de gerenciamento, que contém as principais funcionalidades da aplicação e que permite o gerenciamento da locadora.

Ao início da fase de desenvolvimento, foram criados os primeiros objetos que correspondem aos objetos de persistência dos dados do banco, os VO's (*Value Objects*), objetos que carregam os dados da aplicação. De forma a seguir a padronização MVC, os primeiros pacotes e pastas foram criados seguindo o modelo descrito.

Após os VO's terem sido criados e mapeados referenciando as tabelas do banco, a partir dos arquivos de configuração do Hibernate, foram criados as classes DAO's (*Data Access Object*), que são as classes dos objetos que acessam os dados do banco. A partir da criação dessas classes puderam-se realizar os primeiros testes de conexão e persistência dos dados do banco. Nesse primeiro momento a aplicação estava funcionando somente em modo *debug*, onde se verificava em tempo de execução a persistência dos dados de algumas tabelas do banco e as mensagens de *logs* do sistema.

Com a verificação de que as configurações do banco, configurações de *logger*, mapeamentos das entidades relacionais estavam funcionando corretamente poderíamos começar a desenvolver as funcionalidades do sistema de gerenciamento.

Como descrito na Especificação de Requisitos de Software do sistema, o sistema de gerenciamento e o sistema de divulgação foram divididos em módulos, de forma a organizar as tarefas e as prioridades de cada módulo no projeto. Essa organização

permitiu que o desenvolvimento fosse realizado de forma organizada, levando em consideração a prioridade de cada módulo em relação aos demais.

Essa prioridade foi definida tendo como base a importância de cada módulo na aplicação, verificando as suas dependências em relação aos outros módulos. Com essa definição ao final do desenvolvimento de cada módulo, este poderia ser testado plenamente, sem a preocupação de que dependesse de outro módulo, pois na dependência de outro módulo, este outro já estaria desenvolvido, pois uma análise prévia das dependências e organização dos módulos já havia sido realizada.

Cada módulo desenvolvido foi sendo adicionado aos seus pacotes, seguindo a padronização MVC. A camada *Model* é a camada de lógica da aplicação e, portanto os pacotes referentes a essa camada foram divididos em quatro grupos, o grupo dos pacotes que contêm classes dos *Value Objects*, o grupo que contêm as classes BO (*Business Object*), que são os objetos que contêm os métodos de negócio da aplicação, o grupo que contêm as classes DAO e o grupo que contêm as classes *actions* da aplicação.

A camada *Controller* é gerenciada pela *framework* WebWork através dos seus arquivos *xwork.xml* que contêm o mapeamento dos fluxos da aplicação. Por fim a camada *View* é referenciada pelos diretórios que contêm os arquivos *.jsp* que exibem para o usuário as informações das telas do sistema.

Ao final do desenvolvimento do módulo de gerenciamento, começou-se a fase de desenvolvimento do módulo de divulgação da locadora. Esse módulo foi desenvolvido mais rapidamente do que o módulo de gerenciamento visto que este possuía uma quantidade bem menor de funcionalidades e também as funcionalidades destes módulos envolviam na sua maioria formas de pesquisa dos dados do banco, diferentemente do módulo administrativo que possuía funcionalidades de inserção e remoção.

O módulo de divulgação utilizou as mesmas ferramentas de desenvolvimento do módulo administrativo e também seguiu a padronização MVC. Assim o desenvolvimento foi consideravelmente mais rápido, pois grande parte da implementação dos códigos das diversas camadas (Modelo, Controle, Visão) já estavam prontos, já que haviam sido implementados no módulo de gerenciamento.

O mesmo critério de divisão em módulos também foi adotado para o caso do sistema de divulgação, onde se mapeou os submódulos mais críticos quanto a dependência em relação aos outros submódulos, o que garantiu, como na fase de desenvolvimento do módulo de gerenciamento, uma implementação com poucos atrasos.

O módulo de divulgação visa a divulgar para o público alvo da locadora informações relevantes e prover funcionalidades que permitam uma interação virtual do associado com a locadora e para isso possui funções como: exibir informações dos filmes cadastrados, diversas formas de pesquisa de filmes, visualização do perfil do associado (quando este se loga através do sistema), criação de lista de desejo dos filmes que o associado queira alugar, envio de e-mail, exibição do histórico de filmes que o associado alugou, etc.

Por ser um módulo voltado para o público a sua interface com o usuário deve ser mais elegante e limpa do que a interface do módulo de gerenciamento. Assim, durante o desenvolvimento a utilização propriedades de arquivo CSS (Folha de Estilo em Cascata) foi mais largamente utilizado. A CSS é um mecanismo simples para adicionar estilos (exemplos: fontes, cores, espaçamentos) em documentos Web, ou seja, a *CSS* é um padrão de formatação (Web Standards) para páginas que permite ir além das limitações impostas pelo HTML.

Web Standards é um conjunto de normas, diretrizes, recomendações, notas, artigos, tutoriais e afins de caráter técnico, e destinados a orientar fabricantes,

desenvolvedores e projetistas para o uso de práticas que possibilitem a criação de uma Web acessível a todos, independentemente dos dispositivos usados ou de suas necessidades especiais.

Foi proposto pelo World Wide Web Consortium, W3 Consortium, o qual é uma comissão que define os padrões de programação para a WWW.

O uso de folhas de estilo permite:

- maior versatilidade na programação do layout de páginas, sem aumentar o seu tamanho.
- maior controle sobre os atributos de uma página, como tamanho e cor das fontes, espaçamento entre linhas e caracteres, margem do texto, caixas de texto, botões de formulário, entre outros.
- a utilização de "layers", permitindo a sobreposição de objetos, textos e imagens, em camadas.
- modificar rapidamente todo o "layout" de um "site", ou de um certo grupo de formatação (*Exemplo*: alterar a cor de todos os links).

3 Fundamentação Teórica

Neste capítulo, serão apresentadas as principais tecnologias de desenvolvimento, utilizadas no projeto.

3.1.1 J2EE

O J2EE (*Java 2 Enterprise Edition*) é uma plataforma de programação de computadores que faz parte da plataforma Java. Ela é voltada para aplicações multicamadas, baseadas em componentes que são executados em um servidor de aplicações. A plataforma J2EE é considerada um padrão de desenvolvimento já que o fornecedor de software nesta plataforma deve seguir determinadas regras se quiser declarar os seus produtos como compatíveis com J2EE. Ela contém bibliotecas desenvolvidas para o acesso à base de dados, RPC, CORBA, etc. Devido a essas características a plataforma é utilizada principalmente para o desenvolvimento de aplicações corporativas.

A plataforma J2EE contém uma série de especificações, cada uma com funcionalidades distintas. Dentre elas, tem-se:

- JDBC (*Java Database Connectivity*): utilizado no acesso a bancos de dados.
- *Servlets*: são utilizados para o desenvolvimento de aplicações Web com conteúdo dinâmico. Ele contém uma API que abstrai e disponibiliza os recursos do servidor Web de maneira simplificada para o programador.
- JSP (*Java Server Pages*): uma especialização do servlet que permite que conteúdo dinâmico seja facilmente desenvolvido.
- JTA (*Java Transaction* API): é uma API que padroniza o tratamento de transações dentro de uma aplicação Java.
- EJBs: utilizados no desenvolvimento de componentes de software. Eles permitem que o programador se concentre nas necessidades do negócio do cliente, enquanto questões de infra-estrutura, segurança, disponibilidade e escalabilidade são responsabilidade do servidor de aplicações.
- JCA (*Java Connector Architecture*): é uma API que padroniza a ligação a aplicações legadas.

Neste projeto, serão utilizados os conceitos de JDBC, *Servlets* e JSP.

3.1.2 Design Patterns (Padrões de Projeto)

Os padrões de projeto de software ou padrões de desenho de software, também muito conhecido pelo termo original em inglês *Design Patterns*, descrevem soluções para problemas recorrentes no desenvolvimento de sistemas de software orientados a objetos. Um padrão de projeto estabelece um nome e define o problema, a solução, quando aplicar esta solução e suas conseqüências.

Os padrões de projeto visam facilitar a reutilização de soluções de desenho - isto é, soluções na fase de projeto do software, sem considerar reutilização de código. Também acarretam um vocabulário comum de desenho, facilitando comunicação,

documentação e aprendizado dos sistemas de software. No projeto, como forma a otimizar e padronizar o código foi-se utilizado os seguintes padrões de projeto abaixo:

Business Delegate

O padrão *Business Delegate* é uma abstração dos serviços de negócio na camada cliente, de modo a aumentar o desacoplamento entre as camadas. É especialmente útil quando se tem uma API de negócios pouco madura, sujeito às mudanças freqüentes.

A camada cliente se relaciona com os serviços de negócio por meio de chamadas ao *Business Delegate*. Uma das grandes vantagens do *Business Delegate* é que seus clientes não precisam se preocupar com quais são os serviços de negócio e nem como devem ser acessados para realizar uma tarefa que faça sentido para seu cliente.

Singleton

O padrão *Singleton* é utilizado quando se quer garantir que somente uma instância de uma classe pode existir. Geralmente opta-se por este padrão quando se quer substituir uma classe, geralmente utilitária, que apresenta vários métodos estáticos e não deve apresentar distintos estados por instância.

O construtor do *Singleton* deve ser privado, de modo que nenhuma classe externa possa criar uma instância do *Singleton* e garantir que o acesso ao mesmo será somente por meio de um método estático assessor, geralmente chamado de *getInstance*. Toda lógica de instanciação e inicialização fica encapsulada no método assessor.

DAO

O padrão DAO (*Data Access Object*) é utilizado quando se desejam abstrair da camada de negócios as fontes de dados e persistência.

Um objeto DAO provê métodos para adicionar, editar, excluir, consultar objetos de dados independentemente da forma de persistência dos dados. Dessa forma, a camada de negócios acessa os DAO para obter dados persistidos, sem se comprometer com uma forma específica de persistência.

Para aplicar o padrão DAO, define-se uma interface que é implementada por diferentes classes, cada uma de acordo com o repositório a que se referem.

Factory

O padrão *Factory* é utilizado quando se quer abstrair a criação de objetos do cliente que os utilizará. O cliente utiliza o objeto fabricado por meio de uma classe base ou interface, sem se preocupar com qual a real classe derivada ou implementação da instância fabricada.

A *Factory* é responsável por determinar qual a classe derivada ou implementação de uma interface deve ser instanciada, de acordo com a necessidade do cliente. Dessa forma, a *Factory* pode fazer uso de, por exemplo, um *pool* de instâncias, ou determinar qual a melhor implementação de classe que se aplica a uma determinada requisição.

3.1.3 Eclipse IDE

O Eclipse é um *framework* para integrar diferentes tipos de aplicações. Uma de suas aplicações é a JDT (*Java Development Tools*), a qual já vem com o Eclipse.
Essas aplicações são oferecidas em forma de *plugins* e automaticamente reconhecidas e integradas pela plataforma.

Tendo seus próprios recursos para gerenciamento de mecanismo, que são geralmente arquivos no seu HD. Eles residem no seu *workspace*, uma pasta especial localizada no seu sistema de arquivos. As aplicações instaladas comunicam-se entre si, com isso, se uma aplicação altera um recurso qualquer, todas as outras aplicações instaladas serão notificadas sobre essa mudança, garantindo uma consistência e integração em todo o seu ambiente de desenvolvimento.

Um usuário sempre trabalha no *workbench*, a parte "visível" da plataforma (GUI). A perspectiva escolhida determina a aparência atual do *workbench*. A perspectiva é uma coleção conhecida como "*views* e editores" que contribuem com ações especiais para o menu e a *toolbar*.

A maioria das *views* mostra informações especiais sobre os recursos. Dependendo da *view* somente partes ou relacionamentos internos dos recursos poderão ser mostrados. Como no caso de arquivos do tipo XML.

Um editor trabalha diretamente sobre um recurso (classes Java como exemplo). O Eclipse segue um ciclo de carrega-altera-salva que somente se um editor salvar suas alterações, a plataforma será capaz de notificar as outras aplicações.
Views especiais podem ser conectadas diretamente a um editor, adicionando recursos complementares ao manuseamento dos recursos. Por exemplo, a *Outline View* da perspectiva Java é conectada diretamente ao Editor Java.

O que torna o Eclipse uma IDE especial, é a extrema flexibilidade na qual podem ser combinadas *views* e editores. Dessa forma o *workbench* pode ser arrumado de uma forma livre e que melhor adapte o desenvolvedor. As *views* e editores podem ser adicionados em uma perspectiva aberta (mesmo se eles foram definidos em um *plugin* totalmente diferente). Portanto é possível ter a total liberdade para criar o ambiente de desenvolvimento que melhor agrade ao desenvolvedor, de uma forma agradável e customizada.

3.1.4 *Framework* WebWork

Desenvolver sistemas para Web requer mais do que simplesmente um amontoado de páginas, também é importante se preocupar com a organização, padronização e facilidade de implementação. Isso permite o desenvolvimento de sistemas modulares e de fácil manutenção.

Atualmente existe uma necessidade muito grande em se utilizar programação em 3 camadas no desenvolvimento de sistemas para Web. O desenvolvimento com MVC (*Model – View - Controller*), separando o modelo de negócios da camada de visualização e do controlador, possibilitando assim que o desenvolvimento e a manutenção de sistemas tornem-se muito mais fáceis.

O WebWork é um projeto da OpenSynphony, o qual é um projeto de software livre fundado e patrocinado pela empresa australiana Atlassian. O objeto da OpenSynphony é criar componentes J2EE *open-source* que sejam simples de usar e de integrar com outros sistemas, facilitando assim o trabalho do desenvolvedor *Java*.

Esse *framework* é considerado um irmão casula do *framework* Struts (pelo fato de não repetir alguns “erros” do irmão mais velho), provendo uma série de ferramentas úteis e flexíveis, capazes de aliviar consideravelmente o trabalho muitas vezes mecânico e repetitivo do programador, integrando-se facilmente com outras ferramentas que possuem o mesmo propósito.

Sendo um *framework* muito inteligente, baseado no projeto XWork (*Inteceptors*) da OpenSynphony, muito bem aceito no mercado, possui um comunidade bem ativa no mundo inteiro e é considerado como uma grande alternativa ao *framework* Struts.

Arquitetura do *Framework* WebWork

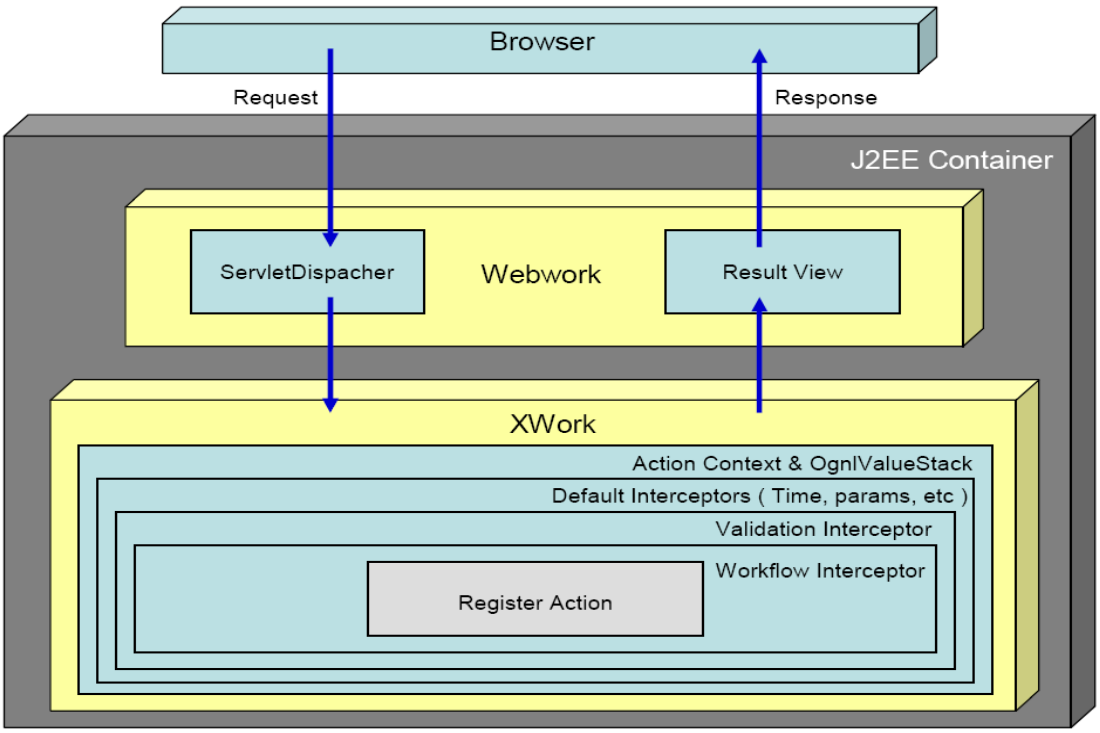

Figura 2: Arquitetura do *Framework* WebWork

Conceitos e Recursos

Ter códigos de acesso a dados, regras de negócio e códigos de apresentação misturados, pode gerar muitos problemas e dificultar muito a manutenção. Por causa da interdependência entre as partes do código, alterar um trecho pode provocar um efeito qualquer, podendo tornar o sistema não funcional.

Com códigos misturados a reutilização de código é muito difícil, ou impossível. Quando se deseja alterar a visualização do sistema, ou quando se deseja mudar de banco de dados, o sistema precisa ser quase inteiramente reescrito, e isso pode gerar outros problemas.

Para resolver estes problemas, existe o padrão de desenvolvimento chamado MVC que visa separar a camada de visualização das regras de negócio através de um controlador.

Quando um sistema é desenvolvido utilizando a arquitetura MVC, os códigos referentes às diferentes camadas ficam separados do resto do sistema, e sua manutenção se torna mais simples. A troca completa de uma camada pode ser feita sem gerar efeitos nas outras. Outra vantagem é que múltiplas camadas de visualização, podem compartilhar da mesma camada de negócio.

Definindo:

- ***Model*** – é onde estão as regras de negócio e o acesso aos dados no sistema.
- ***View*** – é onde os dados serão apresentados ou requisitados ao usuário. Os dados são acessados na base pela camada *Model* e repassados para a camada *View* para serem apresentados ou são requisitados pela camada *View* e repassados para a camada *Model* para serem persistidos.

- ***Controller*** – é o responsável por fazer a ligação entre o *View* e o *Model*. É o *Controller* que vai saber qual ação do *Model* deve ser chamada em cada requisição do *View*.

WebWork

WebWork é um *framework* MVC (*Model* - *View* - *Controller*), assim como outros frameworks como o Struts, JBanana, entre outros. É um *framework* propriamente dito, construído totalmente baseado no XWork com um conjunto de interceptadores, *results* e *dispatchers*, provendo assim um suporte a camada de visualização (*View*), possibilitanto o uso de praticamente qualquer tecnologia para camada *View*, como JSP, Velocity, FreeMaker e outros.

Recursos:

- **Ações** – o gerenciamento de ações ou de comando é característica mais básica do WebWork. Descrevendo uma ação no arquivo de configuração e criando uma classe *Java* esta ação é o suficiente para ver o WebWork funcionando.
- **Redirecionamento** – como resultado de uma ação temos um redirecionamento. Ao ser chamada uma ação, esta retorna para onde deseja redirecionar a aplicação na camada de visualização.
- **Validação de formulários** – o WebWork pode automatizar a validação de formulários com arquivos simples em XML, separando e facilitando sem a necessidade de programação em *Java*.
- **Componentes** – com a característica de Inversão de Controle do XWork, é possível ter objetos que ficam disponíveis para as ações em um determinado escopo.
- **Interceptadores (*Interceptors*)** - com os interceptadores, uma ação pode ser interceptada antes e depois de sua execução podendo ter seu fluxo desviado. Os interceptadores são modulares e múltiplos e podem ser utilizados para uma ação.
- **Internacionalização** – o WebWork possuir arquivos de recursos separados por ações e por línguas. Isso permite a fácil criação de recursos internacionalizados nos aplicativos Web.

XWork

É uma implementação genérica do padrão de projeto *Command Pattern* e é totalmente desacoplado do ambiente Web o que facilita o teste de suas ações.

Provê ao processamento das ações uma poderosa linguagem de expressões conhecida como OGNL (Linguagem de Navegação Objeto-Gráfica | *Object-Graph Navigation Language*), um container de IoC (Inversão de Controle), interceptadores, conversão de tipos e um *framework* de validação.

Fornecendo assim todo poder e flexibilidade ao WebWork, separando aquilo que é específico para Web do que é genérico.

ActionSuport (Ações) – Na distribuição do *framework* existe a classe *com.opensynphony.xwork.ActionSupport* que implementa diversas interfaces que são usadas pelo *framework* para prover a maioria dos serviços que desejamos como internacionalização, tratamento e validação de entrada e o comportamento padrão de uma *action*.

As ações são as classes que recebem a requisição do WebWork. Estas classes devem estar configuradas no arquivo *xwork.xml* e deve estar prontas para receber a requisição do WebWork.

Utilizando o WebWork, os JSP's terão apenas a tarefa de controlar a visualização do sistema, a camada *View*, e eles é quem irão fazer a requisição das ações ao WebWork. Os resultados são o destino do WebWork após executar uma ação. Ao ser configurada uma ação, devem ser especificados todos os possível retornos, definindo ainda o tipo de retorno. Abaixo segue alguns retornos do WebWork:

- *dispatcher*
  Encaminha o resultado para um determinado local, este é o tipo de retorno mais utilizado no WebWork.

```
<action name="associadoPesquisa"
class="com.mj.sisloc.action.admin.associado.AssociadoLocalizarAc
tion">
      <result name="success" type="dispatcher">
         /admin/associadoLista.jsp
      </result>
```

- *redirect*
  Encaminha o resultado para um determinado local. Diferente do dispacher, o redirect não envia as propriedades da ação para a página de resultado.

```
<action name="associadoCadastrar" class="com.mj.sisloc.
action.admin.associado.AssociadoCadastrarAction">
      <result name="success" type="redirect">
         associadoDetalhe.action?associadoid=${associado.id}
      </result>
```

- *velocity*
  Encaminha o resultado para um arquivo de modelo do velocity. O WebWork trabalha nativamente com o *Velocity* e sua configuração é bem simples.

```
<action name="associadoDetalhe" class="com.mj.sisloc.
action.admin.associado.AssociadoDetalheAction">
      <result name="success" type="velocity">
         /template/simple/associado-details.vm
      </result>
```

- *chain*
  Encaminha o resultado de uma ação para outra ação.

```
<action name="aluguelSelecionarAssociado" class="com.mj.sisloc.
action.admin.aluguel.AluguelSelecionarAssociadoAction">
      <result name="carrinhoMidia" type="chain">
         midiaCarrinhoLista
      </result>
```

Interceptors (Interceptadores) – A maioria dos recursos existentes no WebWork foi codificada como Interceptors que gerenciam o processamento passo a passo da requisição e resposta. Depois de registrados os *interceptors*, podem ser aplicados à ação conforme a necessidade de utilização de cada um. O *framework* tem a finalidade de configurarmos pilhas de *interceptors* e provê inclusive duas pilhas padrão com os serviços básicos que precisamos.

Configuração dos *interceptors* no arquivo *xwork.xml*:

```
<interceptors>
    <interceptor name="hibernate"
    class="com.mj.common.interceptor.HibernateInterceptor" />
    <interceptor name="security"
    class="com.mj.sisloc.interceptor.SecurityInterceptor" />

    <interceptor-stack name="defaultWebStack">
          <interceptor-ref name="timer"/>
```

```
            <interceptor-ref name="defaultStack"/>
            <interceptor-ref name="security"/>
        </interceptor-stack>
            .
            .
            .
```

Utilização dos *interceptors* configurados nas *actions*:

```
<action name="associadoCadastrar"
class="com.mj.sisloc.Action.admin.associado.AssociadoCadastrar
Action">
      <interceptor-ref name="validationWebStack"/>
      <interceptor-ref name="hibernateStack"/>
      <result name="success" type="redirect">
          associadoDetalhe.action?associadoid=${associado.id}
      </result>
      <result name="input" type="dispatcher">
          /admin/associadoForm.jsp
      </result>
      <interceptor-ref name="fileUploadStack"/>
</action>
```

Validação – O suporte a validação de entradas no WebWork pode ser feito através da implementação de validadores gerenciados pelo XWork ou da implementação do método *validate()* na *action*. O método validate não tem retorno nenhum (*void*) e assim como validadores não irá impedir a execução da *action*. Na validationWorkflowStack existe um interceptor chamado workflow que é responsável por verificar se existe erro nos mapas de erro e se existir interrompe a execução da *action* retornando para o INPUT.

Pode-se utilizar os dois meios em conjunto criando um suporte de validação extremamente eficiente. O arquivo *validators.xml* deve ser encontrado na pasta Web-INF\classes.

Para configurar validações para respectivas *action* esse XML de validação deve estar no mesmo local que a classe da *action* considerando a seguinte ordem de busca de validadores:

- Busca o arquivo *NomeDaClasseDaAction-validation.xml* e caso não encontre faz a mesma composição com o nome das super classes da *action* que encontre *Java*.lang.Object;
- Busca o arquivo com o *NomeDaClasseDaAction-NomeDaAction-validation.xml* onde NomeDaAction é o *alias* da *action* e irá também fazer a mesma composição com as super classes.

Internacionalização – O WebWork oferece ótimo suporte à internacionalização do sistema, isso permite que você crie arquivos, um para cada idioma que desejar suportar, contendo todas as mensagens que serão utilizadas pelo sistema. Desde as mensagens de erro na validação, até os textos de conteúdo do sistema podem ser automaticamente baseado no idioma do usuário.

Baseia-se no *Locale* informando na requisição http, isso normalmente significa que o *Locale* é o do sistema operacional, embora possa ser configurado pelo usuário.

Os arquivos de internacionalização utilizados pelo *framework* a cada processamento seguem o seguinte padrão:

- Busca a chave em um arquivo *.properties* com o nome da classe *action* (*ActionClass.properties*) no mesmo endereço da classe (mesmo pacote - *package*), se não encontrá-la procura o mesmo padrão para as super classes da *action* até que encontre *java.lang.Object*;

- Caso não tenha sucesso faz o mesmo processo anterior concatenando ao nome da classe o *Locale* com um *underscore*. (num caso de um *Locale* de português Brasil, procura por *ActionClass_pt_BR.properties*);
- Se ainda assim não encontrar buscará nos arquivos de recursos registrados como padrão;

Processamento feito pelo WebWork ao iniciar, registra os seguintes arquivos de recursos para recuperar as mensagens:

- *com/opensynphony/WebWork/WebWork-messages*
- *com/opensynphony/xwork/xwork-messages*

Componentes – O XWork implementa IoC (Inversão de controle) através da injeção de interface, ou seja, para dizer que determinada *action* irá utilizar um componente basta que esta *action* implemente uma interface específica. A idéia é prover automaticamente para determinadas *actions* instâncias de objetos necessárias para realizar a ação.

A configuração dos componentes é feita no arquivo *components.xml* que deve estar no seguinte lugar: WEB-INF\classes.

O tempo de vida dos componentes é definido pelo escopo que pode ser por requisição (*request*), sessão (*session*) ou aplicação (*application*), a definição desses escopos é análoga ao ciclo dentro da aplicação em um servidor Web.

IoC (Inversão de Controle) – Inversão de Controle, também conhecida como *Dependency Injection* (Injeção de Dependência), é uma forma de administrar dependências entre objetos. Um container que, como o WebWork, provê injeção de dependência e automaticamente atribui a determinados objetos instâncias de classes que a eles são necessárias.

View – O WebWork tem uma vasta biblioteca de *tags* que podemos utilizar em nossa camada de visualização. Basicamente essas *tags* trabalham utilizando todos os recursos do *framework* como internacionalização, tratamento de erros e entradas. Os *templates* das *tags* podem ser alterados ou reescritos, os que são utilizados por padrão estão no JAR do WebWork na package template.

Velocity – Velocity é um processador de modelos que, entre outras coisas, nos permite substituir as páginas JSP por *templates* contendo referências a objetos *Java*. A idéia por trás do Velocity é separar as atribuições dos Web designers (manter as páginas HTML) das tarefas do desenvolvedor (escrever código *Java*).

O Velocity é parte do projeto Jakarta da Apache Foundation. Existem duas maneiras de utilizar o Velocity no WebWork. A primeira é usando o tipo de resultado *velocity* em suas ações, a segunda é registrando o *WebWorkVelocityServlet* para atender as requisições HTTP diretas.

A primeira abordagem possibilita ao desenvolvedor um maior controle das páginas visitadas, pois elas só poderão ser abertas por meio de uma ação, porém muitas vezes é útil permitir o acesso direto aos *templates* a qual seria a segunda maneira de utilizar.

Se escolher utilizar o Velocity somente por meio de ações recomenda-se colocar os *templates* dentro do diretório WEB-INF da aplicação Web, para que não seja possível acessá-los diretamente.

Caso opte usar na segunda maneira, registrando o *servlet WebWorkVelocityServlet* para atender as solicitações diretamente, deve-se incluir o mapeamento do servlet no arquivo *web.xml*.

Independente da forma de uso, os recursos disponíveis são os mesmos quando criamos *templates*, o principal recurso que o uso do Velocity com o WebWork

proporciona é o fácil acesso às propriedades da ação que acaba de executar. As propriedades da ação podem ser obtidas pelo uso de uma simples referência a uma variável. Assim a referência $nome seria substituída pelo valor retornado do método *getNome()* da ação recém executada.

Outros objetos úteis também estão disponíveis automaticamente, os objetos *HttpServletRequest* e *HttpServletResponse* estão registrados sob os nomes *$req* e *$res*, respectivamente. A *Value Stack* é uma instância de OGNLTool estão disponíveis nas referências *$stack* e *$ognl* (*Value Stack* é uma pilha que armazena os valores disponíveis para as páginas JSP ou Velocity; OGNL é uma linguagem de navegação em grafos usada para recuperar informações da *Value Stack*).

As *tags* JSP não funcionam em *templates* Velocity, no entanto, o WebWork provê as velomacros *#tag* e *#bodytag*, que permitem o uso de *tags* do WebWork.

WebWork x Struts

É inevitável a comparação do WebWork com o Struts. O propósito dos dois é o mesmo, ser um controlador na arquitetura MVC, mas eles trabalham de forma um pouco diferente. Algumas diferenças são listadas a seguir:

- *Abstração da API do Servlet* – Uma ação do Struts é totalmente dependente dos servlets porque a ação que deve ser estendida é uma extensão de *HTTPServlet*. Na assinatura do seu método de entrada já é necessário ter os atributos *ServletRequest* e *ServletResponse* como parâmetros. No modelo de ações do XWork, as ações não dependem da API do *Servlet* e podem funcionar sem estar na Web. É possível ter acesso ao contexto do *Servlet* através da classe *ActionContext* mas isto não é necessário.
- *Facilidade de implementação* – O WebWork é bem mais simples de se trabalhar do que o Struts. Criar uma ação no WebWork é mais produtivo, não é necessário criar os *FormBeans* do Struts, e o conceito *Model Driven* do WebWork faz com que os dados do formulário sejam colocados automaticamente em uma classe de negócio.
- *Testabilidade* – Testar uma ação do Struts é um processo trabalhoso, tendo em vista que os objetos de *Request* e *Response* são necessários. Já uma ação do WebWork pode ser facilmente testada por ser independente da API do *Servlet*.
- *Interceptadores modulares* – Os interceptadores do WebWork são modulares, e podem ser implementados sem ligação direta com uma determinada ação. No Struts os processamentos feitos antes e depois das ações estão ligados à hierarquia das classes de ação, o que dificulta a modularização e a implementação de múltiplos interceptadores.

3.1.5 Hibernate

O Hibernate é uma ferramenta de mapeamento objeto/relacional para Java. Ela transforma os dados tabulares de um banco de dados em um grafo de objetos definido pelo desenvolvedor.

O objetivo do Hibernate é facilitar a construção de aplicações Java dependentes de bases de dados relacionais e facilitar o desenvolvimento das consultas e atualizações dos dados.

Além do mecanismo de mapeamento objeto/relacional, o Hibernate também pode trabalhar com um sistema de *cache* das informações do banco de dados, aumentando ainda mais a performance das aplicações. O esquema de *cache* do

Hibernate é complexo e totalmente extensível, existindo diversas implementações possíveis, cada uma com suas próprias características.

O acesso ao banco, independentemente de como a aplicação é construída, é sempre realizado através de *drivers* que suportam o padrão JDBC.

O Hibernate é responsável apenas pelo mapeamento das tabelas do modelo relacional para classes da linguagem Java. As questões relacionadas ao gerenciamento de transações e a tecnologia de acesso à base de dados são de responsabilidade de outros elementos da infra-estrutura da aplicação.

Usando o Hibernate, o desenvolvedor se livra de escrever muito do código de acesso a banco de dados e de SQL que ele escreveria não usando a ferramenta, acelerando a velocidade do seu desenvolvimento de uma forma fantástica. O uso de ferramentas de mapeamento objeto relacional, como o Hibernate, diminui a complexidade resultante da convivência de modelos diferentes: o modelo orientado a objetos (da linguagem Java) e o relacional (da maioria dos SGBD's).

Apesar da API do Hibernate possuir operações para o controle transacional, por exemplo, ele simplesmente delega estas tarefas para infra-estrutura no qual foi instalado. No caso de aplicações construídas para serem executadas em servidores de aplicação, o gerenciamento das transações é realizado, normalmente, segundo o padrão JTA. Nas aplicações *standalone* (que são independentes de um servidor de aplicação), o Hibernate delega o tratamento transacional ao *driver* JDBC.

O Hibernate trabalha através da interação das classes *Java* (**.java*) com arquivos de mapeamento Hibernate (**.hbm.xml*). Geralmente para cada classe VO existe um xml *.hbm.xml* relacionado. Este arquivo de formato XML contém informações sobre a tabela e campos do banco de dados a serem vinculados à classe e suas propriedades. Ex.: *Diretor.hbm.xml*

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd" >

<hibernate-mapping package="com.mj.sisloc.model">
      <class name="Diretor" table="diretor">
            <id
                  column="DIRETOR_ID"
                  name="id"
                  type="long"
                  unsaved-value="null"
            >
                  <generator class="native" />
            </id>
            <property
                  column="NOME"
                  length="255"
                  name="nome"
                  not-null="true"
                  type="string"
             />
            <property
                  column="URL"
                  length="255"
                  name="url"
                  not-null="false"
                  type="string"
             />
            <set
                  table="filme_diretor"
                  inverse="true"
```

```
                lazy="true"
                name="filmes"
            >
                <key column="DIRETOR_ID"/>
                <many-to-many class="Filme" column="FILME_ID"/>
            </set>
        </class>
</hibernate-mapping>
```

O mapeamento que é realizado através desse arquivo é de vital importância para o funcionamento do sistema, pois todos os seus objetos irão ser carregados a partir do mapeamento realizado pelo Hibernate com base nesses arquivos *.*hbm.xml*. As propriedades do arquivo *Diretor.hbm.xml* é definida a seguir.

Tag hibernate-mapping: *tag* pai que inicia o mapeamento objeto-relacional da classe com a tabela do banco de dados.

- *package*: propriedade da *tag hibernate-mapping* e especifica o nome do pacote onde a classe está localizada.

Tag class: *tag* filha da *tag hibernate-mapping* e apresenta as configurações de mapeamento da classe.

- *name*: nome da classe que será mapeada.
- *table*: nome da tabela a que essa classe é referenciada.

Tag id: *tag* filha da *tag class* e mapeia o atributo que será referenciado como id da tabela.

- *column*: nome da coluna da tabela que contém o id.
- *name*: nome do atributo da classe que referenciará a coluna especificada.
- *type*: tipo do objeto que será instanciado a coluna mapeada.
- *unsaved-value*: especifica que esse valor não pode ser salvo ou alterado.

Tag generator: *tag* filha à *tag id* e especifica quem gerará os valores do atributo id da tabela.

- *class*: especifica a classe que será a responsável por gerenciar o incremento do id. O valor *native* especifica que quem controlará a ação será o próprio banco de dados.

Tag property: *tag* filha da *tag class* e especifica uma propriedade da tabela que será mapeada. A maioria das propriedades dessa *tag* são idênticas às propriedades da *tag id* já especificada.

- *length*: especifica o comprimento máximo do atributo mapeado.
- *not-null*: especifica se será permitido valores não nulos na tabela.

Tag set: *tag* filha da *tag class* e é responsável por realizar os mapeamentos 1-n, n-1 e n-n. Um *set* ou um conjunto representa uma coleção de objetos não repetidos que podem ou não estar ordenados.

- *table*: nome da tabela que o set fará o mapeamento.
- *name*: nome do atributo da classe que referenciará a coleção de elementos mapeados.
- *inverse:* é utilizado para que o Hibernate saiba como tratar a associação entre as duas tabelas. Quando um lado da associação define o atributo *inverse* como **true**, indica que a ligação do relacionamento entre a associação será de responsabilidade do "outro lado" da associação.

Tag key: tag filha da *tag set* e mapeia a coluna da tabela relacionada que possui a chave estrangeira para a classe Diretor.

- *column*: nome da coluna da tabela que contém a identificação da classe Diretor.

A próxima *tag* a ser inserida dentro da *tag set* especificará o tipo de mapeamento (1-n - <*one-to-many*> ou n-n - <*many-to-many*>).

O arquivo *hibernate.cfg.xml* é utilizado para a configuração do Hibernate. Nele são incluídas informações como: o dialeto "falado" pelo banco de dados, a classe Java do *driver* JDBC, a identificação do usuário e senha para acesso ao banco de dados, e uma lista de arquivos de relacionamentos (**.hbm.xml)*. A seguir mostramos o arquivo *hibernate.cfg.xml* e as definições de suas configurações.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration
    PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-
2.0.dtd">

<hibernate-configuration>
      <session-factory>

            <property
name="hibernate.connection.driver_class">org.gjt.mm.mysql.Driver</prop
erty>
            <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/sisloc</pr
operty>
            <property
name="hibernate.connection.username">root</property>
            <property
name="hibernate.connection.password">s1sL0c</property>
            <property
name="hibernate.connection.pool_size">2</property>

            <! - configuração do DBCP  -->
            <property name="hibernate.dbcp.ps.maxActive">4</property>
            <property name="hibernate.dbcp.ps.maxIdle">4</property>
            <property name="hibernate.dbcp.ps.maxWait">60000</property>
            <property
name="hibernate.dbcp.ps.whenExhaustedAction">2</property>

            <property name="hibernate.dbcp.validationQuery">SELECT
1+1</property>


            <!-- dialeto para MySQL -->
            <property
name="dialect">net.sf.hibernate.dialect.MySQLDialect</property>
            <property name="hibernate.show_sql">false</property>

        <mapping resource="com/mj/sisloc/model/Ator.hbm.xml" />
        <mapping
resource="com/mj/sisloc/model/ArquivoCobranca.hbm.xml" />
        <mapping
resource="com/mj/sisloc/model/HistoricoCobranca.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/Pais.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/Diretor.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/LogEvento.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/Permissao.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/Sexo.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/Genero.hbm.xml" />
        <mapping
resource="com/mj/sisloc/model/ParametroGlobal.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/Lingua.hbm.xml" />
```

```
        <mapping resource="com/mj/sisloc/model/Midia.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/MidiaStatus.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/MidiaTipo.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/Associado.hbm.xml" />
        <mapping
resource="com/mj/sisloc/model/AssociadoEventoCobranca.hbm.xml" />
        <mapping
resource="com/mj/sisloc/model/AssociadoContato.hbm.xml" />
        <mapping
resource="com/mj/sisloc/model/AssociadoAluguelMidia.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/PlanoAssociado.hbm.xml"
/>
        <mapping
resource="com/mj/sisloc/model/OperadoraCelular.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/Loja.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/Filme.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/FilmeGenero.hbm.xml" />
        <mapping resource="com/mj/sisloc/model/FilmeConceito.hbm.xml"
/>
        <mapping resource="com/mj/sisloc/model/Usuario.hbm.xml" />
        <mapping
resource="com/mj/sisloc/model/UsuarioPermissao.hbm.xml" />
        <mapping
resource="com/mj/sisloc/model/ItemListaDesejo.hbm.xml" />
      </session-factory>
</hibernate-configuration>
```

Na documentação do Hibernate pode-se verificar todas as opções de propriedades que podem ser utilizadas e seus respectivos resultados. Para este projeto, as seguintes propriedades foram utilizadas:

- *hibernate.dialect*: é a implementação do dialeto SQL específico do banco de dados a ser utilizado
- *hibernate.connection.driver_class*: é o nome da classe do *driver* JDBC do banco de dados que está sendo utilizado
- *hibernate.connection.url*: é a URL de conexão específica do banco que está sendo utilizado
- *hibernate.connection.username*: é o nome de usuário com o qual o Hibernate deve se conectar ao banco
- *hibernate.connection.password*: é a senha do usuário com o qual o Hibernate deve se conectar ao banco.

Essa segunda parte do arquivo são as configurações do *pool* de conexões escolhido para a nossa aplicação. Neste projeto, o pool utilizado é o DBCP, que provê uma oportunidade de coordenar os esforços necessários para criar e manter um eficiente pool de conexão.

- *hibernate.dbcp.maxActive*: O máximo número de conexões ativas que podem ser alocadas ao mesmo tempo.
- *hibernate.dbcp.maxIdle*: O número máximo de conexões que podem permanecer no pool sem serem utilizadas.
- *hibernate.dbcp.maxWait*: O número máximo de milisegundos que um pool esperará (quando não existem conexões disponíveis) para uma conexão ser retornada antes de gerar uma exceção.
- *hibernate.dbcp.whenExhaustedAction*: O número máximo de tentativas para conseguir uma conexão.

- *hibernate.dbcp.validationQuery*: A *query* SQL que será usada para validar conexões deste *pool* antes de retornar a conexão para o chamador.

Na terceira parte, estão algumas opções para ajudar a debugar o comportamento do Hibernate, a propriedade *show_sql* faz com que todo o código SQL gerado seja escrito na saída default.

A última parte do arquivo é onde são indicados os arquivos de mapeamento que o Hibernate deve processar antes de começar a trabalhar. Se algum arquivo de mapeamento de qualquer classe não for indicado, essa classe não vai poder ser persistida pela *engine* do Hibernate. Outro detalhe importante, é que quando você usa mapeamentos com Herança, o mapeamento pai deve sempre vir antes do filho.

Nas diversas aplicações existentes sempre que for necessário propagar o estado de um objeto que está em memória para o banco de dados ou vice-versa, há a necessidade de que a aplicação interaja com uma camada de persistência. Isto é feito, invocando o gerenciador de persistência e as interfaces de consultas do Hibernate.

Quando interagindo com o mecanismo de persistência, é necessário para a aplicação ter conhecimento sobre os estados do ciclo de vida da persistência. Em aplicações orientadas a objetos, a persistência permite que um objeto continue a existir mesmo após a destruição do processo que o criou. Na verdade, o que continua a existir é seu estado, já que pode ser armazenado em disco e então, no futuro, ser recriado em um novo objeto.

Em uma aplicação não há somente objetos persistentes, pode haver também objetos transientes. Objetos transientes são aqueles que possuem um ciclo de vida limitado ao tempo de vida do processo que o instanciou. Em relação às classes persistentes, nem todas as suas instâncias possuem necessariamente um estado persistente. Elas também podem ter um estado transiente ou *detached*.

O Hibernate define estes três tipos de estados: persistentes, transientes e *detached*. Objetos com esses estados são definidos como a seguir:

- Objetos Transientes: são objetos que suas instâncias não estão nem estiveram associados a algum contexto persistente. Eles são instanciados, utilizados e após a sua destruição não podem ser reconstruídos automaticamente;
- Objetos Persistentes: são objetos que suas instâncias estão associadas a um contexto persistente, ou seja, tem uma identidade de banco de dados.
- Objetos *detached*: são objetos que tiveram suas instâncias associadas a um contexto persistente, mas que por algum motivo deixaram de ser associadas, por exemplo, por fechamento de sessão, finalização de sessão. São objetos em um estado intermediário, nem são transientes nem persistentes.

No Hibernate, assim como no JDBC, existem os conceitos de sessão e transação. Uma sessão é uma conexão aberta com o banco de dados, onde nós podemos executar *queries*, inserir, atualizar e deletar objetos, já a transação é a demarcação das ações, uma transação faz o controle do que acontece e pode fazer um *rollback*, assim como uma transação do JDBC, se forem encontrados problemas.

Nesse projeto foi criada uma classe que encapsula os conceitos de sessão e transação, provendo de forma rápida pra o usuário a sua utilização pelas classes de acessos de dados. Abaixo exibimos a classe *HibernateUtil* que contém essas implementações.

```
public class HibernateUtil {
   private static final Logger log = Logger.getLogger( HibernateUtil.class );

   private static SessionFactory sessionFactory;
   private static final ThreadLocal threadSession = new ThreadLocal();
```

```
private static final ThreadLocal threadTransaction = new ThreadLocal();

static {
    try {
        Configuration cfg = new Configuration();
        sessionFactory = cfg.configure().buildSessionFactory();
    } catch ( Throwable e ) {
        log.error( e );
        throw new ExceptionInInitializerError( e );
    }
}

public static Session getSession() {
    Session session = (Session)threadSession.get();
    try {
        if( session == null ) {
            session = sessionFactory.openSession();
            threadSession.set( session );
        }
    } catch ( HibernateException e ) {
        log.error( e );
    }
    return session;
}

public static void closeSession() {
    try {
        Session session = (Session)threadSession.get();
        threadSession.set( null );
        if( session != null && session.isOpen() ) {
            session.close();
        }
    } catch ( HibernateException e ) {
        log.error( e );
    }
}

public static void beginTransaction() {
    Transaction tx = (Transaction)threadTransaction.get();
    try {
        if( tx == null ) {
            tx = getSession().beginTransaction();
            threadTransaction.set( tx );
        }
    } catch ( HibernateException e ) {
        log.error( e );
    }
}

public static boolean transactionStarted() {
    if (threadTransaction.get() != null) {
```

```
            return true;
        } else {
            return false;
        }
    }

    public static void commitTransaction() {
        Transaction tx = (Transaction)threadTransaction.get();
        try {
            if( tx != null && !tx.wasCommitted() && !tx.wasRolledBack() ) {
                tx.commit();
            }
            threadTransaction.set( null );
        } catch ( HibernateException e ) {
            rollbackTransaction();
        }
    }

    public static void rollbackTransaction() {
        Transaction tx = (Transaction)threadTransaction.get();
        try {
            threadTransaction.set( null );
            if( tx != null && !tx.wasCommitted() && !tx.wasRolledBack() ) {
                tx.rollback();
            }
        } catch ( HibernateException e ) {
            log.error( e );
        } finally {
            closeSession();
        }
    }
}
```

O bloco estático (as chaves marcadas como *static*) instancia um objeto de configuração do Hibernate (*org.hibernate.cfg.Configuration*), chama o método configure() (que lê o nosso arquivo *hibernate.cfg.xml*) e depois que ele está configurado, criamos uma *SessionFactory*, que é a classe que vai ficar responsável por abrir as sessões de trabalho do Hibernate.

Como forma de otimizar os recursos da sessão, que serão amplamente utilizados durante a utilização do sistema, os métodos da classe HibernateUtil procuram nas *threads* locais o objeto requisitado. Essas *threads* estão referenciadas pelos atributos *threadSession* e *threadTransaction* e têm a finalidade de guardar um sessão ou transação que está sendo utilizada nesse momento e caso seja requisitado uma sessão ou transação, a mesma que está sendo utilizada é repassada, como forma de manter o pool de conexão e a sessão em um número limitado de instâncias.

Existem três meios de se fazer buscas com o Hibernate: usando a sua linguagem própria de buscas, a *Hibernate Query Language* (HQL), usando a sua *Criteria Query API* (para montar buscas programaticamente) e usando SQL puro. A maioria das suas necessidades deve ser suprida com as duas primeiras alternativas. Para o resto, sempre se pode usar SQL pra resolver.

A HQL é uma extensão da SQL com alguns adendos de orientação a objetos, nela você não vai referenciar tabelas, vai referenciar os seus objetos do modelo que foram mapeados para as tabelas do banco de dados.

A *Criteria Query API* é um conjunto de classes para a montagem de queries em código Java, você define todas as propriedades da pesquisa chamando os métodos e avaliações das classes relacionadas.

Outro complemento importante do Hibernate é o suporte a paginação de resultados. Para paginar os resultados, nós chamamos os métodos setFirstResult(int first) e *setMaxResults(int max)*. O primeiro método indica de qual posição os objetos devem começar a ser carregados, o segundo indica o máximo de objetos que devem ser carregados. Estes métodos estão presentes tanto na interface *Query* quanto na interface "*Criteria*". Uma propriedade específica da HQL, que foi herdada do JDBC, é o uso de parâmetros nas queries.

Assim como no JDBC, os parâmetros podem ser numerados, mas também podem ser nomeados, o que simplifica ainda mais o uso e a manutenção das queries, porque a troca de posição não vai afetar o código que as usa.

3.1.6 DisplayTag

A DisplayTag Library é uma suíte *open-source* de *custom tags* que fornecem uma apresentação Web de alto nível para ser usada em uma aplicação MVC. Sua biblioteca possibilita um aumento significante de funcionalidade mesmo sendo simples o seu uso.

Com este componente é possível mostrar tabelas, isto é, listar as informações de objetos de uma aplicação. Seus diferenciais são as possibilidades na apresentação dessas tabelas. É possível criar tabelas com diferenciação nas cores das linhas, ordenação nas colunas, paginação dos dados, agrupamento de informações, exportação dos dados, links e decoração customizável.

O uso mais simples que existe da DisplayTag é criar uma tabela com uma lista de objetos. Mesmo assim já podemos ver ganhos de produtividade no momento que podemos criar uma tabela usando apenas uma *tag* e indicando a coleção que contém os objetos que serão mostrados.

A DisplayTag provê uma série de funcionalidades. Algumas delas são:

- Exportação para Excel, CSV e XML.
- Ordenação.
- Paginação.
- Utilização de CSS.
- Decorators.

Essa biblioteca permite o uso de *tags*, parecidas com *tags* html para exibir os dados da lista em uma tabela personalizada. As duas principais *tags* são:

<*display:table*> - Gera uma tabela HTML.

name – Faz referência ao objeto que irá popular a tabela.

<*display:column*> - Exibe uma propriedade de um objeto em uma linha dentro da tabela.

property – Armazena o valor de uma propriedade de um Objeto.

title – Exibe um título para a coluna.

Ordenação

Uma das funcionalidades mais interessantes da DisplayTag é a possibilidade de ordenar por colunas. Para ordenar por uma coluna basta adicionar os seguintes atributos:

<display:table>

sort – Indica o tipo de ordenação. Aceita os seguintes parâmetros:

list – Toda a lista é ordenada.

page – apenas a página corrente é ordenada.

<display:column>

sortable – Indica se a coluna aceita ordenação. Aceita os seguintes parâmetros:

true – A coluna aceita ordenação.

false – A coluna não aceita ordenação.

Caso o usuário não selecione uma coluna, podemos deixar uma coluna ordenada por default. Para isso, devemos indicar a coluna a ser ordenada na *tag* <display:table>

<display:table>

defaultsort – Indica a coluna que receberá ordenação por default. Os índices das colunas começam em zero.

Ao adicionar suporte a ordenação, a listagem deve ficar assim:

```
<display:table name="usuario" sort="list" defaultsort="1">
    <display:column property="id" title="ID" sortable="true"/>
    <display:column property="nome" title="Nome" sortable="true"/>
</display:table>
```

Paginação

A DisplayTag oferece um ótimo suporte a paginação de dados. Para que sua lista tenha acesso a essa funcionalidade, adicione os seguintes atributos:

<display:table>

pagesize – Indica o número de linhas por página. Você deve inserir o número de linhas que você deseja exibir por página.

requestURI – Indica a URL em que a paginação será realizada.

Exportação

Outra funcionalidade que a DisplayTag fornece é a de exportação de dados. Os dados podem ser exportados para os seguintes formatos:

- CVS
- Excel
- XML
- PDF

Para que a listagem tenha acesso ao suporte de exportação, você deve adicionar os seguintes atributos:

<display:table>

export – Indica se a listagem poderá exportar os dados.

true - A listagem poderá exportar.

false – A listagem não poderá exportar.

4 Detalhamento da Solução

O projeto foi implementado através da linguagem de programação JAVA, totalmente orientado a objetos e seguindo a arquitetura MVC, provida pelo *framework* WebWork. Ele foi dividido em dois módulos, sendo estes módulos posteriormente detalhados no apêndice Especificação de Requisitos de Software.

- Módulo de gerenciamento (interface Web): este módulo é responsável pela administração e pelo gerenciamento do sistema. Através desse módulo, os usuários da locadora têm acesso as principais funcionalidades do sistema que garantem o funcionamento da locadora.
- Módulo de divulgação (interface Web): este módulo é responsável pela divulgação da locadora para seu público alvo e apresentará funcionalidades, tais como as informações dos filmes que existem na locadora e interação virtual do associado com a locadora.

Ambos os módulos foram desenvolvidos utilizando os conceitos de sistemas baseados na plataforma J2EE, tais como *action*, JSP e banco de dados e os *design patterns* que garantem a qualidade e a uniformidade da programação.

A arquitetura do sistema é exibida abaixo e através dela podemos verificar como são organizadas e implementadas as classes do sistema e como elas se comunicam com as diversas ferramentas que foram utilizadas tais como Hibernate, WebWork e MySQL.

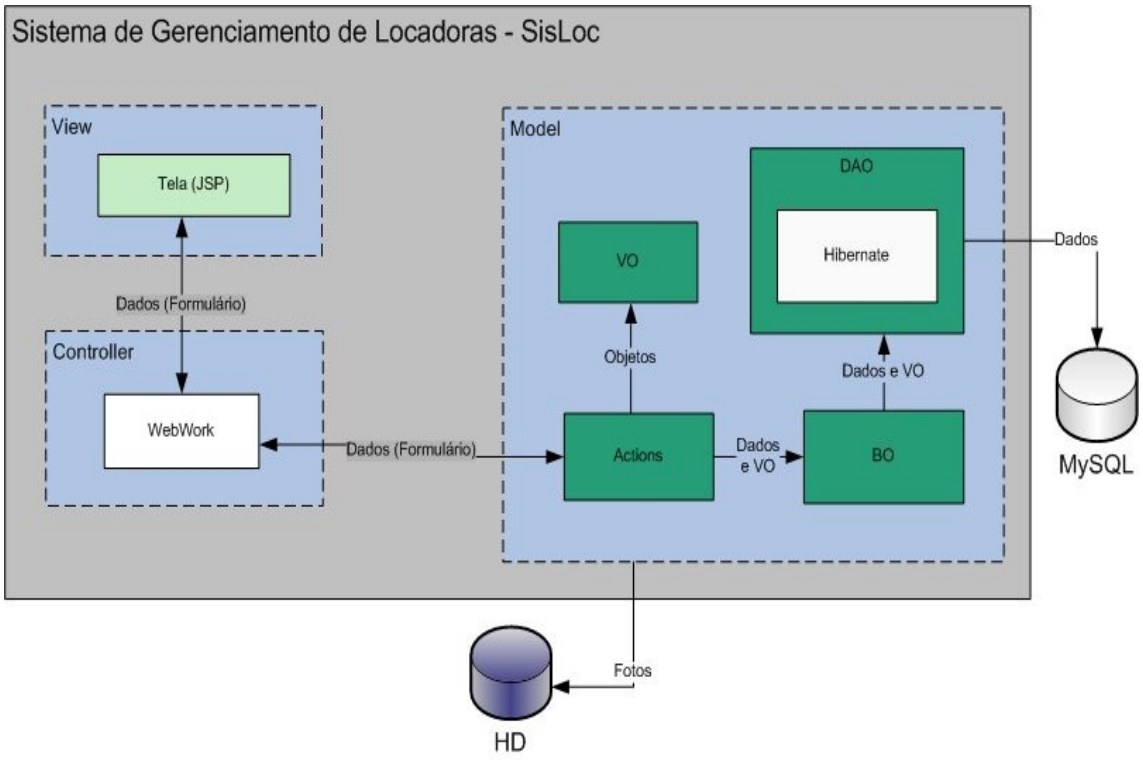

Figura 3: Arquitetura Detalhada do Sistema

Pela análise da arquitetura acima, podemos verificar como são organizados cada parte do sistema como as *actions,* as classes BO, VO e DAO e as suas interações com os

frameworks Hibernate e WebWork. Na arquitetura também verificamos onde cada parte se organiza na padronização MVC.

A primeira camada do sistema, a *View*, garante a visualização o acesso do usuário no sistema e é composta pelas páginas em JSP. Através dessas páginas os usuários acessarão as funcionalidades do sistema.

Ao submeterem as informações passadas através dos formulários nas páginas JSP, os usuários invocam a camada de Controle da aplicação, que é gerenciada pelo *framework* WebWork. Essa camada faz o gerenciamento dos fluxos das aplicações, conhecendo todos os caminhos mapeados. Assim o *framework* WebWork fica responsável por encaminhar as informações submetidas pelos usuários para as classes *actions* responsáveis pelo fluxo em questão.

As classes *actions* são encontradas na camada *Model* e são classes que tartarão os dados submetidos pelo usuário de forma adequada. As *actions,* com os dados submetidos pelos usuários, instanciam objetos VO's a partir desses dados. Esses objetos são então enviados juntamente com outros dados submetidos pelos usuários para as classes BO. Essas classes como explicado anteriormente, abstraem os métodos de acesso ao banco de dados escritos nas classes DAO e também controlam as transações da aplicação.

As classes DAO são responsáveis por prover dados relativos ao banco de dados à classe BO e essa classe contém as *queries* que buscam, inserem e removem informações no banco de dados da aplicação. As classes DAO utilizam o *framework* Hibernate para garantir o acesso aos dados do banco.

Em alguns fluxos, como no caso de inserção de associados e de filmes é necessário salvar no disco os dados da foto que foram cadastrados para o associado ou filme.

Nas subseções a seguir, serão desenvolvidas explicações sobre as tecnologias e classes utilizadas, além de cada módulo implementado nas fases descritas acima ser detalhado.

4.1 Módulo de Gerenciamento

O módulo de gerenciamento é o módulo responsável pela garantia das operações básicas que uma locadora possui, além de possuir outras funcionalidades que visam a garantia do controle financeiro e das mídias alugadas pelos associados.

O módulo de gerenciamento utiliza as seguintes tecnologias associadas:

- Plataforma: Java. Utilização do modelo MVC como padrão de projeto
    - Vista: HTML, Java *Servlet* Pages (JSP's), Ajax e Javascript.
    - Controle: Mapeamento de *actions* através do uso do *framework* WebWork.
    - Modelo (Negócio): Hibernate e implementação utilizando *design patterns* conforme especificação J2EE (DAO, BO, VO, *Factory*, *Singleton*).
- Modelo de Banco de Dados: DBDesigner 4 [18]
- Banco de Dados: MySQL 4.
- Application Server: Apache Tomcat 5.0

As funcionalidades que o módulo de gerenciamento apresenta são:

- Inserir/Editar Loja;
- Inserir, Listar Usuários;

- Editar, Remover Usuários (apenas super-usuário);
- Inserir, Editar, Remover, Listar Diretores;
- Inserir, Editar, Remover, Listar Atores;
- Inserir, Editar, Remover, Pesquisar Filmes;
- Inserir, Editar, Remover Mídias;
- Inserir, Editar, Remover, Listar Planos Mensais. Os planos mensais são responsáveis pelo funcionamento do sistema conforme o modelo de negócio proposto. Para isso no cadastro de um associado no sistema, sempre é necessário especificar em qual plano esse associado irá se associar. O plano mensal garante ao associado a locação de uma certa quantidade de mídias máxima em sua posse. Além das informações de quantidade máxima de mídias que o associado poderá possuir em casa, o plano mensal também possui informações sobre preço mensal do plano, quantidade máxima de dependentes e número máximo de filmes na lista de desejo;
- Inserir, Editar, Ativar/Desativar, Pesquisar Associados;
- Alugar mídias para associados na loja. Esse tipo de locação será utilizado quando o associado está na loja e deseja alugar as mídias que escolheu;
- Alugar mídias para associados por telefone. A locação de mídias por telefone se baseia na forma de locação em que o associado entra em contato com a loja por telefone e o funcionário que está na loja executa a locação das mídias pedidas na conta do associado;
- Incluir, Remover filmes na lista de desejo. A lista de desejo é uma funcionalidade bastante interessante. A lista de desejo tem como característica permitir ao associado escolher os filmes que deseja ver e que por ventura todas as mídias já estejam alugadas na loja e colocar esses filmes nessa lista. Assim conforme as mídias vão sendo retornadas à loja os funcionários da loja verificam pelo sistema quais são os associados que preencheram a lista de desejo e caso tenha algum informa ao associado que já existem mídias disponíveis daquele filme;
- Gerar relatórios (mídias em locação, atividade de locação, histórico de locações, dinâmicos). O relatório de mídias em locação exibe informações das mídias que foram alugadas e ainda não foram devolvidas. O relatório de atividade de locações exibe informações de usuários que possuem mídias alugadas a partir de tempo estipulado pelo usuário. O relatório de histórico de locações exibe informações sobre o histórico de locações da loja e o relatório dinâmico permite que o próprio usuário produza os seus relatórios através da utilização de *queries* que irão acessar o banco de dados e retornar o resultado a ser exibido. Esse relatório poderá então ser exportado para XML, XLS (Excel) ou PDF, permitindo ao gerente da loja possuir consigo informações úteis que desejar;
- Gerar Cobrança (cobrança mensal). Como o sistema se baseia em um modelo de negócio em que existe um compromisso entre o associado e a locadora e que este associado deve pagar uma mensalidade referente ao seu plano. Devido a esse compromisso, foi implementada a funcionalidade que visa garantir esse controle dos pagamentos dos associados, assim a funcionalidade de cobrança mensal, garante a geração da cobrança para todos os associados.
- Sair do Sistema.

Através dessas funcionalidades é possível gerenciar e obter outras informações do funcionamento da loja, que permitem controlar o fluxo de mídias alugadas e também o controle financeiro.

Para utilização do sistema é necessário efetuar o *login*. Por ser um sistema de gerenciamento foram implementadas funcionalidades que visam garantir a segurança dos dados. Para isso foram desenvolvidas três funcionalidades importantes nesse aspecto, que são: *Login*, gravação da senha encriptada no banco de dados e controle de sessão dos usuários.

Abaixo vemos parte do código responsável por essa segurança, como a implementação da encriptação de uma String que utiliza a API de segurança "*MessageDigest*" da Sun.

```
public static String crypt(String str)  {
        try {
                if (str == null || str.length() == 0) {
                        throw new IllegalArgumentException(
                        "String nao pode ser nula ou vazia");
                }
                String seed = "jsfjsd993753475laldfjasdj__$&((#Hshfshdfsalalallll";
                str = seed + str;
                StringBuffer hexString = new StringBuffer();
                MessageDigest md;
                md = MessageDigest.getInstance("MD5");
                md.update(str.getBytes());
                byte[] hash = md.digest();

                for (int i = 0; i < hash.length; i++) {
                        if ((0xff & hash[i]) < 0x10) {
                        hexString.append("0" + Integer.toHexString((0xFF & hash[i])));
                        }
                        else {
                                hexString.append(Integer.toHexString(0xFF & hash[i]));
                        }
                }
                return hexString.toString().toUpperCase();
        } catch (NoSuchAlgorithmException e) {
                return null;
        }
}
```

Após efetuar o *login* com sucesso, o usuário está apto a visualizar as funcionalidades do sistema. A implementação do *login* de acesso ao sistema foi desenvolvida verificando primeiramente a existência do usuário que possui o *username* informado e depois verificado se a senha que o usuário informou é igual a senha cadastrada, mas como a senha que está no banco é a senha encriptada, a senha que foi informada é então encriptada e assim verificada com a senha que está no banco.

Depois do sucesso da verificação, o objeto relativo ao usuário é setado na sessão da aplicação. A sessão da aplicação é um objeto provido pela biblioteca do *framework* WebWork, que é responsável pelo controle da aplicação, e que é instanciado uma única vez por *login* do usuário. Assim, sempre que é necessário guardar informações que posteriormente serão utilizadas no sistema, a sessão garante essa funcionalidade.

Como forma a restringir o acesso de usuários a certas funcionalidades, foi criada no banco de dados a tabela de permissões que contém informações sobre o acesso aos

módulos do sistema. Assim, cada usuário do sistema tem acesso permitido aos módulos que lhe foram relacionados no momento do seu cadastro ou da edição dos seus dados. Com essa funcionalidade implementada, o sistema somente exibe no menu de acesso das funcionalidades, os módulos que foram cadastrados para o usuário, o que garante por exemplo o acesso do módulo responsável pelos dados da loja somente ao gerente da loja e não aos atendentes.

Após o usuário entrar no sistema e possuindo este usuário acesso aos módulos do sistema de gerenciamento, pode-se começar a utilizar as funcionalidades que o sistema provê. Uma das primeiras tarefas a serem executadas será o cadastramento de outros usuários do sistema, que serão os funcionários da loja. Cadastrando os usuários, estes poderão realizar o *login* e acessar os módulos que foram permitidos no momento do seu cadastro.

A próxima funcionalidade a ser utilizada é o cadastramento de filmes, pois serão estes o principal elemento do modelo de negócio, pois trata-se de um sistema para o gerenciamento de uma locadora. Para se cadastrar um filme são requisitados informações encontradas nas capas das mídias compradas, tais como nome original do filme, nome traduzido, duração, sinopse, país, idiomas, atores, diretores, etc. Os atores e diretores dos filmes também necessitam ser cadastrados caso não o tenham sido previamente, e estes serão utilizados como forma de pesquisa tanto no sistema de gerenciamento como no sistema de divulgação.

Após o cadastramento do filme é necessário cadastrar as mídias relativas a este filme. As mídias são os elementos que possuem valor agregado neste sistema, pois são elas que são compradas e serão elas que serão utilizadas pelos associados do sistema. È necessário cadastrar no sistema todas as mídias que foram compradas de um filme, pois elas serão utilizadas para formarem o carrinho de locações do associado. No momento do cadastramento de uma mídia o sistema cria um código de barra único para cada mídia a fim de identificar e automatizar o processo de locação e de pesquisa de mídias. Esse processo de criação de código de barra é explicado a seguir.

O código de barra é uma funcionalidade que garante uma maior automatização dos processos mais correntes de uma locadora. Essa funcionalidade é aplicada aos elementos do sistema que são mais utilizados em transações. Analisando o ramo de negócio de uma locadora verificamos que esses elementos seriam os associados da locadora, as mídias de uma locadora e os filmes que já possuem código de barra, pois são colocados na capa dos dvd's ou vhs's comprados.

O código de barra tem essa importância na automação, pois com uma máquina leitora de código de barra é possível ler um código de barra e jogar os números interpretados pela máquina no campo em que o cursor do teclado está localizado.

Assim, o código de barra poderá ser utilizado nas mídias que foram compradas e também nas carteirinhas dos associados. Um problema que foi solucionado é que o código de barra deve ser um número único com um tamanho fixo.

Para isso foi implementado um algoritmo de forma a ter o número de identificação do elemento no banco (ID), o número de identificação da loja em questão (ID da loja), um número que identifica se o elemento é uma mídia ou um associado e um número que é o dígito verificador, que valida o código de barra. O algoritmo referente à criação do código de barra é mostrado abaixo:

```
public static String generateMidiaCodigoBarra(Midia midia) {
    String codigoBarra = "";
    codigoBarra = SisLocStatics.CODIGO_BARRA_CLASSE_MIDIA +
                  StringUtil.lpad(midia.getLoja().getId().toString(), "0", 5) +
                   StringUtil.lpad(midia.getId().toString(), "0", 6);
    codigoBarra += generateDigit(codigoBarra);
```

```
    return codigoBarra;
}
```

No método estático *generateMidiaCodigoBarra*, o código de barra é gerado utilizando um número constante referente ao elemento mídia, um outro número gerado de forma a ter um tamanho de 5, que utiliza o id da loja, e também com mais um número gerado de forma a ter um tamanho de 6. Ao final é gerado o digito verificador a partir do número de tamanho 12 gerado anteriormente.

```
public static String lpad(String value, String pad, int numChars) {
    String result = "";
    int valueLength = value.length();
    if (valueLength > numChars) {
       result = value.substring(0,numChars);
    } else {
       for (int i = 0; i < numChars - valueLength; i++) {
          result += pad;
       }
       result = result + value;
    }
    return result;
}
```

Esse método constrói um número de tamanho máximo *numChars* e que completa o número passado através da variável *value* com o valor passado através da variável *lpad*, até completar o tamanho máximo *numChars*.

```
private static String generateDigit(String codigoBarra) {
    String digit = null;
    int length = codigoBarra.length();
    int sum = 0;
    boolean even = false;

    for (int index = 0; index < length; index++) {
       if (index % 2 != 0) {
          sum += Integer.valueOf(codigoBarra.substring(index, index + 1)).intValue()
* 3;
       } else {
          sum += Integer.valueOf(codigoBarra.substring(index, index + 1)).intValue()
* 1;
       }
    }
    if (sum % 10 == 0) {
       digit = "0";
    } else {
       double nextDecimal = ((Math.ceil(sum / 10) + 1) * 10);
       digit = new Integer((new Double(nextDecimal - sum).intValue())).toString();
    }
    return digit;
}
```

O método *generateDigit*, gera um dígito verificador a partir do número criado nos métodos anteriormente explicado.

Após o cadastramento dos filmes e das mídias associadas a estes filmes, pode-se começar a cadastrar os associados do sistema. Só que antes de cadastrar um associado é necessário cadastrar os planos mensais que essa loja irá possuir. Os planos como explicado anteriormente contêm informações sobre a quantidade de mídias que um associado poderá alugar e do quanto que esse associado deverá pagar mensalmente à locadora para usufruir desse plano. Então após o cadastramento dos planos mensais, os associados podem ser cadastrados.

O associado é o agente mais importante do sistema, pois é o associado que move todo o modelo de negócio do sistema de Gestão. O associado é o sócio da locadora e é o agente que garantirá a agregação de valor ao novo sistema desenvolvido.

Por ter grande importância no sistema e por ser o sistema um modelo de negócio que se baseia na confiança mútua entre cliente e fornecedor, os dados do cliente devem conter as mais fidedignas informações. Entretanto para garantir as validades dessas informações foram criados no sistema algoritmos e funcionalidades que validam e guardam tais informações fornecidas pelo associado.

Entre essas funcionalidades temos a possibilidade de cadastramento de foto do associado e também a validação de CPF, pois é o CPF o número de identificação e o documento legal para identificação do cidadão brasileiro. Esses dois métodos diminuem bastante a possibilidade de uma pessoa se cadastrar na locadora, alugar filmes e depois nunca mais devolver os filmes, pois com estes dados e com os dados do cadastro em geral já é possível acionar essa pessoa nas esferas judiciais. Abaixo segue o algoritmo de validação de CPF.

```
public static boolean checkCpf( String cpf ) {
    int d1 = 0;
    int d2 = 0;
    int digito1 = 0;
    int digito2 = 0;
    int resto = 0;
    int digitoCPF = 0;
    String nDigResult;
    for( int nCount = 1; nCount < cpf.length() - 1; nCount++ ) {
       digitoCPF = Integer.valueOf( cpf.substring( nCount - 1, nCount ) ).intValue();
       d1 = d1 + ( 11 - nCount ) * digitoCPF;
       d2 = d2 + ( 12 - nCount ) * digitoCPF;
    }
    resto = ( d1 % 11 );
    if( resto < 2 ) {
       digito1 = 0;
    } else {
       digito1 = 11 - resto;
    }
    d2 += 2 * digito1;
    resto = ( d2 % 11 );

    if( resto < 2 ) {
       digito2 = 0;
    } else {
       digito2 = 11 - resto;
```

```
    }
    String nDigVerific = cpf.substring( cpf.length() - 2, cpf.length() );
    nDigResult = String.valueOf( digito1 ) + String.valueOf( digito2 );
    return nDigVerific.equals( nDigResult );
}
```

Realizado estes cadastramentos, o sistema já pode ser utilizado para o seu principal objetivo que é a locação das mídias. Para este objetivo, o sistema possui duas funcionalidades que permitem a locação de mídias. Uma funcionalidade é utilizada quando o associado está presente na loja e escolhe os filmes que deseja alugar. A outra foi pensada no caso em que o associado não está presente na loja e então liga para a loja desejando alugar as mídias que ele já pesquisou através do sistema de divulgação. Essa funcionalidade funcionará como entrega em domicílio e foi pensada na perspectiva de que a locadora possui entregadores que levarão as mídias alugadas pelo associado na sua casa.

A lógica do fluxo da funcionalidade de locação por telefone é diferente da locação na loja. Na locação na loja o carrinho de locação é composto de mídias, pois são as mídias que o associado escolheu na prateleira da loja, enquanto que na locação por telefone o carrinho de locação é composto de filmes, pois são os filmes que o associado especificou para o funcionário através do telefone.

Outra diferença está no cálculo de mídias que o associado pode possuir em seu poder, pois na locação por telefone o sistema faz o cálculo permitindo que o associado alugue as mídias que deseja mesmo se contabilizando o total de mídias atuais e as mídias que possui em casa ultrapasse o plano mensal do associado.

O sistema permite isso porque na locação por telefone quem levará as fitas até a casa do associado são os entregadores da loja, então se o associado possui mídias na sua casa e se o total de mídias ultrapassa o permitido, o entregador pegará as mídias que estão na casa do associado e retornará as mesmas para a locadora e retornando também no sistema.

Utilizando estes dois módulos os outros módulos do sistema podem ser utilizados de forma a exibir ou executar os dados utilmente. A funcionalidade de retorno de mídias permite retornar as mídias que foram alugadas pelos associados e com isso permite ao associado alugar novas mídias caso estivesse com o total de mídias alugadas em número máximo permitido. O retorno de mídias também permite que associados que estivessem na fila de espera pelo filme da mídia retornada, através da lista de desejo, alugue a mídia em questão.

A funcionalidade de lista de desejo é utilizada para permitir aos associados do sistema uma forma de guardar os filmes que o associado queira alugar, mas que por questão de que todas as mídias desse filme já estão alugadas não é possível alugar neste momento. Assim, o associado pode preencher a sua lista de desejo e conforme as mídias vão sendo retornadas ao sistema e à locadora, o sistema informará na tela para o funcionário se o filme daquela mídia consta para algum associado na lista de desejo. Com isso, o funcionário pode acessar o módulo de lista de desejo e contactar o associado informando-lhe da mídia em questão.

Os relatórios são acessados para exibir aos usuários do sistema informações sobre as mídias que estão alugadas, sobre o tempo de locação das mídias, dos associados que têm quantidade de mídias alugadas superior ao seu plano mensal, do histórico de locações da loja e também é possível gerar relatórios dinâmicos como forma de permitir ao gerente da loja maiores informações sobre os dados do sistema, já que estes relatórios dinâmicos também podem ser exportados para diversos tipos de arquivos como XML, XLS e PDF.

O relatório dinâmico utiliza a as funcionalidades providas pela biblioteca DisplayTag, já explicada anteriormente. Parte do código que contém a lógica de implementação é mostrada a seguir:

```
Session session = HibernateUtil.getSession();
 try {
    Connection conn = session.connection();

    PreparedStatement ps = null;
    ResultSet rs = null;

    ps = conn.prepareStatement(queryString);
    rs = ps.executeQuery();
    results = new RowSetDynaClass(rs, false);
} catch (HibernateException e) {
    return ERROR;
} catch (SQLException e) {
    return ERROR;
}
```

O atributo *queryString* contém a *string* que foi digitada pelo usuário do sistema na tela de geração do relatório dinâmico. Essa *string* é referente ao comando SQL que o usuário deseja executar e que retornará as linhas a serem exibidas na tela, e este resultado é colocado no atributo *results*, através do objeto *RowSetDynaClass* que é provido pelo arquivo *.jar* da biblioteca DisplayTag.

A biblioteca DisplayTag disponibiliza também o uso da *tag* <*display*> que será a responsável por exibir a tabela de dados e também pela exportação dos dados para os diversos tipos de arquivos. Abaixo verificamos como esse resultado é listado na tela através da *tag display*.

```
<display:table name="results.rows" requestURI= "/queryExecute.action"
export="true" />
```

A última funcionalidade implementada permite o controle da geração da cobrança dos associados. A cobrança dos associados é realizada verificando o plano mensal relacionado de cada associado titular e assim criada uma linha na tabela de cobrança_associado referente ao mês/ano da cobrança, o valor, o pagamento ou não e o valor pago pelo associado.

4.2 Módulo de Divulgação

O módulo de divulgação é responsável por disponibilizar as informações da locadora para o público. Além de ser um meio de divulgação este módulo também permite que os associados da locadora, efetuando o *login* pelo site, possam interagir com a locadora, pois o sistema possui algumas funcionalidades reservadas a esse fim.

O módulo de divulgação utiliza as mesmas tecnologias já especificadas no módulo de gerenciamento. O primeiro passo para acessar o site de divulgação é digitar a URL *domínio/site/index.action,* onde domínio referencia a URL de acesso à raiz do servidor Tomcat, que é o servidor onde a aplicação foi instalada.

Ao acessar esta URL o usuário é levado a pagina inicial do sistema de divulgação. Esse site foi criado utilizando os conceitos que vem sendo utilizados por grandes portais da Web como, por exemplo, o site da Globo.com, que é a navegação por abas. Esse tipo de navegação auxilia o usuário a encontrar mais rapidamente a página que deseja, pois as abas contêm as palavras-chaves mais importantes relativos à locadora.

Assim, como esse sistema não é um sistema grande foram criadas somente três abas que foram divididas em "Inicial", "Informação" e "*E-mail*", que encaminha o usuário respectivamente para página que contém as funcionalidades mais importantes do *site*, para a página que contém informações sobre o novo modelo de negócio o qual a locadora é orientada e por fim para a página que permite ao usuário enviar um e-mail para a locadora. Um trecho do código da página JSP que exibe as abas é visto a seguir.

```
<ww:if test="associadoInfo!=null">
    <ww:if test="(selectedTab == 1) || (selectedTab == 0)">
      <td><a href="home.action?selectedTab=1" class="selected"><img
src="assets/img/home_05.gif" border="0" title="Inicial"></a
    </ww:if>
    <ww:else>
      <td><a href="home.action?selectedTab=1"><img
src="assets/img/home_05.gif" border="0" title=" Inicial"></a
    </ww:else>
</ww:if>
<ww:else>
    <ww:if test="(selectedTab == 1) || (selectedTab == 0)">
       <td><a href="index.action?selectedTab=1" class="selected"><img
src="assets/img/home_05.gif" border="0" title=" Inicial"></a
    </ww:if>
    <ww:else>
       <td><a href="index.action?selectedTab=1"><img
src="assets/img/home_05.gif" border="0" title=" Inicial"></a
    </ww:else>
</ww:else>
```

Como percebemos, as abas são figuras que são carregadas e para exibir um aspecto de seleção é utilizada as propriedades de estilo que são configuradas em arquivo CSS e carregadas através do atributo *class* dentro das *tags*.

A primeira página, que é acessada pela aba "Inicial", exibe oito filmes que foram cadastrados como lançamentos nos últimos 60 dias. Para que haja um rodízio dos oito filmes a serem exibidos foi implementado um algoritmo que mistura os filmes da lista de lançamentos e assim exibe-os na página inicial. O algoritmo que cria a lista aleatória de filmes é mostrado a seguir.

```
public ArrayList randomize(Collection[] listaFilmes, boolean disponivel) {
    Set colecao = createFilmePool(listaFilmes, disponivel);
    ArrayList listaAleatoria = null;
    Random indiceAleatorio = new Random();
    int tamanhoTotal = colecao.size();
    listaAleatoria = new ArrayList(tamanhoTotal);
    for (int i = 0; i < tamanhoTotal; i++) {
       boolean flag = false;
       while (!flag) {
          int foundPos = indiceAleatorio.nextInt(tamanhoTotal);
          Object obj = (colecao.toArray()[foundPos]);
          Filme filme = getFilme((Object) obj);
          if (!listaAleatoria.contains(filme)) {
```

```
                listaAleatoria.add(i, filme);
                flag = true;
            }
        }
    }
    return listaAleatoria;
}
```

Pela análise do código verificamos que antes é criada uma coleção de filmes que possuem mídias disponíveis (variável *coleção*), depois é utilizada a classe *Random* que gera números inteiros aleatórios de valor máximo igual ao tamanho da lista, e então se obtém o objeto da lista cujo índice é esse número aleatório.

Na página inicial é possível utilizar os principais recursos do sistema que são as diversas formas de pesquisa de filmes. A pesquisa é realizada utilizando somente o campo de busca que é exibido na página. Para efetuar a pesquisa o usuário necessita unicamente digitar a palavra que deseja e o sistema retornará todos os resultados encontrados para aquela palavra.

O sistema retornará os seguintes resultados quanto às palavras a serem pesquisadas: os filmes TOP 5 que possuam em seu nome original ou traduzido a busca requisitada, todos os filmes que contenham em seu nome original ou traduzido a busca requisitada, os atores ou atrizes que possuam em seu nome a busca requisitada e os diretores que possuam em seu nome a busca requisitada.

Todos esses resultados serão apresentados numa única tela de forma organizada e selecionada. Para exibir tal organização, foi utilizada novamente a característica de abas e estilos por CSS. Cada aba possui um nome referente ao tipo de resultado da pesquisa que são respectivamente: "*Top 5*", "Filme", "Ator/Atriz" e "Diretores".

Cada resultado de cada aba possui um *link* que levará o usuário para o detalhamento do resultado. Por exemplo, se o usuário *clicar* em um resultado referente à aba "*Top 5*" ou "Filme" será levado para o detalhamento do filme, pois os resultados dessas abas exibem os nomes dos filmes; para o caso de clicar em um resultado da aba "Ator/Atriz" será levado para o detalhamento do ator ou atriz que conterá a lista dos filmes em que esses atores fazem parte do elenco e assim acontece também com os diretores.

A aba "Informações" somente contém informações sobre o modelo de negócio da locadora e não apresenta nenhuma funcionalidade. A aba *E-mail* com um formulário onde o usuário pode enviar um e-mail para a loja a fim de enviar sugestões críticas e opiniões. O envio de e-mail foi implementado utilizando a API JavaMail [5] provida pela Sun.

O sistema de divulgação apresenta também algumas funcionalidades particulares aos associados da locadora. Quando o associado é cadastrado na locadora, o sistema de gerenciamento envia uma mensagem para o e-mail do associado contendo informações sobre o *username*, que é o e-mail cadastrado e a senha do associado para este se logar no site da locadora. Através dessa informação o associado está apto a usufruir das funcionalidades privativas que o site oferece.

Entre essas funcionalidades está a visualização do perfil que permite ao associado visualizar as suas informações cadastrais, tais como nome, endereço, telefones para contato, e-mail, plano mensal cadastrado. Outra funcionalidade importante é a visualização do histórico, que exibe ao associado uma lista paginada dos títulos que o associado ou sua família alugaram na loja e essa lista contém informações sobre o nome do filme, a data da locação e a data de retorno do filme à loja.

O site também permite que o associado dê notas aos filmes que existem na locadora. Essas notas são utilizadas para gerar os diversos *rankings* que existem no site

(*Top 5, Top 16, e Top 32*) e tem também a finalidade de informar aos outros visitantes do sistema como estão as cotações populares dos filmes. A funcionalidade de notas é disponibilizada em todos os filmes da loja e aparece abaixo da figura referente ao filme. Essa funcionalidade foi implementada utilizando AJAX e parte do código é mostrado abaixo.

```
<script>
      function retrieveURL(filmeid,conceito) {
            var url = 'addConceito.action';
            var pars = "filmeid="+filmeid+"&conceito="+conceito;
            var myVar = new Ajax.Updater('conceito'+filmeid,url,
      {method: 'get', parameters: pars});
      }
</script>
<div class="conceito" id="conceito<ww:property value="filme.id"/>">
<ww:if test="associadoInfo != null">
      <ww:set name="imageOn" value="'assets/img/g_myrate_on.gif'"/>
      <ww:set name="imageOff" value="'assets/img/g_myrate_off.gif'"/>
</ww:if>
<ww:else>
      <ww:set name="imageOn" value="'assets/img/g_rate_on.gif'"/>
      <ww:set name="imageOff" value="'assets/img/g_rate_off.gif'"/>
</ww:else>
<ww:iterator value="stars">
    <ww:if test="[0].lighted">
        <a href="javascript:retrieveURL(<ww:property
value="filme.id"/>,<ww:property value="[0].value"/>);"><img
src="<ww:property value="imageOn"/>"></a>
    </ww:if>
    <ww:else>
        <a href="javascript:retrieveURL(<ww:property
value="filme.id"/>,<ww:property value="[0].value"/>);"><img
src="<ww:property value="imageOff"/>"></a>
    </ww:else>
</ww:iterator>
<a href="javascript:retrieveURL(<ww:property
value="filme.id"/>,0);"><img src="assets/img/g_rate_skip.gif"></a>
</div>
```

O AJAX [19] utilizado no site atualiza os valores das notas de um associado referentes ao filme em questão. Essas notas são referenciadas pela iteração da coleção stars, onde cada valor dessa lista é especificado pelo seu índice que vai até 5. Assim quando o associado clica em uma figura o valor dessa figura (valor é referenciado pela variável *stars*) é enviado a uma *action* que atualiza os dados da tabela referentes ao conceito dos filmes.

A funcionalidade mais importante do sistema de divulgação quanto aos seus associados é a funcionalidade que permite ao associado adicionar os filmes que deseja alugar à sua lista de desejos. Essa lista de desejo será verificada posteriormente pelos funcionários da loja e caso possua algum filme que tenha mídias disponíveis o funcionário entra em contato com o associado e informa-lhe sobre a disponibilidade dos filmes.

Através do site o associado poderá então adicionar e remover filmes à sua lista de desejo. Lembrando sempre que a quantidade máxima de filmes permitidos é indicada pelo plano mensal de cada associado.

5 Resultados

Os resultados encontrados durantes as fases de testes demonstram que o sistema é estável e possui uma boa relação tempo/resposta por parte da aplicação.

Através dos resultados verificamos que o sistema funcionou de acordo com o modelo de negócio proposto e a adição de algumas funcionalidade particulares fizeram do sistema uma nova oportunidade no ramo de locadoras. A utilização do Hibernate como *framework* de controle dos dados do banco de dados acelerou o andamento do projeto e garantiu a persistência transacional das informações do sistema.

Quanto à base de dados verificamos que o modelo relacional proposto funcionou conforme especificado, pois através do sistema de gerenciamento conseguimos gerenciar as principais atividades de uma locadora e ainda prover informações para o módulo de divulgação. Essa característica sem dúvida pode trazer um diferencial ao ramo de locadoras, pois agrega novas características aos clientes da locadora, pois permite que estes interajam com a locadora através da internet.

Além dos resultados quanto ao funcionamento da aplicação baseado neste novo modelo de negócio, o projeto permitiu verificar também que o WebWork é uma excelente *framework,* que auxilia e aumenta a produtividade do desenvolvedor em projetos Web e que possui características à altura do Struts e que futuramente pode vir a substituí-lo como *framework* de controle de dados de aplicações Web.

Como o WebWork foi utilizado fortemente nesse projeto, tais informações podem vir a ser úteis futuramente para projetos semelhantes que venham a utilizar a plataforma J2EE.

Os resultados visuais das interfaces dos usuários podem ser encontrados na Especificação de Requisitos de Software na seção Interface dos Usuários 3.2.1.

6 Conclusão

Com o intuito de se criar uma nova oportunidade no ramo de locadoras, onde verificamos que existe um modelo de negócio tradicional onde algumas questões não atendem as necessidades do consumidor, foi desenvolvido um sistema Web no qual possui em suas funcionalidades e arquitetura a utilização de ferramentas modernas e que visam atender a expectativa do consumidor tanto em relação ao modelo de negócio como em divulgar de forma transparente e moderna as informações da locadora.

O sistema foi desenvolvido utilizando em sua maioria ferramentas consagradas de desenvolvimento Web, adotando padrões de projeto conceituados e indicados por fundações especializadas em desenvolvimento de sistemas empresariais, tornando assim a codificação do sistema uniforme e organizada.

Apesar do pouco tempo de desenvolvimento o projeto pôde ser concluído com êxito, visto a experiência do desenvolvedor em projetos semelhantes e as ferramentas utilizadas já terem sido largamente utilizadas anteriormente. Por ser um sistema que utiliza a linguagem Java, também foi um fator do rápido desenvolvimento, pois tal linguagem possui muitas referências na Internet, além de prover uma quantidade imensa de ferramentas para desenvolvimento de sistemas Web, onde segurança e consistência dos dados são fatores de suma importância.

Como trabalhos futuros, podem ser realizados vários *upgrades* no sistema. Como uma forma de tornar o sistema mais completo e dinâmico pode ser implementado novas funcionalidades que permitam o sistema funcionar tanto no sistema de planos mensais como no sistema locação de mídias avulsas. Essa característica poderia ser desenvolvida juntamente com a base do sistema que funciona baseado em planos mensais, onde poderia ser adicionado um plano avulso onde a lógica de locação seria alterada, pois seria preciso identificar tais associados e permitir somente a locação de mídias caso o associado possuísse créditos na loja, por exemplo.

Outra funcionalidade poderia ser a de aumentar a segurança, adicionando a identificação por leitor de digital. Esse tipo de identificação já possui referências de desenvolvimento na Internet, bem como bibliotecas que já realizam todo o trabalho de identificação dos pontos da digital humana. Através dessa funcionalidade a segurança em relação à confiança fornecedor-cliente seria transferida para a aplicação.

Uma outra funcionalidade que foi implementada pela metade no sistema e que poderia ser desenvolvida é a da possibilidade de os usuários tirarem foto dos associados através da própria tela. A funcionalidade foi implementada até a parte de permitir o usuário escolher se deseja tirar fotos pela *Webcam* ou se deseja colocar uma foto passando o caminho relativo da foto que já está salva no HD. Assim, a implementação do programa que disponibiliza a foto na página deveria ser implementado e este poderia ser desenvolvido utilizando *Java Applets*.

7 Referência Bibliográfica

[1] Ant Demystified, http://www.sitepoint.com/article/apache-ant-demystified, acessado em 15 de setembro de 2006

[2] Ant Apache, http://ant.apache.org/, acessado em 13 de outubro de 2006

[3] Design Patterns, http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html, acessado em 07 de setembro 2006

[4] DisplayTag, http://displaytag.sourceforge.net/, acessado em 10 de novembro de 2006

[5] JavaMail, http://java.sun.com/products/javamail/, acessado em 20 de novembro de 2006

[6] Eclipse, http://www.eclipse.org/, acessado em 10 de setembro de 2006

[7] Hibernate.org, http://www.hibernate.org, acessado em 05 de outubro de 2006.

[8] Hibernate in Action by Christian Bauer, Gavin King, Nick Heudecker, Patrick Peak, August 2004

[9] J2EE Documents, http://java.sun.com/j2ee/1.4/docs/tutorial/doc/, acessado em 23 de novembro de 2006

[10] J2EE Sun: http://java.sun.com/developer/onlineTraining/J2EE/Intro2/j2ee.html, acessado em 07 de setembro de 2006

[11] Log4J Project, http://logging.apache.org/log4j/docs/, acessado em 10 de outubro de 2006.

[12] Página da disciplina Engenharia de Software, http://www.del.ufrj.br/~ac/eel873.htm, acessado em 11 de novembro de 2006

[13] WebWork in Action Patrick Lightbody and Jason Carreira Foreword by Rickard Öberg, September 2005

[14] WebWork, http://www.opensymphony.com/webwork/, acessado em 10 de outubro de 2006

[15] Tomcat, http://tomcat.apache.org/, acessado em 25 de outubro de 2006

[17] MySQL, http://www.mysql.com, acessado em 20 de novembro de 2006

[18] DBDesigner 4, http://fabforce.net/dbdesigner4/, acessado em 25 de novembro de 2006

[19] AJAX, http://www.w3schools.com/ajax/default.asp, acessado em 10 de novembro de 2006

Apêndice – Especificação de Requisitos de Software

1 Introdução

1.1 Finalidade

O objetivo deste documento é descrever as Especificações de Requisito de Software a fim de facilitar o entendimento do sistema pela equipe de desenvolvimento e por seus usuários, bem como para os próprios avaliadores. Neste documento serão especificados o detalhamento da solução.

1.2 Escopo

O projeto desenvolvido é o Sistema de Gerenciamento de Locadora – SisLoc, cujo objetivo é o de gerenciar as atividade de uma locadora conforme um novo modelo de negócio, baseado em planos mensais. O SisLoc engloba um módulo de gerenciamento que é o módulo administrativo ou de gestão e também engloba o módulo de divulgação que permite ao público obter informações sobre a locadora e os filmes que esta possui.

O SisLoc é um sistema Web, desenvolvido de forma a ter uma interface limpa e fácil de usar e que permita uma automação dos processos mais comuns que ocorrem numa locadora, que são as buscas por associados e filmes e as locações de mídias.

1.3 Definições, Acronismos e Abreviaturas

- ERS – Especificação de Requisitos de Software
- DD – Dicionário de Dados
- DER – Diagrama de Entidade Relacionamento
- DS – Diagramas de Seqüências

1.4 Referências

Java Development Kit (JDK) 1.5, Apache Tomcat 5.0, Hibernate 3.0, WebWork 2.0, DisplayTag.

1.5 Resumo

No restante deste documento, serão descritas com detalhes as principais funções do sistema SisLoc, definindo os seus diferentes requisitos e funções, bem como caracterizados o público alvo e as restrições do projeto.

Para isso, utilizaremos uma modelagem conceitual orientada a objeto ao invés da modelagem estruturada. Com o tipo de modelagem definida, decidimos utilizar a padronização UML, que se trata de padronizar a modelagem orientada a objetos de uma forma que qualquer sistema, seja qual for o tipo, possa ser modelado corretamente, com

consistência, fácil de comunicar com outras aplicações, simples de ser atualizado e compreensível.

Posteriormente definiremos todas as regras de negócio e os diagramas de casos de uso. Em seguida serão realizados os diagramas de classes, o diagrama de seqüências e, por fim, a modelagem do banco de dados.

Para a construção do modelo de entidade relacionamento foi utilizado o software DB4Designer.

2 Descrição Geral

2.1 Perspectiva do Produto

O sistema SisLoc não é dependente de nenhum outro sistema e suas funcionalidades e operação utilizam obrigatoriamente recursos próprios do sistema. Esses recursos são:

- Hospedagem do servidor Web - necessita de uma máquina com um servidor Tomcat Apache 5.0.28 instalado. O servidor Tomcat [15] é um servidor Java que traduz as instruções do código Java compilado para os *browsers* de internet interpretarem corretamente.
- Hospedagem do Sistema de Gerenciamento de Banco de Dados (SGBD) MySQL – utiliza a mesma máquina do servidor Web, no qual vai ser instalado o MySQL.
- *Java Development Kit* (JDK) 1.5 – Kit de Desenvolvimento Java que contém a máquina virtual Java que interpretará os códigos compilados. Esse kit é usado pelo servidor Java que traduzirá os códigos compilados para execução do sistema.

O sistema SisLoc foi desenvolvido utilizando a linguagem de programação Java (J2EE) e utilizando duas *frameworks* que visam facilitar a programação para Web. Essas *frameworks* são Hibernate e WebWork.

A *framework* Hibernate é uma aplicação *freeware* e *opensource* que persiste os objetos modelados através de linguagem Java a partir do modelo do banco de dados da aplicação. Com isso as operações executadas pelas classes do modelo de negócio são transacionais, assim o banco de dados da aplicação permanece consistente com seus dados. Além de gerenciar transações com o banco de dados o uso do Hibernate facilita a criação de *queries* que às vezes se tornam muito complexas e dificultam o andamento do projeto.

A *framework* WebWork é uma aplicação que facilita o acesso a dados da parte responsável pelos formulários exibidos e preenchidos pelos usuários, essa parte chama-se Vista da aplicação e segue a orientação MVC (*Model-View-Controller*), onde *Model* é a parte responsável pelos dados da aplicação, *View* é a visão da aplicação, onde os dados são transformados em informação inteligível para o usuário e *Controller* a parte responsável pelo controle, pelos métodos de negócio que a aplicação envolve. A *framework* WebWork é responsável pelo controle dos dados enviados dos formulários das páginas *jsp*'s (*Java Server Page*) e também pelas *actions* que orientam o fluxo da aplicação Web.

2.2 Funções do Produto

O produto foi dividido em dois módulos distintos, sendo o primeiro módulo responsável pelas funcionalidades de administração da locadora e o segundo módulo responsável pela divulgação de informações da locadora para o público e para os associados da loja.

2.2.1 Módulo de Gerenciamento

- Inserir/Editar Loja;
- Inserir, Listar Usuários;
- Editar, Remover Usuários (apenas super-usuário);
- Inserir, Editar, Remover, Listar Diretores;
- Inserir, Editar, Remover, Listar Atores;
- Inserir, Editar, Remover, Pesquisar Filmes;
- Inserir, Editar, Remover Mídias;
- Inserir, Editar, Remover, Listar Planos Mensais.
- Inserir, Editar, Ativar/Desativar, Pesquisar Associados;
- Alugar mídias para associados na loja.
- Alugar mídias para associados por telefone.
- Incluir, Remover filmes na lista de desejo.
- Gerar relatórios (mídias em locação, atividade de locação, histórico de locações, dinâmicos).
- Gerar Cobrança (cobrança mensal).
- Sair do Sistema.

2.2.2 Módulo de Divulgação

- Pesquisar Filmes por Categoria
- Pesquisar Filme por Palavra
- Pesquisar Ator por Palavra
- Pesquisar Diretor por Palavra
- Login de Associado
- Visualizar Perfil
- Avaliar Filmes
- Visualizar Histórico
- Lista de Desejo

2.3 Características do Usuário

O usuário que possui acesso ao sistema SisLoc é obrigatoriamente um funcionário da loja. Esse usuário poderá ser cadastrado por qualquer outro funcionário da loja desde que o último possua privilégios de inserção de usuários.

Ao cadastrar um usuário são configuradas as permissões de acessos a funcionalidades que o usuário irá possuir e somente essas funcionalidades estarão disponíveis para o usuário.

O site da locadora SisLoc permite acesso livre de qualquer visitante ao seu conteúdo. O site terá como finalidade o de informar sobre os filmes que são cadastrados na locadora e também caso o visitante seja um associado da locadora, este poderá se logar no sistema do site e com isso visualizar os seus dados cadastrais, avaliar os filmes do sistema, atribuindo notas a estes e também adicionar filmes à lista de desejo.

No momento do cadastro de um associado, o sistema SisLoc envia um e-mail com os dados de *login* do associado. Estes dados serão compostos pelo e-mail do associado e pela senha que é gerada pelo sistema. Assim com esses dados o associado poderá se logar no site e desfrutar de todas as informações e comodidade oferecidas pelo sistema.

2.4 Restrições

O sistema SisLoc é multi-plataforma podendo funcionar tanto no Windows quanto no Linux, pois como utiliza a linguagem Java, quem interpreta o código Java compilado é a máquina virtual Java que é multi-plataforma.

A máquina servidora que contém o servidor Tomcat instalado deverá possuir uma razoável capacidade de disco e de memória RAM, pois nela serão instalados o servidor Tomcat, a JDK 1.5 e o banco MySQL, além de que durante a execução do sistema em produção após um longo período de funcionamento a massa de dados deverá possuir um tamanho razoável porque serão gerados muito dados visto a atividade comercial de uma locadora ser diária e intermitente. E como o sistema também grava as imagens relativas aos filmes e aos associados pode ser necessário reservar certo espaço de disco para a aplicação.

2.5 Pressupostos e Dependências

O único pressuposto do sistema é que os usuários deste sistema tenham acesso máquina servidora que contém a aplicação e que esse acesso possa ser realizado com o mínimo de atraso, visto que durante a execução do módulo de gerenciamento e visualizando a atividade diária de uma locadora, percebe-se que essa alterna momentos de picos de atividade durante o dia e a semana, assim uma boa resposta do sistema é importante para diminuir o tempo de atraso.

A máquina servidora poderá estar ligada à internet ou rede local. Caso esse pressuposto não seja obedecido, o sistema não será tão amigável com esses usuários.

3 REQUISITOS ESPECÍFICOS

3.1 Requisitos Funcionais

Apresentaremos a seguir os requisitos funcionais que o sistema possui segundo uma análise essencial e direcionada para os diversos tipos de usuários do sistema.

3.1.1 Regras de Negócio

Regras para Usuários

- Um usuário pode inserir, editar loja no sistema, se e somente se, possuir privilégios.
- Um usuário pode inserir, se e somente se, possuir privilégios.

- Um usuário pode resetar a senha do usuário ou editar e remover usuários do sistema, se e somente, for um super-usuário.
- Um usuário pode listar, inserir, editar, remover atores de filmes no sistema, se, e somente se, possuir privilégios.
- Um usuário pode listar, inserir, editar, remover diretores de filmes no sistema, se, e somente se, possuir privilégios.
- Um usuário pode pesquisar, inserir, editar, remover títulos de filmes no sistema, se, e somente se, possuir privilégios.
- Um usuário pode inserir, editar, remover várias mídias referentes a um título no sistema, se, e somente se, possuir privilégios.
- Um usuário pode pesquisar, inserir, editar, desativar associados ao sistema, se, e somente se, possuir privilégios.
- Um usuário pode listar, inserir, editar, remover planos mensais ao sistema, se, e somente se, possuir privilégios.
- Um usuário pode alugar mídias na loja para associados, se, e somente se, possuir privilégios.
- Um usuário pode alugar mídias por telefone para associados, se, e somente se, possuir privilégios.
- Um usuário pode retornar mídias dos associados, se, e somente se, possuir privilégios.
- Um usuário pode incluir, remover ou visualizar os itens da lista de desejo dos associados se e somente se, possuir privilégios.
- Um usuário pode visualizar relatórios se e somente se, possuir privilégios.
- Um usuário pode gerar cobrança dos associados da loja, se e somente se possuir privilégios.
- Um usuário pode sair do sistema.

Regras para Associados

- Um associado pode pesquisar filmes no site da locadora, buscando por diretor, ator ou por título do filme.
- Um associado pode avaliar filmes no site da locadora, atribuindo notas aos filmes da locadora.
- Um associado pode adicionar filmes a sua lista de desejo no site da locadora.
- Um associado pode visualizar o seu histórico de locações.
- Um associado pode visualizar o seu perfil contendo os dados cadastrais.

Regras para Visitantes

- Um visitante pode pesquisar filmes no site da locadora, buscando por diretor, ator ou por título do filme.
- Um visitante pode pesquisar filmes por categorias.

Regras do Sistema

- Uma loja contém zero, um ou muitos usuários cadastrados.
- Uma loja contém zero, um ou muitos planos cadastrados.
- Uma loja contém zero, um ou muitas mídias cadastradas.
- Um associado possui um plano associado a ele.
- Um plano pode ser associado a zero, um ou muitos associados.
- Um título possui zero, um ou vários diretores associados.
- Um diretor possui zero, um ou vários títulos (filmes) associados.

- Um título possui zero, um ou vários atores associados.
- Um ator possui zero, um ou vários títulos (filmes) associados.
- Um título possui zero, um ou vários idiomas associados.
- Um idioma possui zero, um ou vários títulos (filmes) associados.
- Um associado possui zero, um ou muitos eventos de cobrança.
- Um associado possui zero, um ou muitos filmes na lista de desejos.

A arquitetura do sistema será a seguinte:

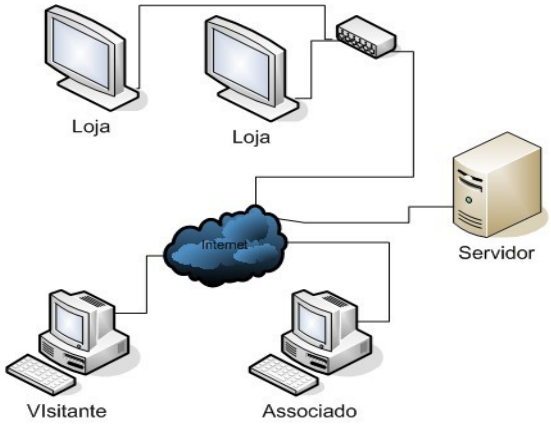

Figura 4: Arquitetura Simplificada do Sistema

Pela arquitetura simplificada nota-se que o sistema pode ser acessível de qualquer máquina com acesso a internet. Entretanto o sistema de gerenciamento, poderia ter o acesso permitido somente de máquinas pertencentes à locadora, visto a aumentar a segurança contra tentativas de acessos indevidos, mas este âmbito não cabe ao projeto proposto.

No caso da loja, é visualizada uma rede local onde as máquinas têm acesso à internet e acessam o aplicativo instalado no servidor. Os visitantes ou associados só teriam acesso ao sistema de divulgação. Os dois sistemas, tanto o de gerenciamento quanto o de divulgação estão instalados e rodando no mesmo servidor Tomcat e acessam o banco de dado MySQL através das configurações de conexão setadas nos arquivos XML, como já explicado anteriormente.

3.1.2 Diagrama de Entidades e Relacionamentos

O sistema usa o modelo lógico de dados que está representado no seguinte DER:

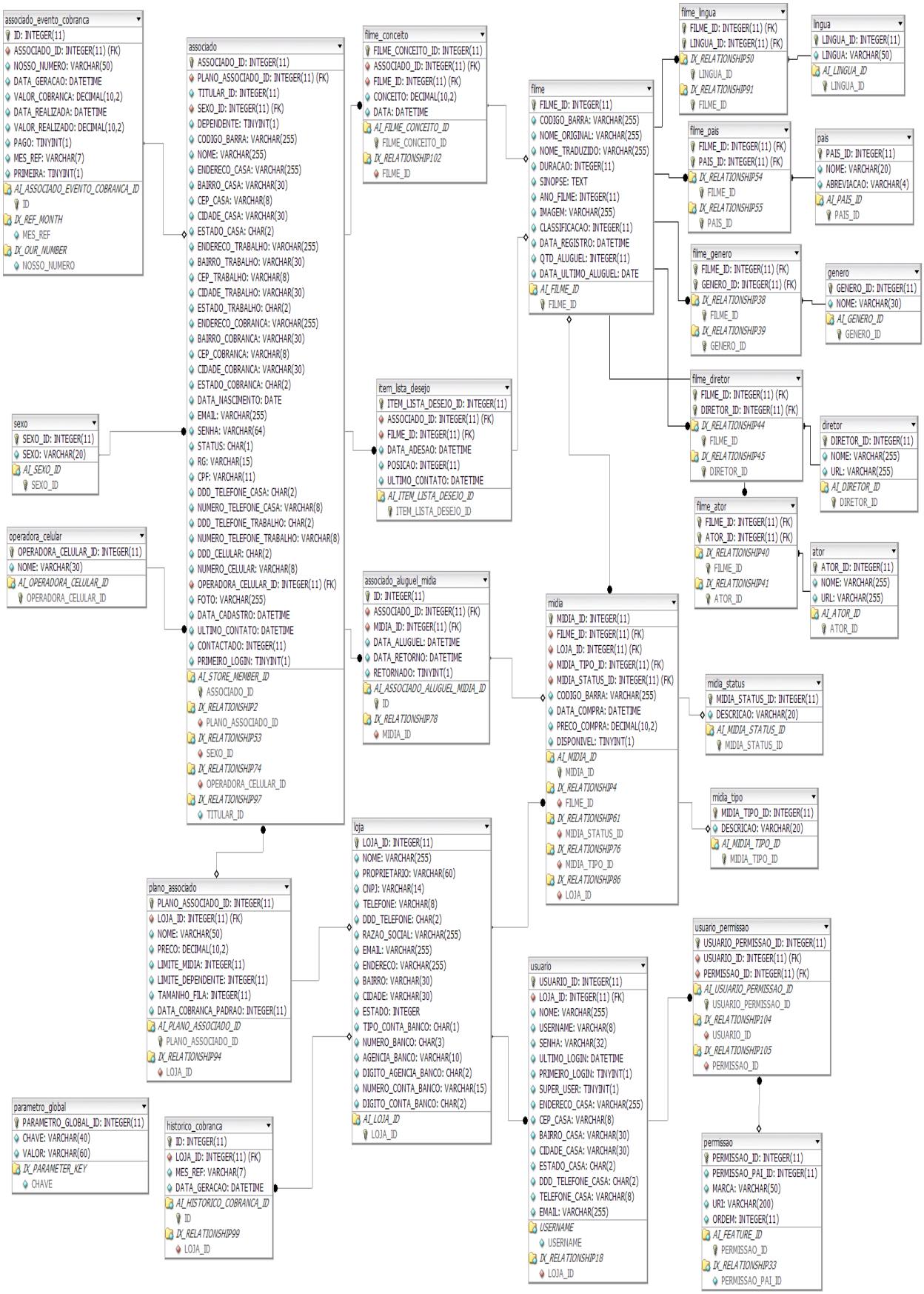

Figura 5: Diagrama de Entidade e Relacionamento do banco de dados do SisLoc

Para o nosso DER temos o seguinte dicionário de dados:

- **loja –** Entidade que define os dados relativos a loja ou locadora
  - **loja_id –** código que identifica unicamente uma loja
  - **nome –** nome da loja
  - **proprietário –** nome do proprietário da loja
  - **cnpj –** cadastro nacional de pessoa jurídica
  - **telefone –** número do telefone da loja
  - **ddd_telefone –** DDD do telefone
  - **razão_social –** razão social da loja
  - **email –** e-mail da loja
  - **endereco –** endereço da loja contendo o nome da rua e o número
  - **bairro –** bairro da loja
  - **cidade –** cidade da loja
  - **estado –** estado da loja
  - **cep –** CEP da loja
  - **numero_banco –** número que identifica o banco da loja
  - **agencia_banco –** número da agência bancária da loja
  - **digito_agencia_banco –** dígito verificador da agência bancária da loja
  - **numero_conta_banco –** número da conta corrente bancária da loja
  - **digito_conta_banco –** digito verificador da cc. bancária da loja

- **usuario –** Entidade que define os dados relativos aos usuários ou funcionários da loja
  - **usuario_id -** código que identifica unicamente um usuário
  - **loja_id -** código que identifica unicamente uma loja (chave estrangeira)
  - **nome –** nome do usuário
  - **username –** *username* que o usuário irá logar no sistema
  - **senha –** senha encriptada do usuário
  - **ultimo_login –** data que armazena o último *login* do usuário
  - **primeiro_login –** identifica se é o primeiro *login* do usuário para o sistema obrigá-lo a trocar a senha
  - **super_user –** identifica se o usuário é super-usuário
  - **endereço_casa -** endereço da casa do usuário contendo o nome da rua e o número
  - **bairro_casa –** bairro da casa do usuário
  - **cidade_casa –** cidade da casa do usuário
  - **estado_casa –** estado da casa do usuário
  - **cep_casa –** CEP da casa do usuário
  - **telefene_casa –** número do telefone da casa do usuário
  - **ddd_telefone_casa –** DDD do telefone da casa do usuário
  - **email –** e-mail do usuário

- **permissao –** Entidade que define os dados relativos as permissões de acesso dos usuários da loja
  - **permissao_id -** código que identifica unicamente uma permissão

    - **permissao_pai_id –** código que identifica unicamente o pai dessa permissão
    - **marca –** nome que identifica essa permissão
    - **uri –** link interno do sistema que define onde a *action* dessa permissão está localizada
    - **ordem –** posição que será exibida no menu

- **usuario_permissao –** Entidade associativa que relaciona usuários e permissões
    - **usuario_permissao_id** - código que identifica unicamente essa relação
    - **usuario_id -** código que identifica unicamente um usuário
    - **permissao_id -** código que identifica unicamente uma permissão

- **plano_associado –** Entidade que define os dados relativos aos planos mensais
    - **plano_associado_id -** código que identifica unicamente um plano
    - **loja_id -** código que identifica unicamente uma loja (chave estrangeira)
    - **nome –** nome do plano mensal
    - **preco –** preço do plano mensal
    - **limite_midia –** quantidade máxima de mídias alugadas permitida
    - **limite_dependente –** quantidade máxima de dependentes
    - **tamanho_fila –** quantidade máxima de itens na lista de desejo
    - **dia_cobranca_padrao –** dia da cobrança

- **sexo –** Entidade que define os dados relativos ao sexo
    - **sexo_id -** código que identifica unicamente um sexo
    - **sexo –** nome do sexo

- **operadora_celular -** Entidade que define os dados relativos as operadoras de celular
    - **operadora_celular_id -** código que identifica unicamente uma operadora celular
    - **nome –** nome da operadora de celular

- **associado –** Entidade que define os dados relativos aos associados da locadora
    - **associado_id -** código que identifica unicamente um associado
    - **loja_id -** código que identifica unicamente uma loja (chave estrangeira)
    - **plano_associado_id -** código que identifica unicamente um plano mensal (chave estrangeira)
    - **sexo_id -** código que identifica unicamente o sexo do associado (chave estrangeira)
    - **operadora_celular_id -** código que identifica unicamente a operadora de celular do associado (chave estrangeira)
    - **titular_id -** código que identifica unicamente o associado titular
    - **dependente –** identifica se o associado é um associado dependente
    - **código_barra –** código de barra pertencente ao associado
    - **nome –** nome do associado
    - **senha –** senha encriptada do associado
    - **endereço_casa -** endereço da casa do associado contendo o nome da rua e o número
    - **bairro_casa –** bairro da casa do associado

- **cidade_casa –** cidade da casa do associado
- **estado_casa –** estado da casa do associado
- **cep_casa –** CEP da casa do associado
- **endereço_trabalho -** endereço do trabalho do associado contendo o nome da rua e o número
- **bairro_trabalho –** bairro do trabalho do associado
- **cidade_trabalho –** cidade do trabalho do associado
- **estado_trabalho –** estado do trabalho do associado
- **cep_trabalho –** CEP do trabalho do associado
- **endereço_cobranca -** endereço de cobrança do associado contendo o nome da rua e o número
- **bairro_cobranca –** bairro de cobrança do associado
- **cidade_cobranca –** cidade de cobrança do associado
- **estado_cobranca –** estado de cobrança do associado
- **cep_cobranca –** CEP de cobrança do associado
- **data_nascimento –** data de nascimento do associado
- **status –** identifica o estado do associado (ativo ou inativo)
- **rg –** RG do associado
- **CPF –** CPF do associado
- **telefene_casa –** número do telefone da casa do associado
- **ddd_telefone_casa –** DDD do telefone da casa do associado
- **telefene_trabalho –** número do telefone do trabalho do associado
- **ddd_telefone_trabalho –** DDD do telefone do trabalho do associado
- **telefene_celular –** número do telefone celular do associado
- **ddd_telefone_celular –** DDD do telefone celular do associado
- **foto –** nome do arquivo da foto do associado
- **data_cadastro -** data do cadastro do associado
- **ultimo_contato –** data que a locadora realizou o último contato com o associado
- **primeiro_login -** identifica se é o primeiro *login* do associado ao site para o sistema obrigá-lo a trocar a senha
- **email –** e-mail do usuário

- **associado_evento_cobranca –** Entidade associativa que define os dados relativos a cobrança de cada associado da locadora.
  - **id -** código que identifica unicamente uma cobrança
  - **associado_id -** código que identifica unicamente um associado
  - **data_geracao –** data da geração da cobrança
  - **valor_cobranca –** valor da cobrança (relativo ao plano mensal)
  - **data_realizada –** data que o pagamento foi realizado pelo associado
  - **valor_realizado –** valor do pagamento realizado pelo associado
  - **pago –** identifica se a quitação do mês foi paga ou não
  - **mes_ref –** mês de referência da cobrança
  - **primeira –** identifica se é a primeira cobrança do associado para efetuar desconto

- **filme –** Entidade que define os dados relativos ao filme
  - **filme_id -** código que identifica unicamente um filme

    - **código_barra –** código de barra do filme
    - **nome_original –** nome original do filme
    - **nome_traduzido –** nome traduzido do filme
    - **duracao –** duração do filme em minutos
    - **sinopse –** sinopse do filme
    - **ano_filme –** ano de lançamento do filme
    - **imagem –** nome do arquivo que contém a imagem da capa do filme
    - **classificacao –** faixa etária mínima
    - **data_registro –** data de registro no banco de dados
    - **qtd_aluguel –** quantidade de vezes que o filme foi alugado
    - **data_ultimo_aluguel** – data do ultimo aluguel

- **midia_tipo –** Entidade que define os dados relativos ao tipo de mídia (VHS, DVD)
    - **id -** código que identifica unicamente um tipo de mídia
    - **descricao –** descrição do tipo de mídia

- **midia –** Entidade associativa que define os dados relativa as mídias pertencentes a um filme
    - **midia_id -** código que identifica unicamente uma mídia
    - **filme_id -** código que identifica unicamente um filme
    - **loja_id -** código que identifica unicamente uma loja
    - **midia_tipo_id -** código que identifica unicamente um tipo de mídia
    - **codigo_barra -** código de barra da mídia
    - **data_compra –** data da compra da mídia
    - **preco_compra –** preço de compra da mídia
    - **disponivel –** identifica se a mídia está ou não disponível (se foi alugada)

- **filme_conceito –** Entidade associativa que define os dados das notas dadas pelos associados aos filmes
    - **id -** código que identifica unicamente um conceito de um filme
    - **associado_id -** código que identifica unicamente um associado
    - **filme_id -** código que identifica unicamente um filme
    - **conceito –** nota dada pelo associado
    - **data –** data da realização da nota

- **item_lista_desejo –** Entidade associativa que define os dados dos itens da lista de desejo
    - **id -** código que identifica unicamente um item
    - **associado_id -** código que identifica unicamente um associado
    - **filme_id -** código que identifica unicamente um filme
    - **posicao –** posição do item na lista de desejo
    - **ultimo_contato –** último contato realizado
    - **data_adesao –** data da adesão do item

- **associado_aluguel_midia –** Entidade associativa que define os dados das mídias que foram alugadas por um associado
    - **id -** código que identifica unicamente uma locação da mídia pelo associado

    - **associado_id -** código que identifica unicamente um associado
    - **midia_id -** código que identifica unicamente uma mídia
    - **data_aluguel –** data da locação da mídia
    - **data_retorno–** data de retorno da mídia à locadora

- **lingua –** Entidade que define os dados relativos ao idioma de um filme
    - **lingua_id -** código que identifica unicamente um idioma
    - **lingua –** nome do idioma

- **filme_lingua –** Entidade associativa que define os dados relativos a associação entre filme e idioma
    - **filme_id -** código que identifica unicamente um filme
    - **lingua_id -** código que identifica unicamente um idioma

- **pais –** Entidade que define os dados relativos ao país de um filme
    - **pais_id -** código que identifica unicamente um país
    - **pais –** nome do país

- **filme_pais –** Entidade associativa que define os dados relativos a associação entre filme e país
    - **filme_id -** código que identifica unicamente um filme
    - **pais_id -** código que identifica unicamente um país

- **genero –** Entidade que define os dados relativos ao idioma de um gênero
    - **genero_id -** código que identifica unicamente um gênero
    - **nome –** nome do gênero

- **filme_genero –** Entidade associativa que define os dados relativos a associação entre filme e genero
    - **filme_id -** código que identifica unicamente um filme
    - **lingua_id -** código que identifica unicamente um gênero

- **ator –** Entidade que define os dados relativos ao ator de um filme
    - **ator_id -** código que identifica unicamente um ator
    - **nome –** nome do ator

- **filme_ator –** Entidade associativa que define os dados relativos a associação entre filme e ator
    - **filme_id -** código que identifica unicamente um filme
    - **ator_id -** código que identifica unicamente um país

- **diretor –** Entidade que define os dados relativos ao diretor de um filme
    - **diretor_id -** código que identifica unicamente um diretor
    - **nome –** nome do diretor

- **filme_diretor –** Entidade associativa que define os dados relativos a associação entre filme e diretor
    - **filme_id -** código que identifica unicamente um filme
    - **diretor_id -** código que identifica unicamente um diretor

3.1.3 Casos de Uso

O diagrama de caso de usos nos permite ter uma visão mais modular das funcionalidades do sistema. Isto é importante para a apresentação do sistema para pessoas leigas.

O sistema possui dois atores, sendo o ator "usuário", o ator mais importante que possui a maioria das funcionalidades do sistema e este ator é o responsável pelo gerenciamento das diversas operações que uma locadora possui.

Entretanto apesar de possuir um único ator responsável pelo gerenciamento, que nesse caso é o funcionário, cada funcionário possuirá acesso somente a funções do sistema que lhe foram configuradas no ato da sua criação. Este tipo de característica permite ao proprietário da loja atribuir funções de altas responsabilidades somente a funcionários de confiança.

O ator "associado" é o sócio da locadora e este possuirá acesso somente ao site da locadora, que possui as suas próprias funcionalidades.

Logo abaixo está esquematizado o diagrama de casos de usos, mostrando todas as funcionalidades do sistema que cada "ator" (usuário e associado) pode utilizar.

Figura 6: Casos de Uso do Sistema de Gerenciamento e Divulgação

3.1.4 Diagrama de Classes

O diagrama de classes representa toda estrutura dos objetos básicos que compõem o software. No diagrama de classes podem estar descritos, tanto relacionamentos entre objetos quanto entre classes, quando representamos classes abstratas.

O diagrama de classes nos permite ter uma visão mais ampla sobre a estrutura de todo o sistema, mas o exato funcionamento deste sistema é difícil de ser derivado a partir dele. Por isso precisamos de um outro diagrama, o de seqüência.

O diagrama de classes do software pode ser visto logo abaixo.

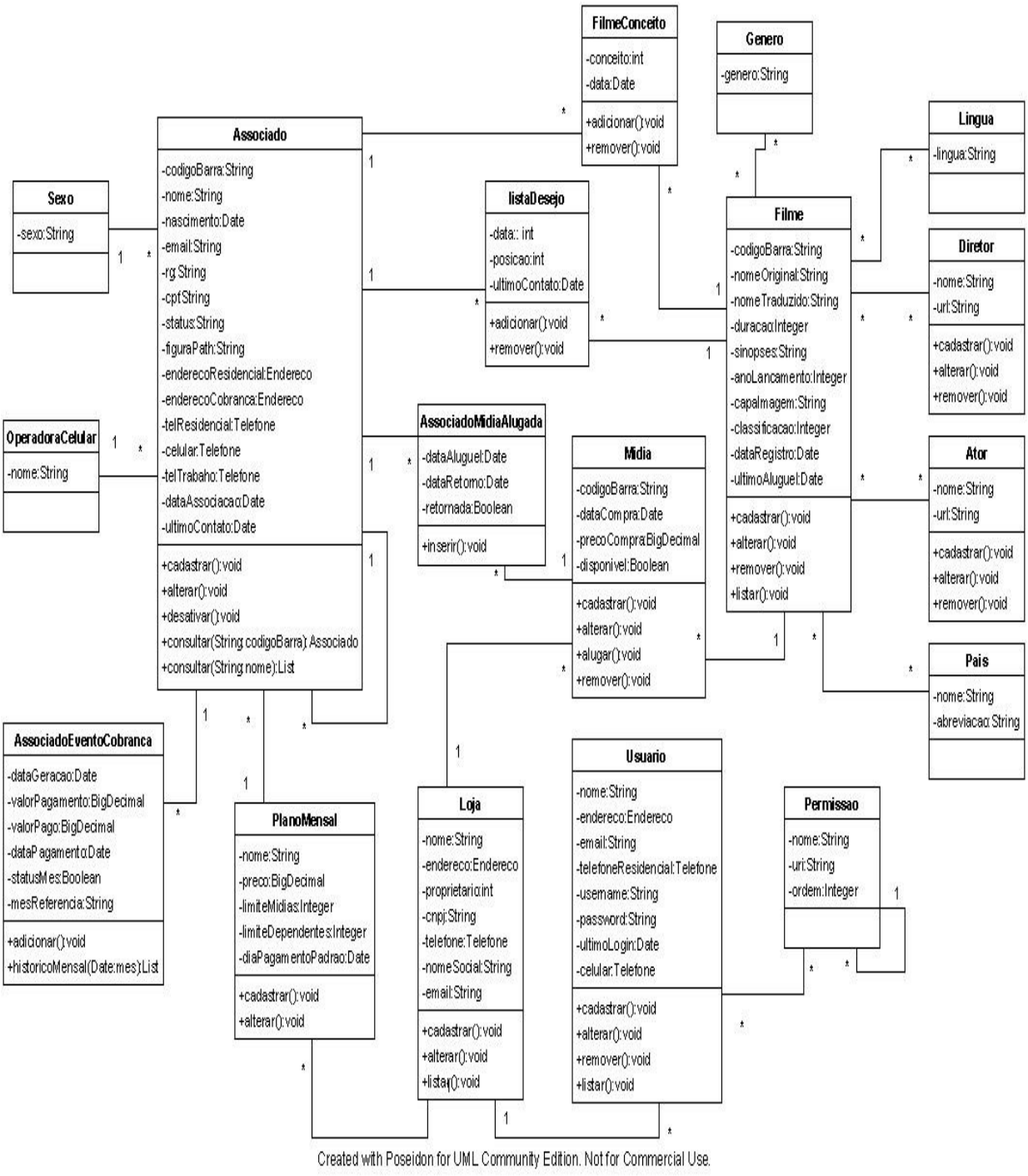

Figura 7: Diagrama de Classes do SisLoc

3.1.5 Diagrama de Seqüência

O diagrama de seqüência mostra como os objetos, que são instâncias das classes do diagrama de classes, se comportam ao longo do tempo e como eles se comunicam entre si. Cada diagrama de seqüência representa um caso de uso descrito anteriormente.

Como a execução do diagrama de seqüências para todos os casos de uso seria muito dispendiosa escolhemos os casos mais importantes para detalhar tanto do módulo de gerenciamento quanto do módulo de divulgação.

3.1.5.1 Locação na Loja (Gestão)

O fluxo mais importante da aplicação trata-se da locação de mídias, que é o fluxo responsável pela principal atividade que ocorre em uma locadora, que são as locações de mídias pelos seus associados. O diagrama de seqüência é mostrado abaixo, junto com a sua descrição detalhada.

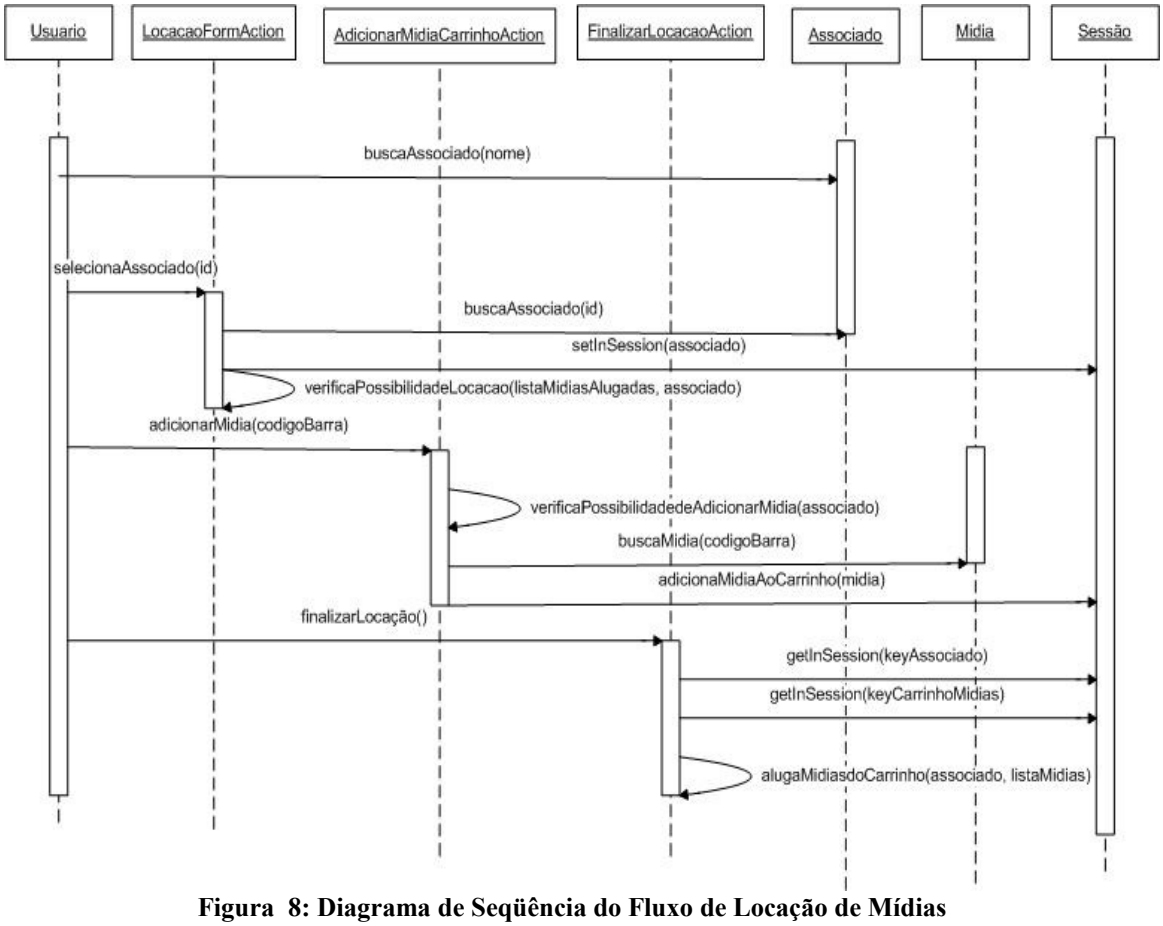

Figura 8: Diagrama de Seqüência do Fluxo de Locação de Mídias

Nesse fluxo o usuário primeiramente busca pelo associado que está na loja e deseja alugar filmes. O sistema retorna ao usuário a lista de associados com aquele nome e o usuário seleciona o associado desejado. Sistema recebe o id desse associado selecionado e busca pelo objeto Associado que possua esse id. Sistema faz a verificação se este associado tem a possibilidade de alugar mídias, verificando através das mídias

que esse associado já possua alugadas. Com a possibilidade de locação, o objeto Associado é colocado no objeto Sessão que o persistirá durante o ciclo de vida do fluxo.

Usuário agora adiciona mídias ao carrinho de locações, passando o código de barra da mídia que o associado deseja alugar. Sistema verifica a possibilidade de se adicionar a mídia tendo como base as mídias que já foram alugadas pelo associado, as mídias que já estão no carrinho de compra e o limite máximo de mídias do plano do associado. Com a possibilidade, sistema busca pela mídia através do código de barra e adiciona essa mídia no carrinho de locação que por vez é colocado também na sessão.

Quando associado deseja finalizar a locação, sistema obtém o associado da sessão e a lista das mídias que o usuário adicionou ao carrinho de locações e então torna cada mídia como locada pelo associado.

3.1.5.2 Locação por Telefone (Gestão)

O diagrama de seqüência do fluxo pela locação de mídias por telefone é mostrado abaixo, junto com a sua descrição detalhada.

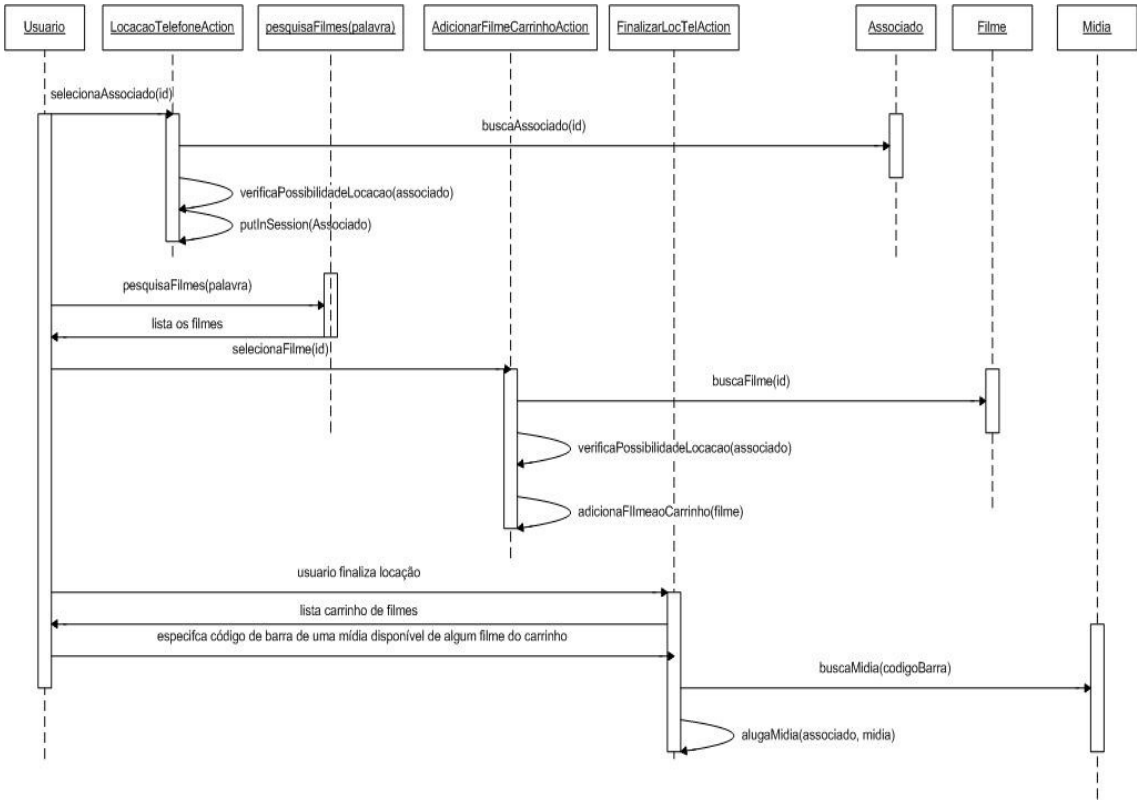

Figura 9: Diagrama de Seqüência do Fluxo de Locação de Mídias

Esse fluxo é responsável pelo aluguel de mídias por telefone, quando o associado não está presente na loja. Para isso o usuário irá primeiramente selecionar o associado que deseja alugar mídias por telefone. O sistema verificará se é possível esse associado continuar o fluxo, baseando-se através da quantidade de mídias que por ventura o associado possui consigo e a quantidade máxima de mídias especificada pelo seu plano mensal. Caso o associado esteja no limite, o sistema seta o objeto referente a esse associado na sessão da aplicação.

Na tela de locação por telefone o usuário agora pesquisa pelo filme que o associado em questão deseja alugar o sistema retorna a lista de filmes da busca. O associado seleciona um filme e o sistema faz a verificação se é possível o associado adicionar este filme ao carrinho de filmes a serem alugados.

O usuário após escolher os filmes do associado finaliza o carrinho de filmes e o sistema envia o usuário para a tela onde deve especificar qual mídia pertencente ao filme do carrinho será locada. Para isso o usuário especifica o código de barra da mídia e submete o formulário e o sistema torna essa mídia como alugada.

3.1.5.3 Retorno de mídia (Gestão)

O retorno de mídia é responsável pela entrega da mídia à loja pelo associado e assim tornando-a disponível para locação novamente. O diagrama de seqüência é mostrado abaixo, junto com a sua descrição detalhada.

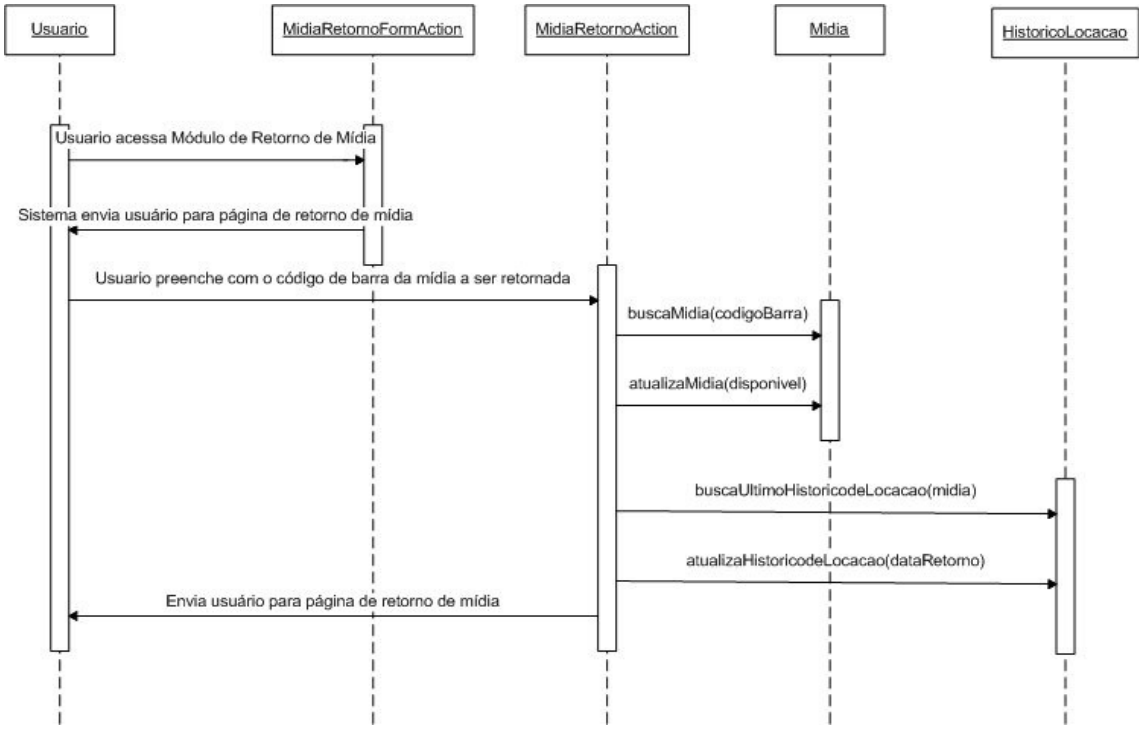

Figura 10: Diagrama de Seqüência do Fluxo de Retorno de Mídias

Nesse fluxo, o usuário primeiramente acessa o módulo de retorno de mídias, sendo enviado a página que contém o formulário de retorno. Então o usuário preenche o formulário com o código de barra da mídia e submete o formulário. Sistema então busca pela mídia correspondente ao código de barra e atualiza a mídia como disponível para locação. Depois sistema atualiza o histórico de locações com a data de retorno da mídia.

3.1.5.4 Adicionar Filmes à Lista de Desejo (Divulgação)

Este diagrama especifica o fluxo que garante ao associado da locadora reservar filmes para serem alugados posteriormente. Essa reserva é realizada através da Lista de desejo. O diagrama abaixo especifica o fluxo e é detalhado adiante.

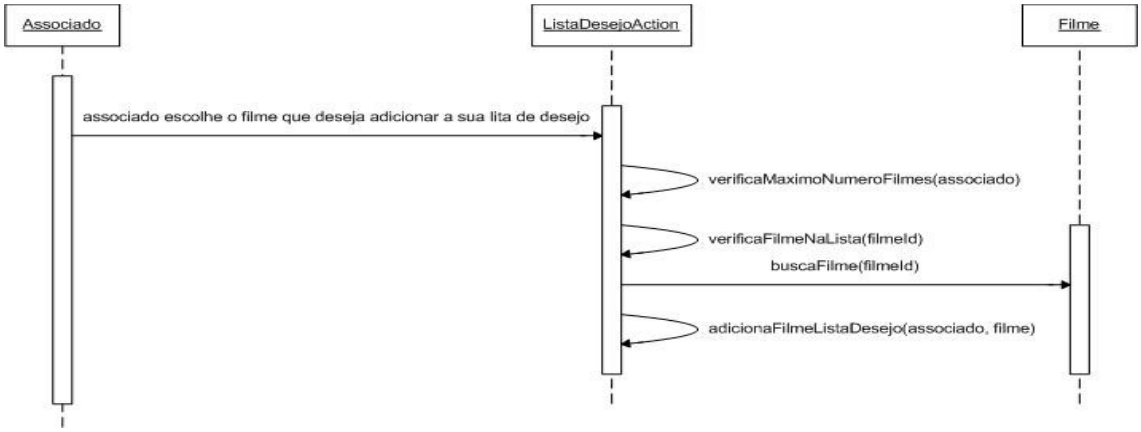

Figura 11: Diagrama de Seqüência do Fluxo de Lista de Desejo

A pré-condição para que este fluxo seja utilizado é que o associado primeiramente se identifique no sistema de divulgação da locadora através do *login*. Após se logar o associado pode adicionar filmes à sua lista de desejo, e para isso deve escolher o filme que deseja através do link relativo.

Após o associado escolher o filme, o sistema verifica se será possível adicionar o filme, verificando o número máximo de filmes permitidos na lista de desejo. Esse número é especificado pelo plano mensal de cada associado. Após essa verificação o sistema verifica também se o filme em questão já está presente na lista de desejo do associado. Caso não esteja o sistema finaliza a operação adicionando o filme à lista de desejo.

3.1.5.5 Avaliar Filmes (Divulgação)

Esse diagrama especifica o fluxo que permite ao associado avaliar os filmes cadastrados na loja, através do módulo de divulgação. O diagrama de seqüência é mostrado abaixo, junto com a sua descrição detalhada.

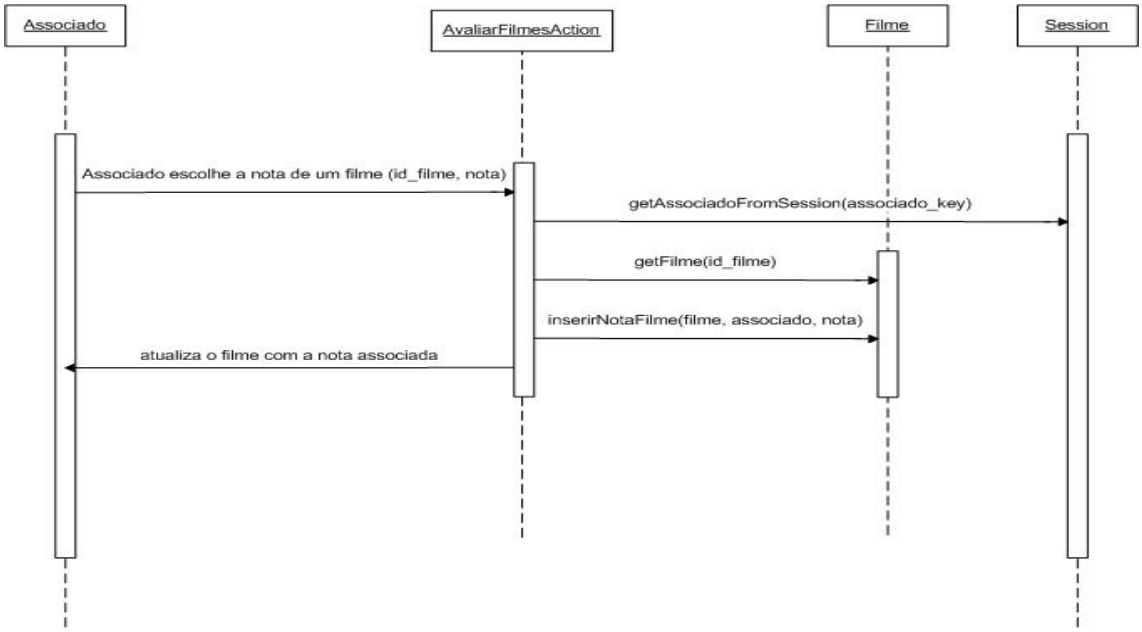

Figura 12: Diagrama de Seqüência do Fluxo de Avaliar Filmes

Primeiramente, o associado que já está logado no sistema, escolhe uma nota para um filme que esteja sendo na tela. Ao escolher uma imagem referente a nota do filme o associado submete ao sistema a informação do filem que está sendo avaliado e a nota

dada. Assim, o sistema busca pela informação do associado logado, que está setada na Sessão do aplicativo. Pelo id do filme sistema busca o objeto referente a esse id e atualiza a nota desse filme passando a informação do associado que avaliou e a nota dada.

3.1.5.6 Pesquisar Filmes (Divulgação)

O diagrama mostrado abaixo especifica o fluxo responsável pela pesquisa de filmes através do módulo de divulgação. Esse diagrama é o principal fluxo do módulo de divulgação, pois permite uma busca completa pelo acervo de filmes cadastrados na loja.

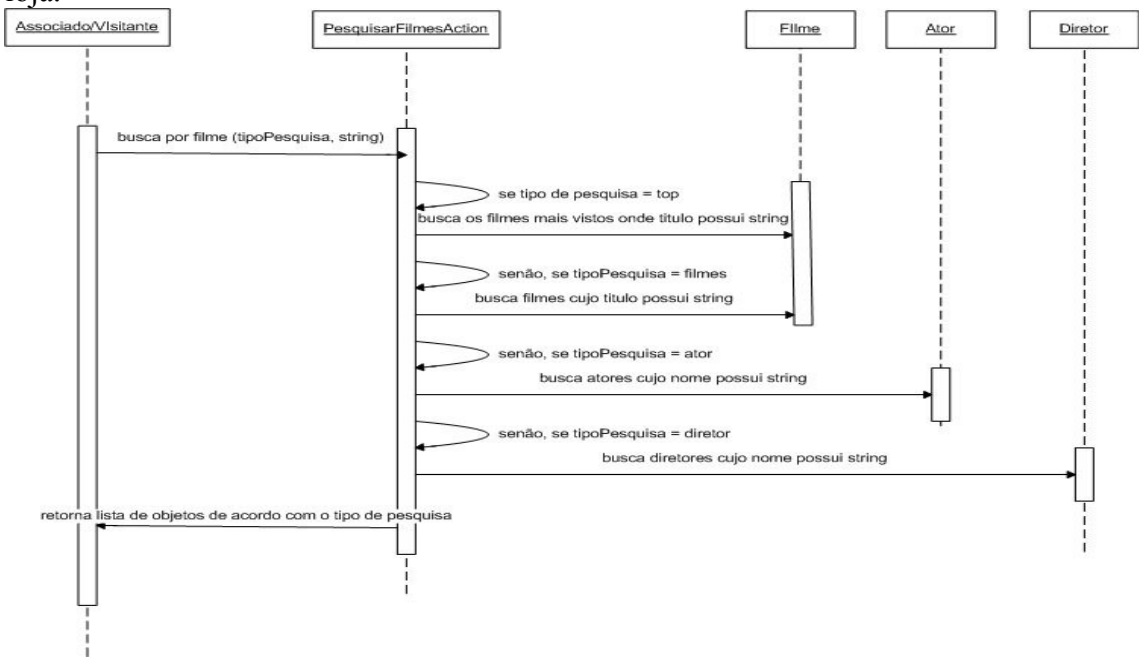

Figura 13: Diagrama de Seqüência do Fluxo de Pesquisa por Filmes

No módulo de divulgação o associado ou visitante pode efetuar uma pesquisa por todo acervo de filmes e informações associadas a filmes tais como atores e diretores. Para efetuar a pesquisa associado deve informar no campo de busca a palavra ou frase que deseja pesquisar. Submetendo essa informação o sistema reconhece o tipo de pesquisa que o associado deseja e a palavra a ser pesquisada. A seguir, sistema verifica o tipo de pesquisa a ser efetuada e obtém a lista de objetos referentes e exibe posteriormente ao associado.

3.2 Interfaces Externas

3.2.1 Interfaces dos Usuários

O Sistema de Gerenciamento de Locadora (SisLoc) foi dividido em dois subsistemas, que visam a complementar as características do produto desenvolvido. Um subsistema será o sistema responsável pela gestão de negócios da locadora, o outro será responsável pela divulgação dos dados inseridos no sistema de gestão e esse subsistema será o site de divulgação do SisLoc.

O subsistema de gestão, que é o sistema responsável pelas informações gerenciais da locadora, é o núcleo do sistema e este será responsável pelo perfeito funcionamento do sistema em geral. Os únicos usuários desse sistema são os funcionários da locadora e estes usuários têm atribuídas funcionalidades de acordo com a responsabilidade de cada um na locadora.

O subsistema de divulgação é o site da locadora e este terá como finalidade divulgar a locadora para o seu público alvo. As funcionalidades desse sistema serão a de divulgar e de informar os filmes que são cadastrados no site. O acesso ao site da locadora será permitido a qualquer pessoa.

Entretanto, os associados da locadora terão acesso a outras funcionalidades especiais do site e para isso terão que se identificar, informando os seus dados de *login* que foram enviados para os e-mails cadastrados no ato da inscrição.

Sistema de Gerenciamento

- **Login**

O sistema de Gestão SisLoc por ser um sistema de administração de locadoras, é fechado para acesso do público, visto a garantir a integridade dos dados da locadora. Para isso foi desenvolvido o módulo de Login que garante que somente pessoas cadastradas na base de dados possam acessar o sistema de Gestão.

A fim de garantir ainda mais segurança à aplicação foram desenvolvidas diversas formas de restringir o acesso indevido ao sistema, tal como controle de sessão do usuário, encriptação de senha utilizando uma API Java de segurança e expiração de sessão.

Para se logar ao sistema, usuário deve fornecer o seu *username* cadastrado e a sua senha. Após submeter o formulário de *login*, o sistema fará as devidas verificações para confirmar a correta identificação do usuário.

Figura 14: Tela de Login do Sistema de Gerenciamento

Após submeter o *login* e este for válido o sistema envia o usuário para a página incial do sistema, que exibe o menu conforme as permissões configuradas para o usuário.

Figura 15: Menu de acesso as funcionalidades do sistema

- **Loja**

A loja é a parte principal do sistema e é a partir dela que o sistema é desenvolvido e as funcionalidades são exibidas para o usuário. Todas as informações que são exibidas para o usuário no sistema, são informações pertencentes à loja que esse usuário é cadastrado. Por isso a primeira linha que deve ser preenchida no banco de dados, após o banco ter sido criado é a linha referente à primeira loja do sistema.

As funcionalidades que envolvem os dados da loja propriamente ditos (tabela 'Loja' do banco de dados) são as funcionalidades de listagem das lojas cadastradas no sistema, inserção de novas lojas no sistema e edição dos dados da loja.

A funcionalidade de remoção de loja não foi implementada visto que a loja como explicado anteriormente, é a parte central de todas as funcionalidades e a exclusão de uma loja por acidente geraria a perda completa do funcionamento do sistema e por isso tal funcionalidade não foi desenvolvida.

- **Inserir Loja**

Essa interface é responsável pela inserção de novas lojas no sistema. Para isso o usuário preenche os campos obrigatórios do relatório e o submete. O sistema faz as devidas validações do formulário e insere a nova loja no sistema. Assim após a inserção dessa nova loja, o sistema poderá ser utilizado com os dados referentes à mesma.

Figura 16: Inserção de nova loja

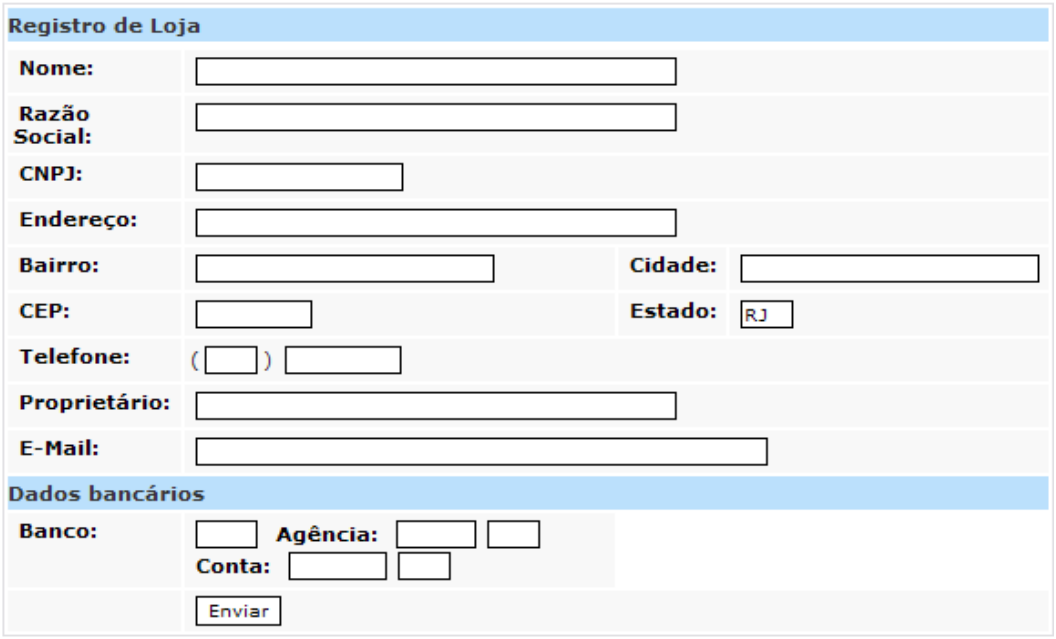

Figura 17: Formulário de dados da nova loja

- **Editar Loja**

Essa interface é responsável pela edição dos dados de uma loja já existente no banco de dados. Para realizar a edição, o usuário deverá primeiramente escolher a loja que deseja alterar através da listagem das lojas e então o sistema exibirá os dados dessa loja e a opção "Alterar".

Após o usuário escolher a opção "Alterar", deve preencher com os novos dados os campos que deseja editar e submeter o formulário. Sistema faz as devidas validações dos campos obrigatórios e altera os dados da loja no banco de dados.

Figura 18: Alteração de dados de uma loja

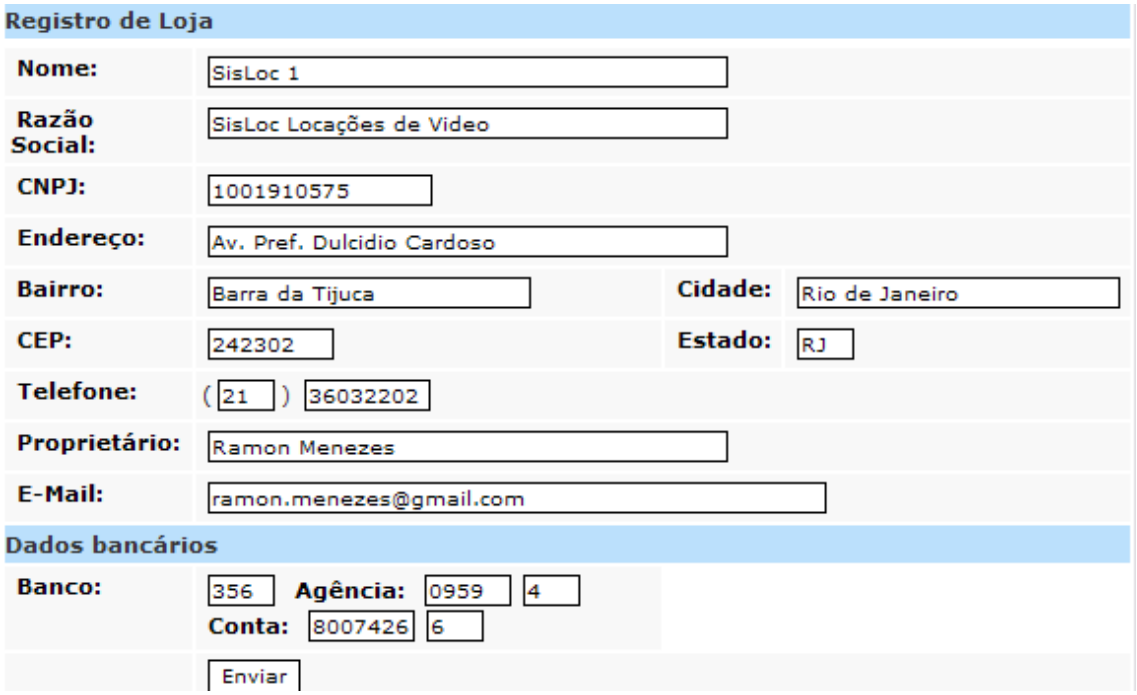

Figura 19: Formulário de alteração dos dados da loja

- **Listar Lojas**

Interface responsável pela listagem de todas as lojas do sistema. A partir desse módulo o usuário poderá verificar a lista de todas as lojas que estão cadastradas no sistema. É a partir desse módulo que o usuário acessará os outros módulos de Inserção de nova loja e de edição de alguma loja.

Figura 20: Listagem de Lojas

- **Usuário**

O usuário é o segundo componente mais importante do sistema, pois são os usuários que serão os responsáveis por inserir dados no sistema. Assim os usuários serão os responsáveis pela integridade dos dados no sistema, pela garantia da correta inserção dos dados no sistema, e assim pelo perfeito funcionamento do sistema.

Para que o sistema seja inicializado na primeira vez será necessário que um usuário seja inserido no banco, através do próprio banco (utilizando do comando *INSERT* da linguagem SQL). Essa inserção se faz necessária visto que se o sistema não possui nenhum usuário cadastrado e então não será possível se logar através do sistema de Gestão.

O usuário como explicado, será o único agente que terá condições de acessar o sistema e assim carregar o sistema com informações. Por ser o único usuário do sistema, foi necessário criar restrições de acessos a estes usuários, visto que nem todos os usuários devem possuir os mesmos níveis de acessos as dados do sistema.

Por se tratar de um sistema de gestão, existem dados que não devem ser exibidos a todos os usuários, visto que alguns dados possuem informações restritas a pessoal da gerência, tal como fluxo de caixa mensal, dados da conta corrente bancária da loja, etc. Assim, para possuir tal funcionalidade foram criadas tabelas que contêm informações das permissões atribuídas a cada usuário do sistema.

As funcionalidades que envolvem os dados dos usuários serão: inserção de novos usuários ao sistema, edição dos dados cadastrais dos usuários, listagem dos usuários cadastrados no sistema e por fim remoção de usuários que por ventura não sejam mais funcionários da loja.

- **Inserir Usuário**

Interface responsável pela inserção de novos usuários no sistema. Para inserir um novo usuário, deve clicar em cima do link "Novo Usuário".

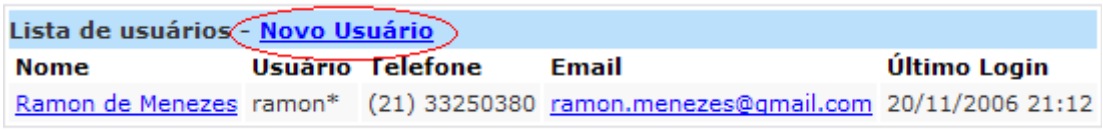

Figura 21: Inserção de novo usuário

Para realizar a inserção, o usuário deverá preencher o formulário, informando os campos obrigatórios de preenchimento e depois submeter o formulário. O sistema então realizará as devidas validações dos campos e criará um novo usuário no banco de dados, conforme os dados preenchidos pelo usuário.

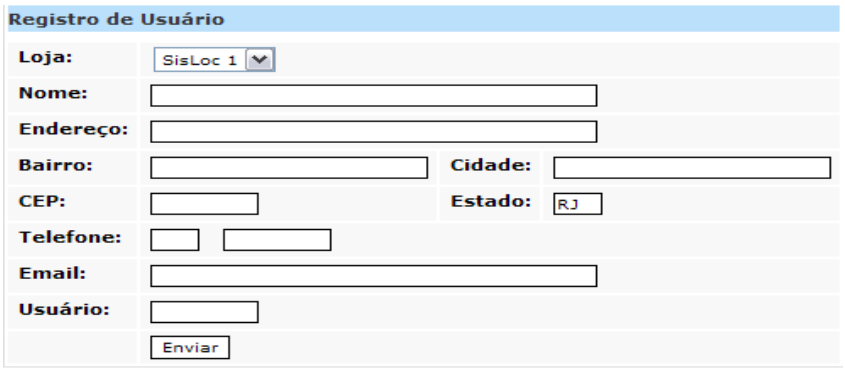

Figura 22: Formulário de dados do novo usuário

- **Editar Usuário**

Interface responsável pela edição dos dados de um usuário já existente no banco de dados. Para realizar a edição, o funcionário deverá primeiramente escolher o usuário que deseja alterar através da listagem dos usuários e então o sistema exibirá os dados desse usuário e a opção "Alterar".

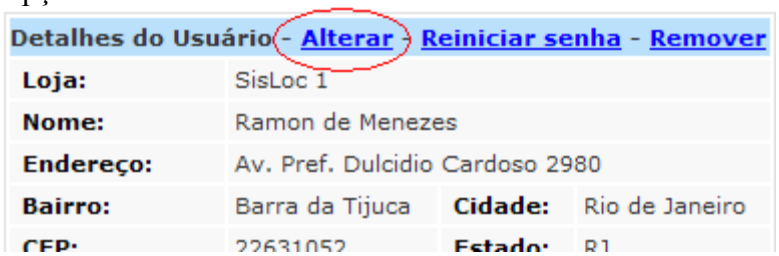

Figura 23: Alteração de dados do usuário

Após o usuário escolher a opção "Alterar", deve preencher com os novos dados os campos que deseja editar e submeter o formulário. Sistema faz as devidas validações dos campos obrigatórios e altera os dados do usuário no banco de dados.

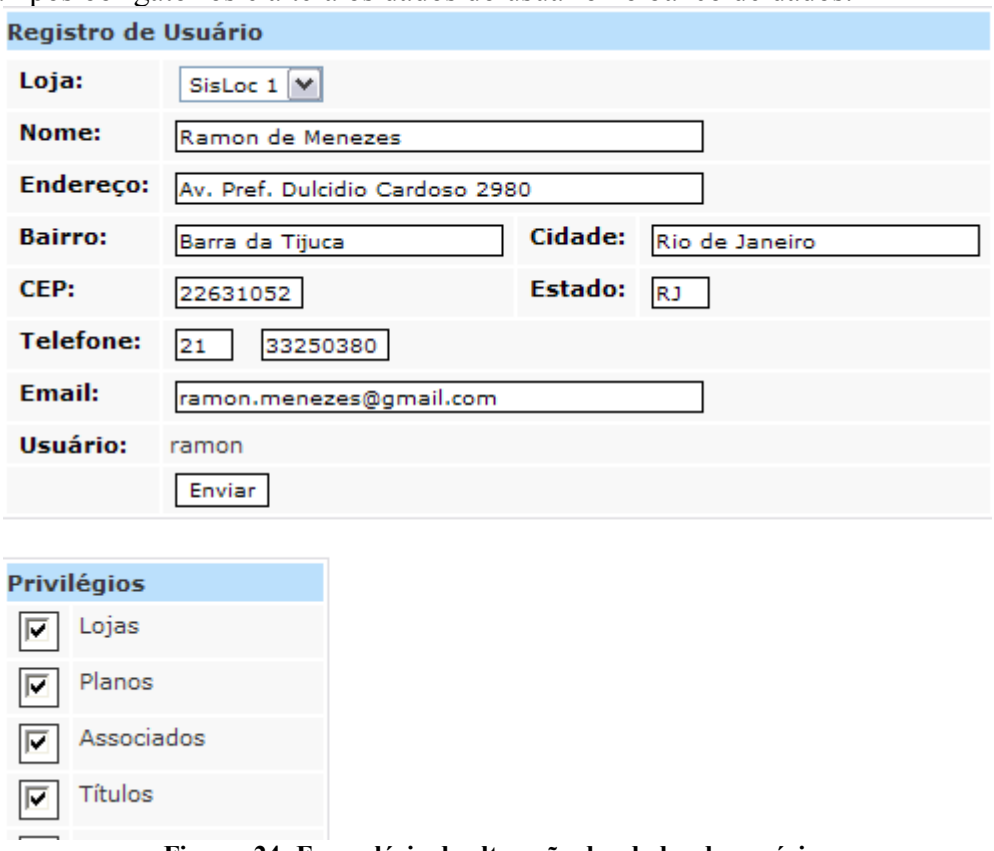

Figura 24: Formulário de alteração dos dados do usuário

- **Listar Usuários**

Interface responsável pela listagem de todos os usuários do sistema. A partir desse módulo o usuário poderá verificar a lista de todos os usuários que estão cadastradas no sistema. É a partir desse módulo que o usuário acessará os outros módulos de inserção de novo usuário e de edição de algum usuário existente.

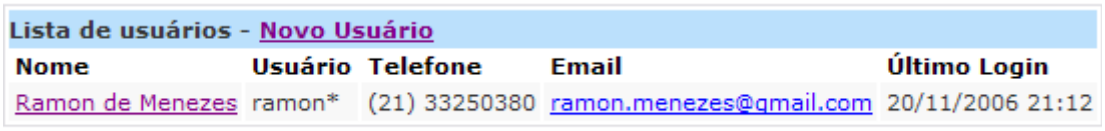

Figura 25: Listagem de todos os usuários

- **Remover Usuário**

Para realizar a remoção, o funcionário deverá primeiramente escolher o usuário que deseja remover através da listagem dos usuários e então o sistema exibirá os dados desse usuário e a opção "Remover".

Após o usuário escolher a opção "Remover", o sistema removerá o usuário do sistema bem como suas dependências em outras tabelas.

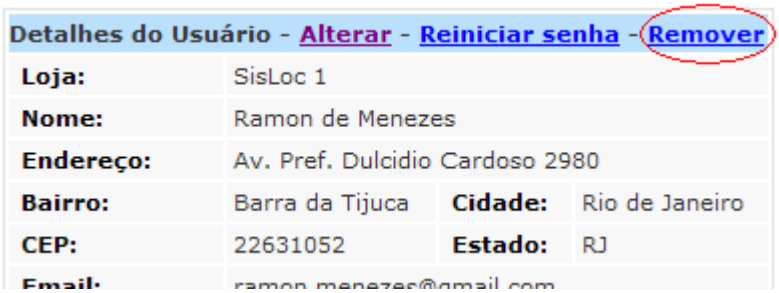

Figura 26: Remover Usuário

- **Plano Mensal**

O plano é a funcionalidade responsável pelo correto funcionamento dos principais fluxos do sistema de Gestão. Esses fluxos são os de locações de mídia na loja e por telefone.

Será através dos planos cadastrados no sistema que é estabelecida a quantidade de mídias que cada associado pode possuir consigo e também o valor da mensalidade de cada usuário. Outras atribuições foram colocadas também nos planos, tais como quantidade máxima de dependentes e quantidade máxima de mídias na lista de desejo.

Por ser responsável pelo perfeito funcionamento do fluxo de locações, o cadastro de algum plano deve ser uma das primeiras inserções que devem ser realizadas pelo usuário na primeira vez que o sistema for iniciado.

Para gerenciar esses planos foram desenvolvidas quatro funcionalidades que são: inserir, editar, listar e remover plano.

- **Inserir Plano**

Para realizar a inserção, o usuário deverá primeiramente clicar no link "Novo Plano" e então o sistema enviará o usuário para a tela de inserção de novo plano.

Figura 27: Novo Plano Mensal

O usuário deve preencher então o formulário, informando os campos obrigatórios de preenchimento e depois submeter o formulário. O sistema então realizará as devidas

validações dos campos e criará um novo plano no banco de dados, conforme os dados preenchidos pelo usuário.

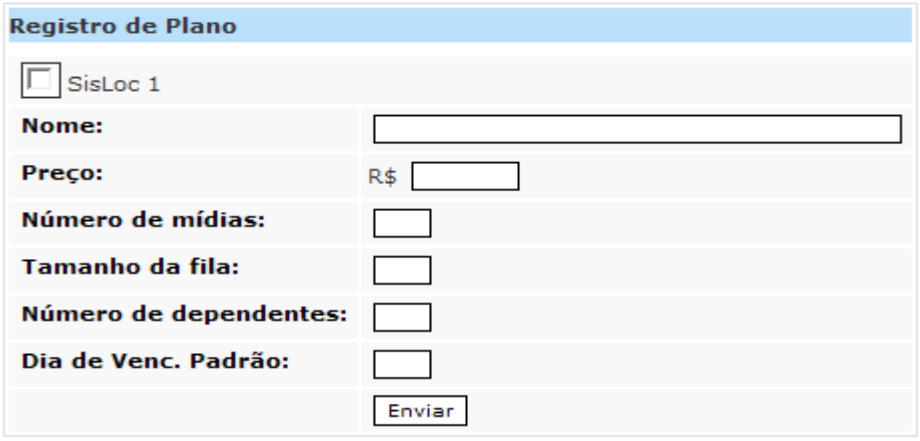

Figura 28: Formulário de dados do Plano Mensal

- **Editar Planos**

Para realizar a edição, o funcionário deverá primeiramente escolher o plano que deseja alterar através da listagem dos planos e então o sistema exibirá os dados desse plano e a opção "Alterar".

Figura 29: Alterar Plano Mensal

Após o usuário escolher a opção "Alterar", deve preencher com os novos dados os campos que deseja editar e submeter o formulário. Sistema faz as devidas validações dos campos obrigatórios e altera os dados do plano no banco de dados.

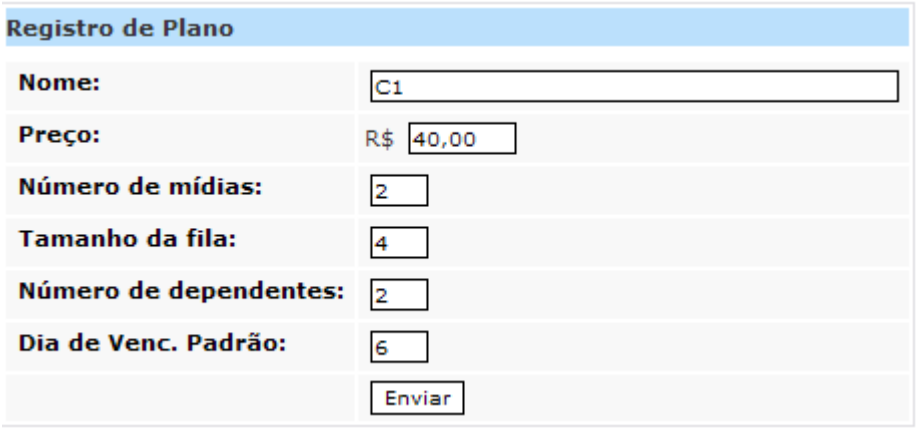

Figura 30: Formulário de Alteração de Plano Mensal

- **Listar Planos**

A partir desse módulo o usuário poderá verificar a lista de todos os planos que estão cadastradas no sistema. É a partir desse módulo que o usuário acessará os outros módulos de inserção de novo plano e de edição de algum plano já existente.

Figura 31: Listagem de todos os Planos Mensais

- **Remover Plano**

Para realizar a remoção, o funcionário deverá primeiramente escolher o plano que deseja remover através da listagem dos planos e então o sistema exibirá os dados desse plano e a opção "Excluir".

Figura 32: Excluir Plano Mensal

Após o usuário escolher a opção "Excluir", o sistema removerá o plano do sistema bem como suas dependências em outras tabelas. Caso algum associado possua este plano como plano mensal o sistema não realizará a transação, e emitirá uma mensagem de erro. Esse fluxo foi desenvolvido, pois no sistema não podem existir nenhum associado sem plano mensal, já que acarretaria o incorreto funcionamento do sistema.

- **Associado**

O associado é o agente mais importante do sistema, pois é o associado que move todo o modelo de negócio do sistema de Gestão. O associado é o sócio da locadora e é o agente que garantirá a agregação de valor ao novo sistema desenvolvido.

Por ter grande importância no sistema e por ser o sistema um modelo de negócio que se baseia na confiança mútua entre cliente e fornecedor, os dados do cliente devem conter as mais fidedignas informações. Entretanto para garantir as validades dessas informações foram criados no sistema algoritmos e funcionalidades que validam e guardam tais informações fornecidas pelo associado.

Para o gerenciamento das informações dos associados através do sistema foram criadas várias funcionalidades que são: inserir, editar, pesquisar e desativar associado.

- **Inserir Associado**

Para realizar a inserção, o usuário deverá primeiramente clicar no link “Novo Associado” e então o sistema enviará o usuário para a tela de inserção de novo associado.

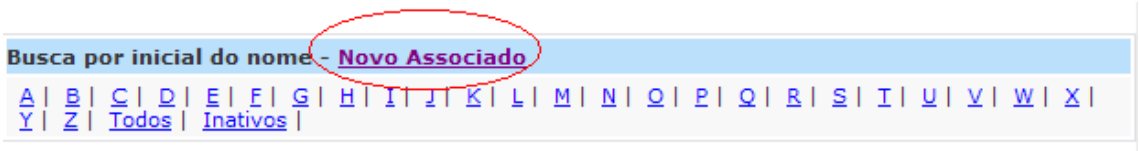

Figura 33: Inserir novo Associado

Para realizar a inserção, o usuário deverá preencher o formulário, informando os campos obrigatórios de preenchimento e depois submeter o formulário. O sistema então realizará as devidas validações dos campos e criará um novo associado no banco de dados, conforme os dados preenchidos pelo usuário.

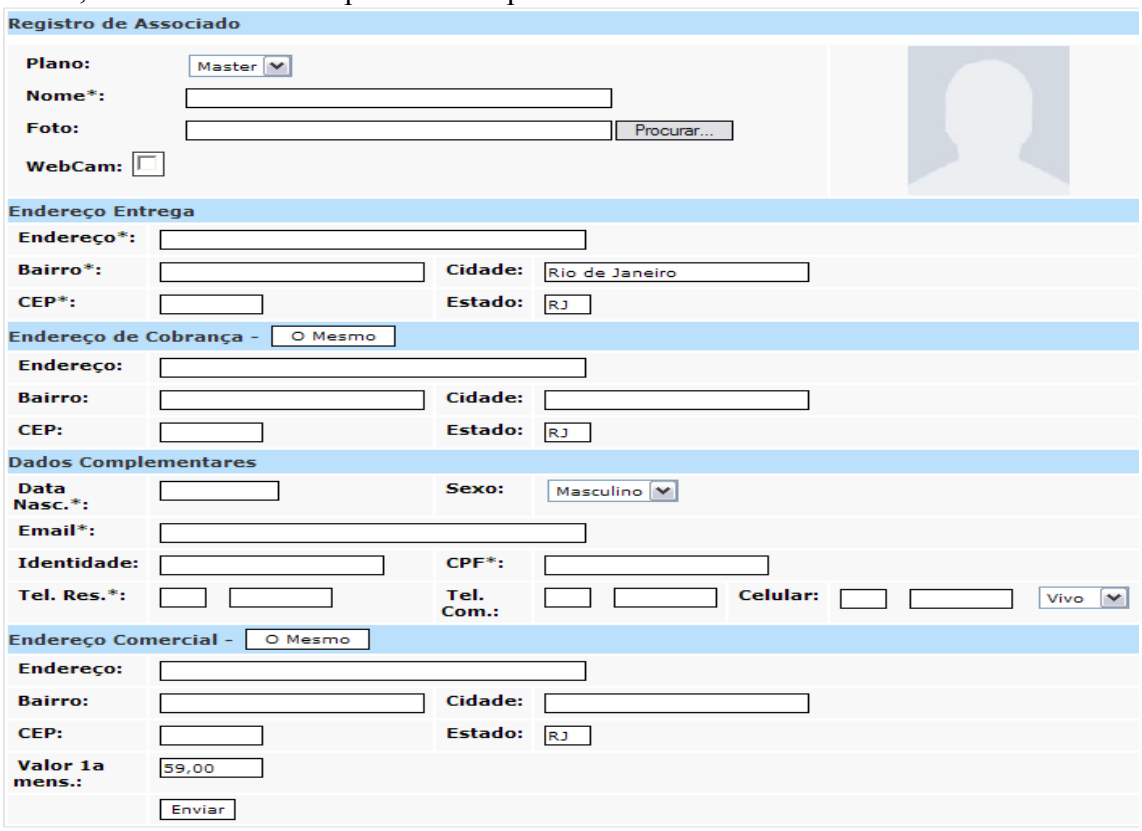

Figura 34: Formulário de dados do novo associado

Como explicado anteriormente os dados do associado titular serão validados para garantir a veracidade da informação fornecida. Para isso foi desenvolvido no sistema duas formas de garantias que são o algoritmo de verificação de número de CPF e também a possibilidade de registro da imagem do associado que pode ser feito através de *Webcam* e fornecido a imagem ao sistema pelo formulário de cadastro e/ou edição de associado.

Essas duas formas certamente provêem uma maior segurança ao sistema em um quesito que geralmente nenhum sistema possui garantia, que é na veracidade das informações fornecidas pelas pessoas.

- **Editar Associado**

Para realizar a edição, o funcionário deverá primeiramente escolher o associado que deseja alterar os dados através das diversas formas de pesquisas de associado que o sistema fornece.

Após o usuário escolher o associado que deseja alterar e clicar no link "Editar" na página de detalhes do associado, o sistema envia o usuário para a página de edição dos dados cadastrais do associado.

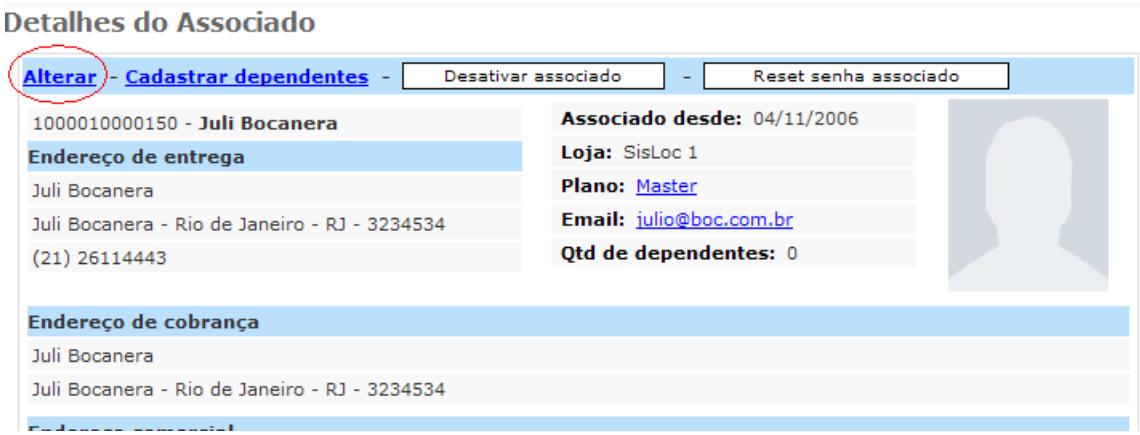

Figura 35: Alteração dos dados do Associado

O usuário deve então preencher com os novos dados os campos que deseja editar e submeter o formulário. O sistema fará as devidas validações dos campos obrigatórios e altera os dados do associado no banco de dados.

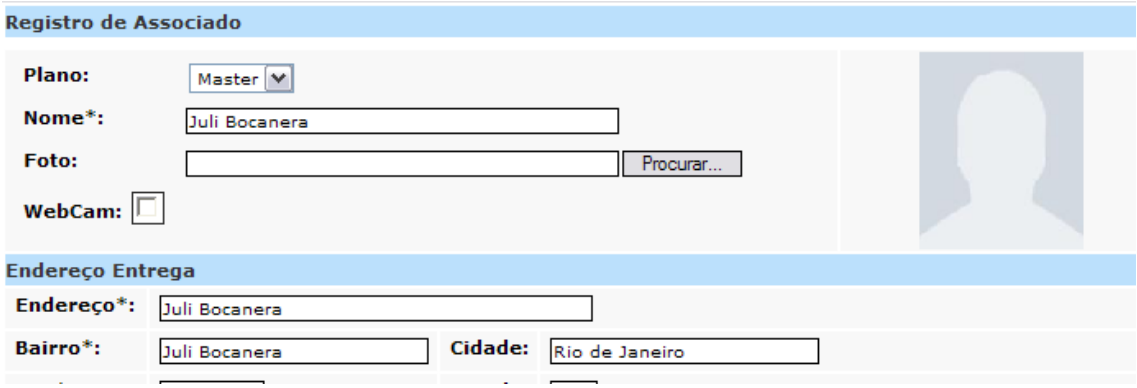

Figura 36: Formulário de Alteração dos dados do associado

- **Pesquisar Associado**

Por ser um sistema voltado para as ações que associado deseja fazer, foram desenvolvidas formas de pesquisa que garantam uma maior rapidez na busca por informações relativas aos associados.

Por ser um sistema de uma locadora, o fluxo de pedidos dos clientes variam nas horas do dia, entretanto o sistema deve garantir que mesmo em horário de maior pico de clientes não haja grandes diferenças no tempo de resposta. Visando a garantir tal necessidade, que foi desenvolvida três formas de pesquisa por associado que são: pesquisa por Código de Barra, pesquisa por Nome e pesquisa por Telefone.

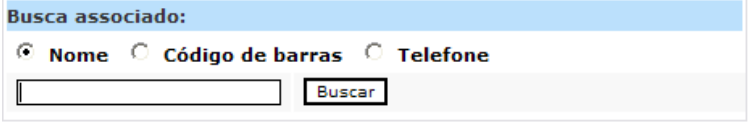

Figura 37: Pesquisa de Associados

Para efetuar alguma pesquisa, usuário seleciona o tipo de pesquisa que deseja realizar e fornece o campo a ser avaliado. Após submeter o formulário de pesquisa o sistema retorna com os dados referentes a pesquisa de associado.

- **Desativar Associado**

Esta interface é responsável pela desativação da conta de algum associado e de seus dependentes. Este módulo foi desenvolvido visando às especificações do modelo de negócio proposto.

Por ser um sistema baseado em planos mensais, tal modelo de negócio implica que todo mês todos os associados pagarão uma mensalidade referente ao seu plano. Então foi desenvolvida no sistema uma forma de desativar a conta de um cliente a fim de que esse associado não seja cobrado ao final do mês.

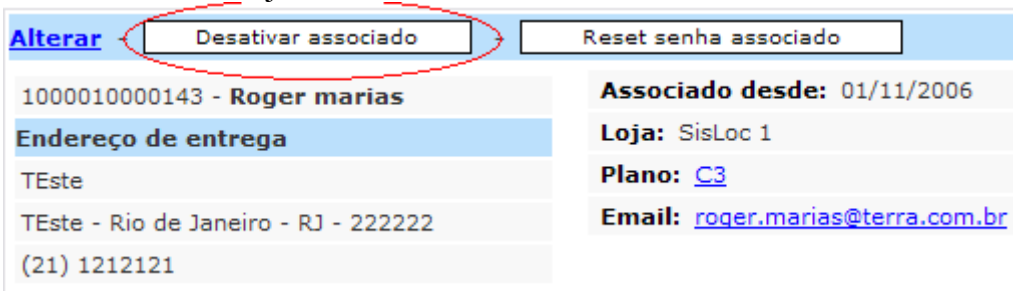

Figura 38: Desativar Associado

Para desativar um associado, o usuário deve selecioná-lo através da pesquisa por associado, para que sejam exibidos os detalhes do associado na tela. Na tela de detalhes do associado, o usuário deve clicar no botão "Desativar". Caso o associado em questão seja um associado titular, todos os seus dependentes terão também as suas contas desativadas.

- **Filmes**

Esta interface é responsável pelo gerenciamento dos filmes cadastrados na base de dados do sistema. Os filmes contêm informações a respeito dos títulos que são cadastrados no sistema e são essas informações que serão exibidas no subsistema de divulgação da locadora.

É a partir dos filmes cadastrados na base de dados que as mídias são adicionadas e assim podem ser alugadas pelos associados. Portanto serão os filmes que conterão as informações necessárias para pesquisa dos usuários através do sistema de Gestão e dos associados através do sistema de divulgação.

Para este módulo foram desenvolvidas cinco funcionalidades que garantirão gerenciamento dos filmes e das mídias através do sistema de Gestão. Essas funcionalidades são: inserir, editar, remover e pesquisar filmes e adicionar, editar e remover mídias.

- **Inserir Filmes**

Para realizar a inserção, o usuário deverá primeiramente clicar no link "Novo Associado" e então o sistema enviará o usuário para a tela de inserção de novo associado.

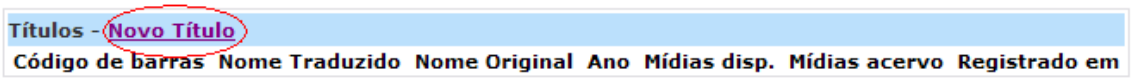

Figura 39: Novo Filme

Para realizar a inserção, o usuário deverá preencher o formulário, informando os campos obrigatórios de preenchimento e depois submeter o formulário. O sistema então realizará as devidas validações dos campos e criará um novo filme no banco de dados, conforme os dados preenchidos pelo usuário.

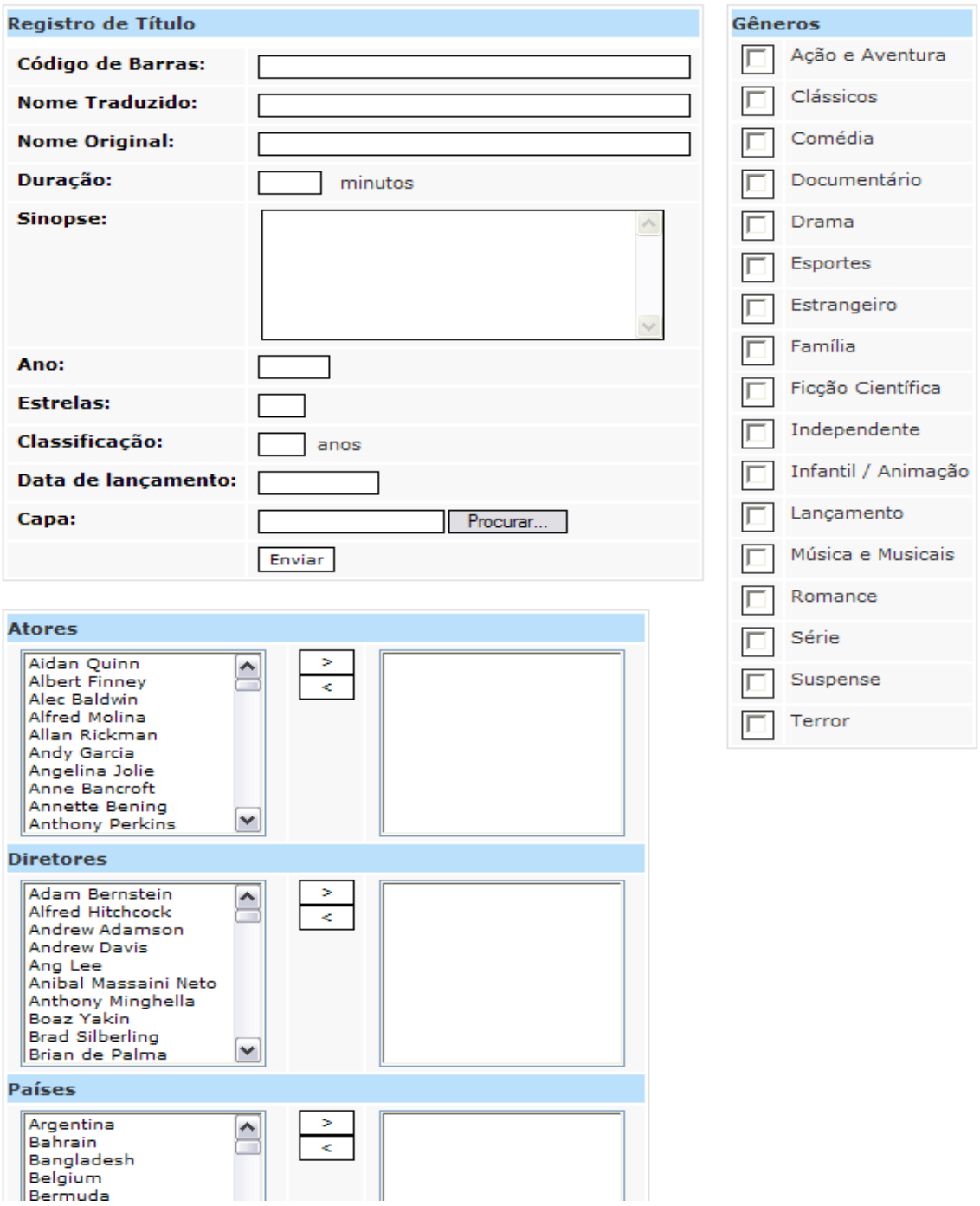

Figura 40: Formulário de dados do novo filme

- **Editar Filmes**

Para realizar a edição, o funcionário deverá primeiramente escolher o filme que deseja alterar os dados através das diversas formas de pesquisas de filme que o sistema fornece e depois de escolher o filme, clicar no link “Alterar”.

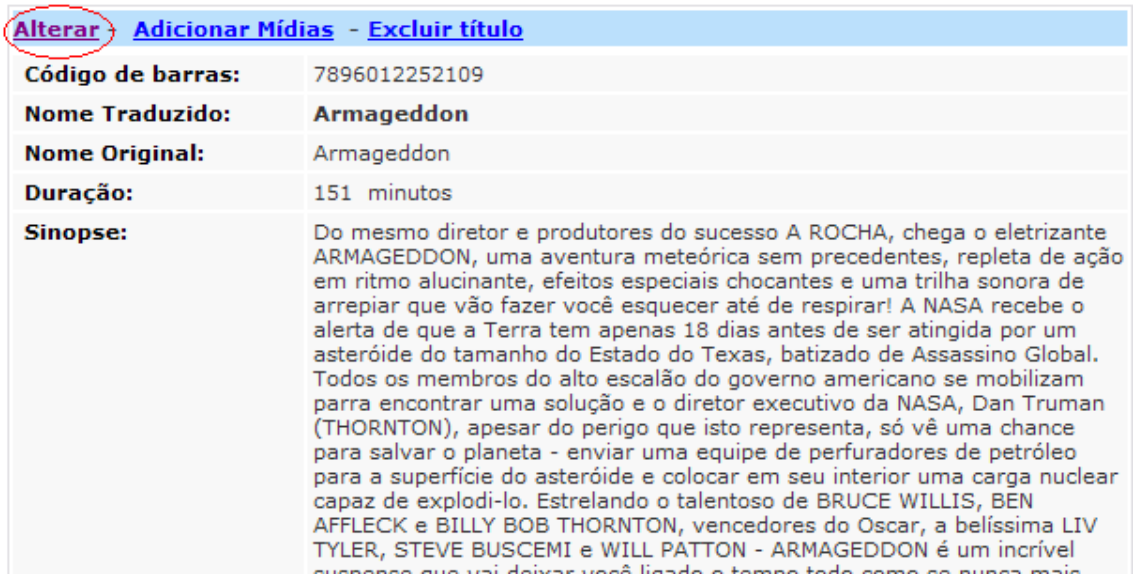

Figura 41: Alteração de Filme

Após o usuário escolher o filme que deseja alterar, o sistema envia o usuário para a página de edição dos dados cadastrais do filme. O usuário deve então preencher com os novos dados os campos que deseja editar e submeter o formulário. O sistema fará as devidas validações dos campos obrigatórios e altera os dados do filme no banco de dados.

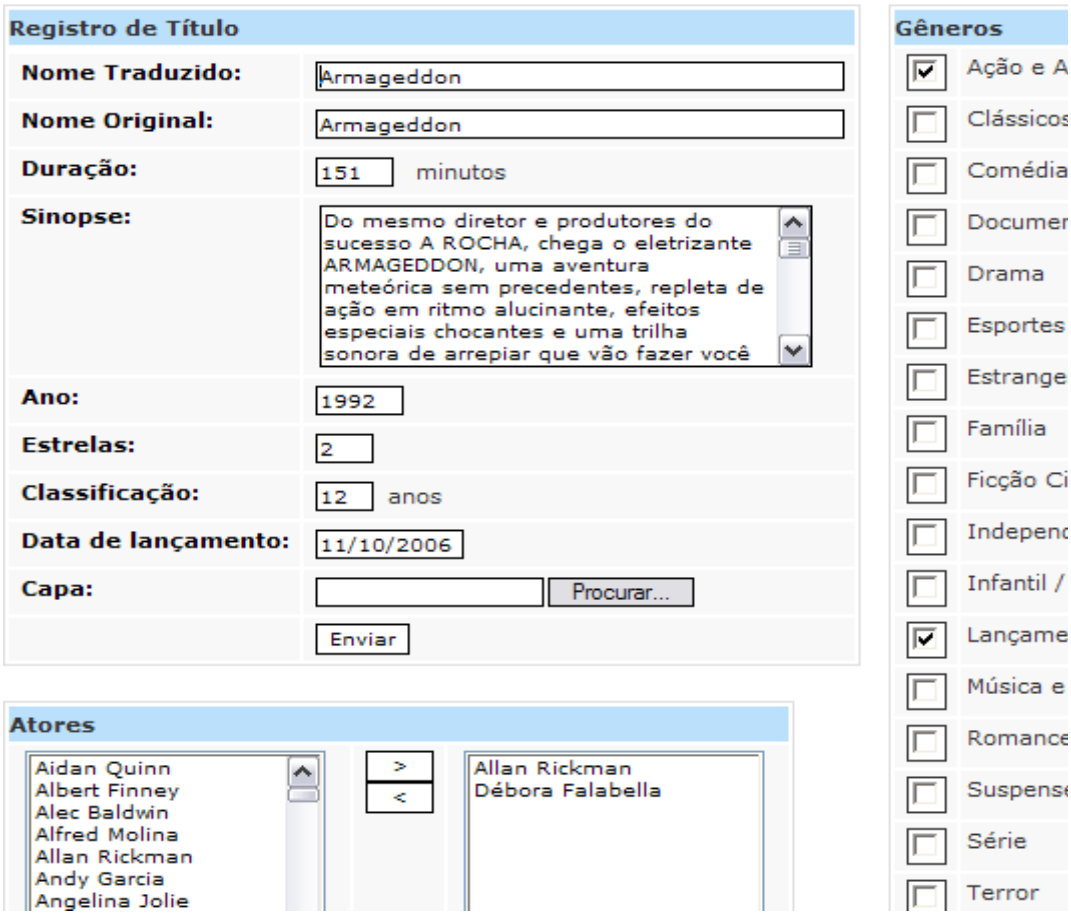

Figura 42: Formulário de Alteração de Filme

- **Remover Filme**

Interface responsável pela remoção dos dados de um filme, bem como suas referências e dependências da base de dados do sistema. Para realizar a remoção de um filme, o usuário deve primeiramente pesquisar pelo filme através das diversas formas de pesquisa que existem, e depois após encontrar o filme, selecioná-lo.

Após selecioná-lo o sistema exibe a tela com as informações do filme e a opção "Excluir" para remover o filme. Usuário clica no link "Excluir" para remover o filme do sistema.

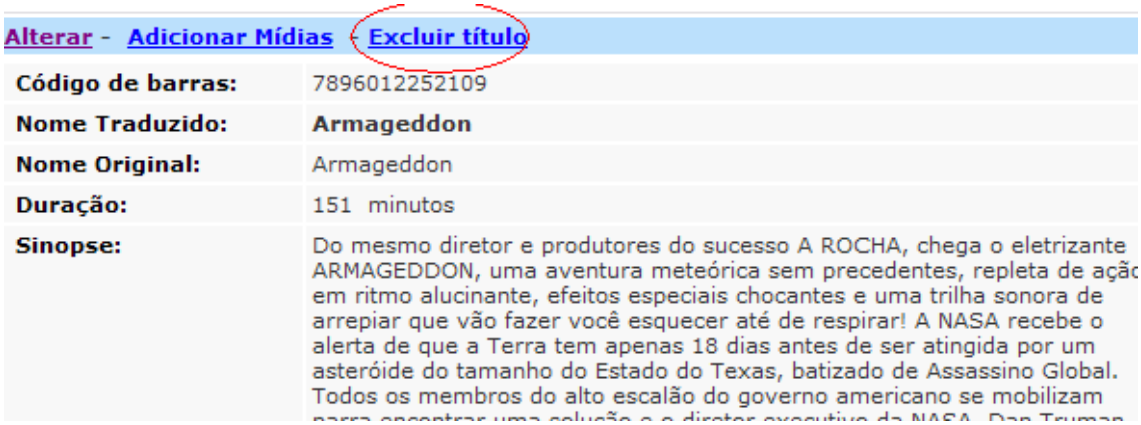

Figura 43: Excluir Filme

- **Pesquisar Filmes**

Interface responsável pelas formas de pesquisa por filme que o sistema fornece. No sistema foram desenvolvidas duas formas de pesquisa por filme: pesquisa por palavra no título e pesquisa por Código de Barra do filme cadastrado.

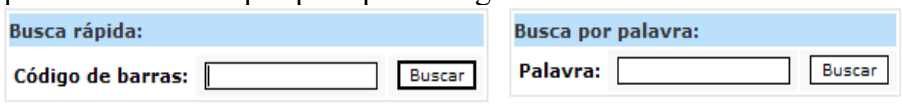

Figura 44: Pesquisar Filmes

Para efetuar alguma pesquisa, usuário seleciona o tipo de pesquisa que deseja realizar e fornece o campo a ser avaliado. Após submeter o formulário de pesquisa o sistema retorna com os dados referentes à pesquisa de filmes.

- **Adicionar Mídias**

Interface responsável por inserir uma nova mídia na base de dados do sistema. Uma mídia que é cadastrada está associada a algum filme já inserido. As mídias serão os objetos responsáveis pelas ações dos associados no sistema, ou seja, pelas locações que ocorrem na loja.

Para um usuário inserir alguma mídia, este deve primeiramente selecionar um filme a que essa mídia irá ser adicionada. Após selecionar algum filme o usuário pode clicar no link "Adicionar Mídias".

Figura 45: Adicionar Mídias

O sistema então enviará o usuário para a tela de cadastro de mídias. O usuário então preenche e submete o formulário, então o sistema criará uma nova mídia no banco de dados, conforme os dados preenchidos pelo usuário.

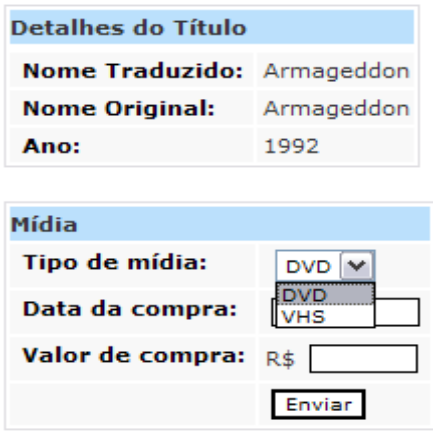

Figura 46: Formulário de nova Mídia

- **Editar Mídias**

Interface responsável pela edição dos dados de uma mídia já existente no banco de dados. Para realizar a edição, o funcionário deverá primeiramente escolher a mídia que deseja alterar os dados através da seleção do filme a que essa mídia pertence.

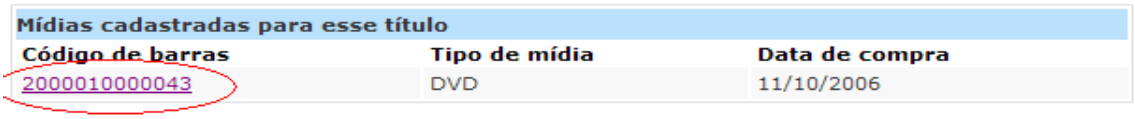

Figura 47: Seleção da mídia

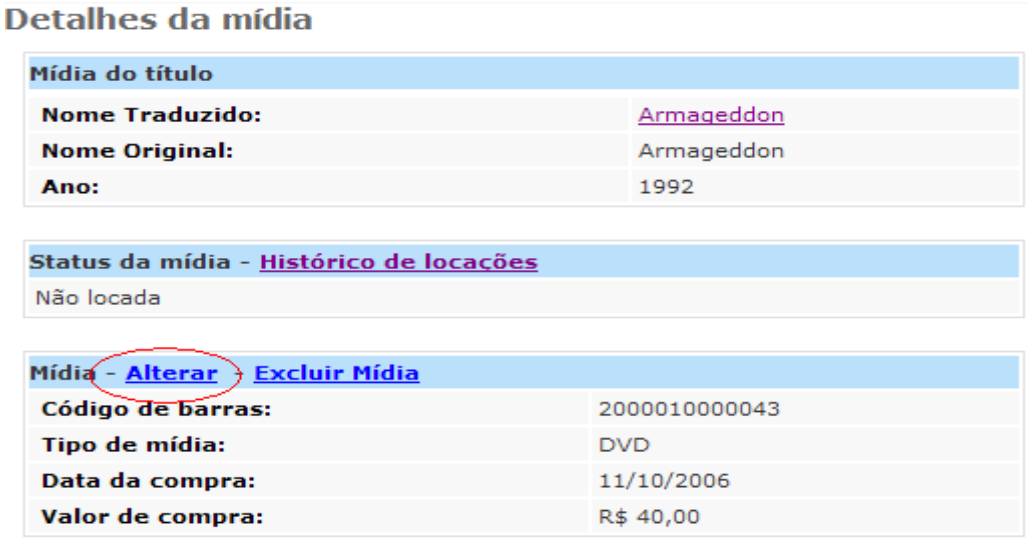

Figura 48: Alterar dados da Mídia

Após o usuário escolher a mídia que deseja alterar, o sistema envia o usuário para a página de edição dos dados cadastrais da mídia. O usuário deve então preencher com os novos dados os campos que deseja editar e submeter o formulário. O sistema fará as devidas validações dos campos obrigatórios e altera os dados da mídia no banco de dados.

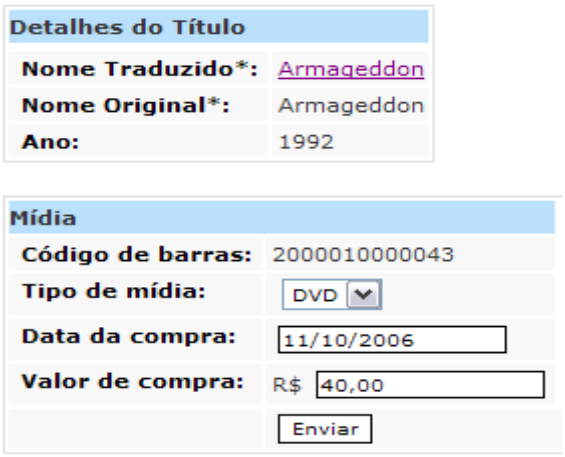

Figura 49: Formulário de Alteração de Mídia

- **Remover Mídias**

Para realizar a remoção de uma mídia, o funcionário deverá primeiramente escolher a mídia que deseja remover através da seleção do filme a que essa mídia pertence.

Após selecioná-la o sistema exibe a tela com as informações da mídia e a opção "Excluir" para removê-la. Usuário clica no link "Excluir" para remover a mídia do sistema.

Figura 50: Excluir Mídia

- **Ator**

O ator é utilizado no sistema como informação adicional aos filmes inseridos e são utilizados para fins informativos e de pesquisa de filmes tanto no sistema de Gestão quanto no sistema de divulgação da locadora.

Para o gerenciamento dos atores no sistema foram criadas quatro funcionalidades que são: inserir, editar, remover e pesquisar atores.

- **Inserir Ator**

Para realizar a inserção, o usuário deverá preencher o formulário, informando os campos obrigatórios de preenchimento e depois submeter o formulário.

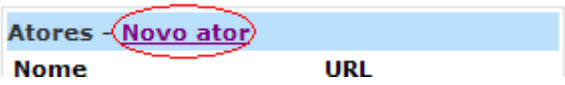

Figura 51: Inserir novo Ator

O sistema então realizará as devidas validações dos campos e criará um novo ator no banco de dados, conforme os dados preenchidos pelo usuário.

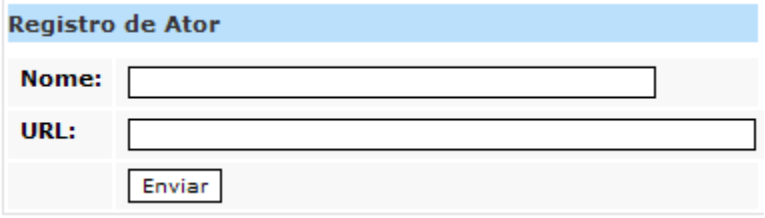

Figura 52: Formulário de inserção de novo ator

- **Editar Ator**

Para realizar a edição, o funcionário deverá primeiramente escolher o ator que deseja alterar através da listagem dos atores e então o sistema exibirá os dados desse ator e a opção "Alterar".

Figura 53: Alterar Ator

Após o usuário escolher a opção "Alterar", deve preencher com os novos dados os campos que deseja editar e submeter o formulário. Sistema faz as devidas validações dos campos obrigatórios e altera os dados do ator no banco de dados.

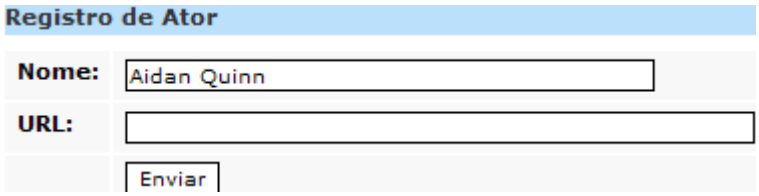

Figura 54: Formulário de alteração de ator

- **Listar Atores**

A partir dessa interface o usuário poderá verificar a lista de todos os atores que estão cadastrados no sistema. É a partir desse módulo que o usuário acessará os outros módulos de inserção de novo ator e de edição de algum ator já existente.

Figura 55: Listagem de todos os atores

- **Remover Ator**

Para realizar a remoção, o funcionário deverá primeiramente escolher o ator que deseja remover através da listagem dos atores e então o sistema exibirá os dados desse ator e a opção "Remover".

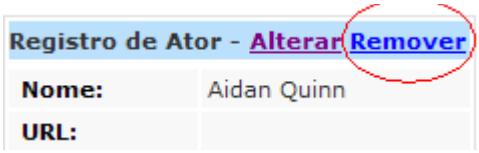

Figura 56: Remover Ator

Após o usuário escolher a opção "Remover", o sistema removerá o ator do sistema bem como suas dependências em outras tabelas.

- **Diretor**

O diretor é utilizado no sistema como informação adicional aos filmes inseridos e são utilizados para fins informativos e de pesquisa de filmes tanto no sistema de Gestão quanto no sistema de divulgação da locadora.

Para o gerenciamento dos diretores no sistema foram criadas quatro funcionalidades que são: inserir, editar, remover e pesquisar diretores.

- **Inserir Diretor**

Para realizar a inserção, o usuário deverá preencher o formulário, informando os campos obrigatórios de preenchimento e depois submeter o formulário.

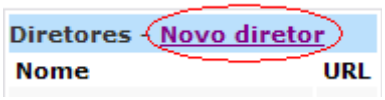

Figura 57: Inserir Novo Diretor

O sistema então realizará as devidas validações dos campos e criará um novo diretor no banco de dados, conforme os dados preenchidos pelo usuário.

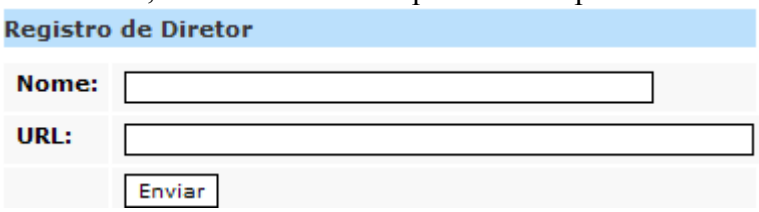

Figura 58: Formulário de inserção de novo diretor

- **Editar Diretor**

Para realizar a edição, o funcionário deverá primeiramente escolher o diretor que deseja alterar através da listagem dos diretores e então o sistema exibirá os dados desse diretor e a opção "Alterar".

Figura 59: Alterar Diretor

Após o usuário escolher a opção "Alterar", deve preencher com os novos dados os campos que deseja editar e submeter o formulário. Sistema faz as devidas validações dos campos obrigatórios e altera os dados do diretor no banco de dados.

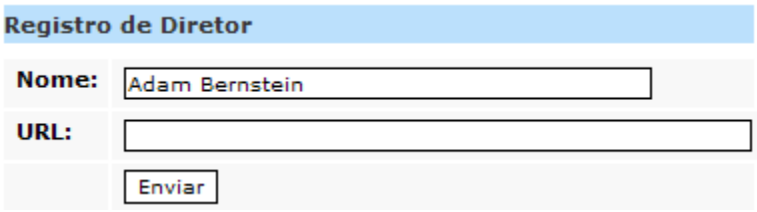

Figura 60: Formulário de alteração de diretor

- **Listar Diretores**

A partir desse módulo o usuário poderá verificar a lista de todos os diretores que estão cadastrados no sistema. É a partir desse módulo que o usuário acessará os outros módulos de inserção de novo diretor e de edição de algum diretor já existente.

Figura 61: Listagem de todos os diretores

- **Remover Diretor**

Para realizar a remoção, o funcionário deverá primeiramente escolher o diretor que deseja remover através da listagem dos diretores e então o sistema exibirá os dados desse diretor e a opção "Remover".

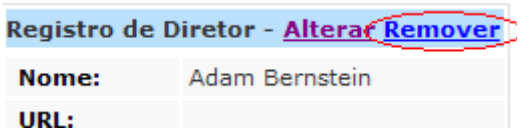

Figura 62: Remover Diretor

Após o usuário escolher a opção "Remover", o sistema removerá o diretor do sistema bem como suas dependências em outras tabelas.

- **Locação de mídias na Loja**

Esta interface é responsável pelo fluxo de locação de mídias na loja, ou seja, esse fluxo é utilizado quando o associado está presente na loja e este está querendo alugar certa quantidade de mídias.

Assim, sempre que um associado está presente na loja e irá alugar alguma mídia esse módulo deve ser utilizado. De acordo com esse fluxo o sistema fará a verificação da quantidade de mídias que o associado deseja locar e confirmará se esse associado poderá alugar essas mídias.

Esse fluxo analisará a quantidade de mídias do plano mensal, a quantidade de mídias que deseja alugar e também a quantidade de mídias que o associado possui em casa, que por ventura tenha anteriormente alugado.

Para utilizar esse fluxo, o usuário deve adicionar as mídias que o associado deseja levar, ao carrinho de mídias. Conforme for adicionando o sistema faz a verificação se é possível adicionar mais mídias ao carrinho de locação. Essa verificação leva em conta as mídias que o associado já possui alugadas mais as mídias que está tentando alugar e verifica se é menor ou igual ao limite do seu plano mensal.

Após todas terem sido adicionadas o usuário pode finalizar a locação. Após o usuário finalizar o sistema fará as devidas verificações da quantidade de mídias permitidas e adicionará essas mídias como mídias alugadas pelo associado.

Figura 63: Locação na Loja

- **Locação por Telefone**

Essa interface é responsável pelo fluxo de locação de mídias por telefone, ou seja, esse fluxo é utilizado quando o associado não está presente na loja e este está querendo alugar certa quantidade de mídias que são especificadas pelo telefone.

Assim, sempre que um associado não está presente na loja e liga para loja querendo alugar alguma mídia esse módulo deve ser utilizado. A principal diferença entre o fluxo de locação na loja e o de locação por telefone é que o fluxo de locação por telefone faz uma prévia verificação das quantidades de mídias que o associado possui em seu poder, ou seja, mídias que o associado já tinha alugado anteriormente e ainda não foram retornadas.

Com essa verificação o sistema permite que o associado possa alugar uma quantidade de mídias maior do que o seu plano, pois o sistema foi desenvolvido de forma que as mídias que estão na casa do associado serão retornadas quando algum funcionário da loja for entregar as mídias que o associado acabou de alugar por telefone.

Para utilizar esse fluxo, o usuário primeiramente adiciona os filmes que o associado deseja alugar, pesquisando e adicionando esses filmes ao carrinho de filmes do módulo.

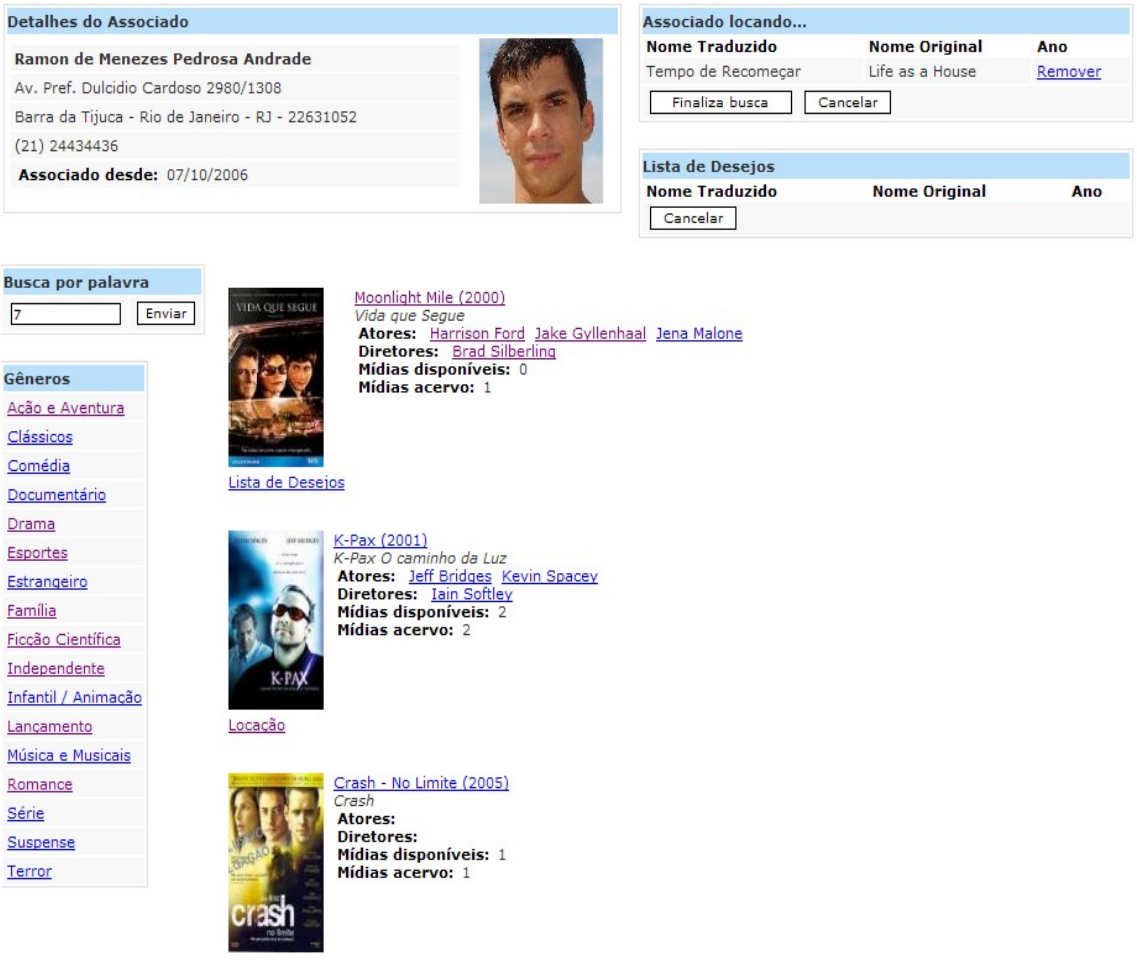

Figura 64: Locação por Telefone

Após o usuário ter completado o carrinho e finalizar a locação, o sistema faz as devidas verificações da quantidade de mídias permitidas no plano do associado e exibe uma nova tela onde o usuário deve preencher com as mídias disponíveis de cada filme que será levada para o associado na casa dele.

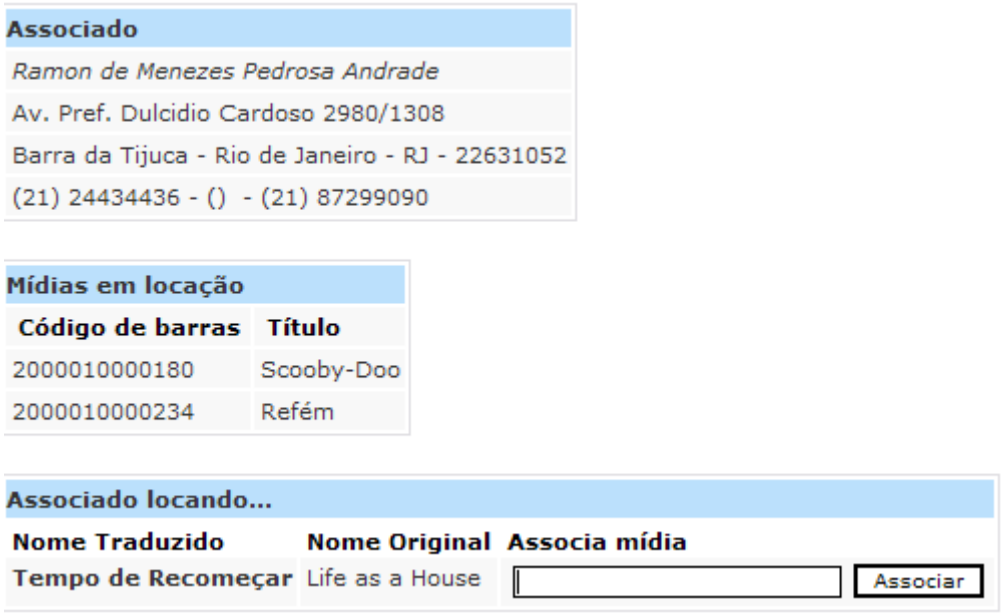

Figura 65: Finalização da Locação por Telefone

- **Relatório**

Esta interface é responsável pela geração de diversos tipos de relatórios que tem por finalidade informar ao usuário do sistema sobre as mais importantes operações do sistema que são as locações de mídias. Como o único dado do sistema que tem possui valor agregado são as mídias, pois elas são o maior patrimônio de uma locadora, é importante possuir o controle sobre as atividades que as mídias estão envolvidas.

Para isso foi desenvolvido esse módulo, que exibirá para o usuário relatórios sobre as situações das mídias alugadas, tais como mídias em locações, histórico das locações do sistema, atividade de locação, associados com mídias acima do plano e também a possibilidade de geração de relatórios dinâmicos, onde o próprio usuário faz a sua *query* de acesso ao banco.

Figura 66: Relatórios

- **Mídias em Locação**

Essa interface exibe informações sobre todas as mídias que estão alugadas, informando o nome do associado que a alugou, a data da locação, o nome do filme que a mídia pertence e a possibilidade de visualizar os detalhes do filme e do associado, clicando em cima do nome dos mesmos.

Código de barras:	Título	Associado	Data locação
2000010000067	Tempo de Recomeçar	Ramon de Menezes Pedrosa Andrade	20/11/2006
2000010000180	Scooby-Doo	Ramon de Menezes Pedrosa Andrade	20/11/2006
2000010000234	Refém	Ramon de Menezes Pedrosa Andrade	20/11/2006
2000010000197	+ velozes+ furiosos	Julia de Marco Silveira	01/11/2006
2000010000227	Anos de Rebeldia	Julia de Marco Silveira	01/11/2006
2000010000012	Moonlight Mile	Marcela Mariana de Almeida Ribeiro	23/10/2006
2000010000173	Scooby-Doo	Marcela Mariana de Almeida Ribeiro	23/10/2006

Figura 67: Relatório de mídias em locação

- **Histórico de Locações**

Essa interface exibe informações sobre todas as locações que ocorreram no sistema, informando o nome do associado que alugou, o nome do filme que a mídia pertencia, a data de locação, a data de retorno de mídia e quantidade de dias que a mídia ficou alugada.

Associado	Código de barras:	Título	Data locação	Data retorno	Dias de locação
Ramon de Menezes Pedrosa Andrade	2000010000067	Tempo de Recomeçar	20/11/2006	-	0
Ramon de Menezes Pedrosa Andrade	2000010000180	Scooby-Doo	20/11/2006	-	0
Ramon de Menezes Pedrosa Andrade	2000010000234	Refém	20/11/2006	-	0
Ramon de Menezes Pedrosa Andrade	2000010000043	Armageddon	19/11/2006	19/11/2006	0
Ramon de Menezes Pedrosa Andrade	2000010000104	Lisbela e o Prisioneiro	19/11/2006	19/11/2006	0
Ramon de Menezes Pedrosa Andrade	2000010000128	K-Pax	19/11/2006	19/11/2006	0
Ramon de Menezes Pedrosa Andrade	2000010000043	Armageddon	05/11/2006	19/11/2006	14

Figura 68: Relatórios de histórico de locações

- **Atividade de Locação**

Essa interface exibe informação dos associados e de suas respectivas locações na loja. Essa funcionalidade permite ao usuário saber qual associado está com mídias em seu poder a certa quantidade de dias e também de associados que não alugam mídias a certa quantidade de dias.

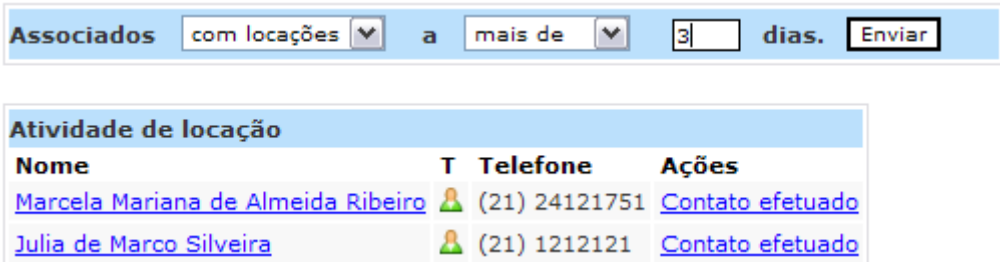

Figura 69: Relatório de atividade de locação

- **Dinâmicos**

Esta interface permite ao usuário do sistema gerar relatórios dinâmicos e importar os relatórios gerados para diversos tipos de formatos tais como XLS (Excel), XML e PDF. O usuário ao acessar esse módulo poderá criar suas próprias *queries* e submeter o formulário. O sistema processará a *query* e exibirá o resultado na tela para o usuário, e este poderá importar o relatório para o formato desejado.

Só serão permitidas a utilização de *queries* de *SELECT* e caso se utilize outro tipo de *query* como *INSERT*, *UPDATE*, *DELETE*, ou qualquer outro comando diferente de *SELECT* não haverá processamento do sistema. Este cuidado visa a garantir a integridade dos dados do banco de dados.

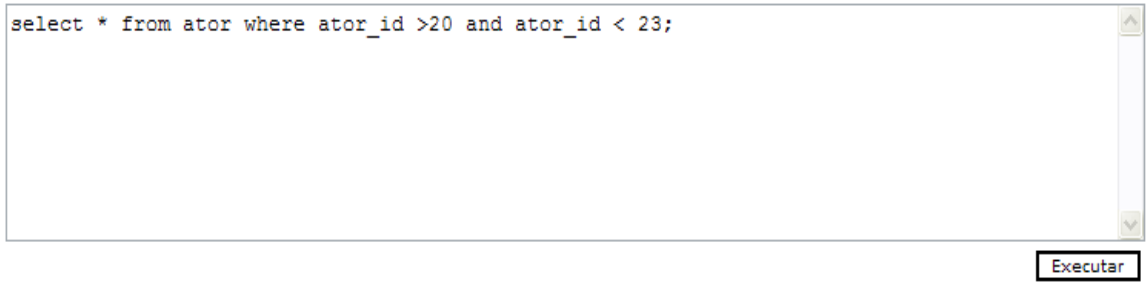

Figura 70: Relatório Dinâmico

- **Gerar Cobrança**

Essa interface é responsável pela garantia da geração da cobrança mensal de cada associado do sistema. A partir desse módulo é gerada a cobrança relativa ao valor da mensalidade de cada associado, baseado nas informações do plano de cada associado.

Os dados gerados serão gravados no banco de dados e permitirão especificar se um associado possui boletos em aberto.

Figura 71: Geração de Cobrança

Sistema de Divulgação

O sistema de Divulgação é o site da locadora e terá como objetivo divulgar e informar ao seu público alvo as informações relevantes dos filmes que são cadastrados no sistema de Gestão e também permitir aos seus associados acesso a outras funcionalidades, tais como visualização de perfil, avaliação de filmes e lista de desejo.

O site da locadora é uma aplicação independente do sistema de Gestão e essa independência garante a segurança do Sistema SisLoc, pois como o site é uma aplicação de acesso livre está mais vulnerável a ataques de *hackers*. Assim através dessa independência os dados do banco têm a garantia de que não serão corrompidos por ataques de fontes externas.

- **Login**

Para permitir o acesso dos associados da locadora a outras funcionalidades do sistema de divulgação, foi desenvolvido esse módulo que permite ao associado da loja se logar e assim visualizar e executar essas novas funcionalidades.

A fim de garantir ainda mais segurança à aplicação foram desenvolvidas diversas formas de restringir o acesso indevido ao sistema, tal como controle de sessão do usuário e encriptação de senha utilizando uma API Java de segurança.

Para se *logar* ao sistema, o associado deve fornecer o seu e-mail cadastrado e a sua senha que o associado recebeu por e-mail que o associado recebeu no momento do seu cadastro. Após submeter o formulário de *login*, o sistema fará as devidas verificações para confirmar a correta identificação do usuário.

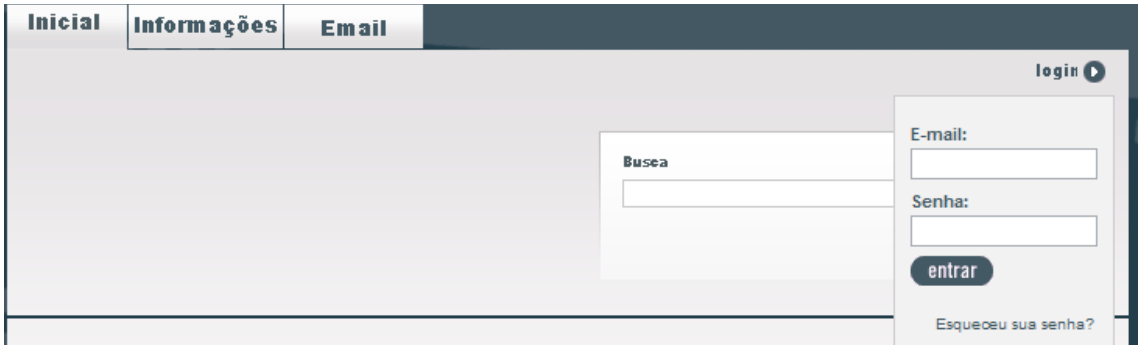

Figura 72: Login no Site

- **Visualizar Perfil**

Esta interface é responsável pela visualização dos dados cadastrais pelo associado. Através dela o associado pode verificar todos os seus dados cadastrais relevantes, tais como: nome do associado, plano mensal, endereço de entrega completo

e outros dados complementares como data de nascimento, CPF, e-mail e telefones de contato.

Para acessá-lo associado, depois de se *logar*, deve clicar no link ”Perfil” através do menu de informações.

Figura 73: Visualização do Perfil

- **Avaliar Filmes**

Através dessa interface o associado pode avaliar os filmes que são cadastrados na base de dados. No site são exibidos os filmes de acordo com a pesquisa realizada pelo associado, então através dos filmes que são exibidos os associados podem avaliar os filmes e essas notas são computadas e assim servem como base para outras pessoas que acessam o site e verificam a avaliação dos associados em relação a este filme.

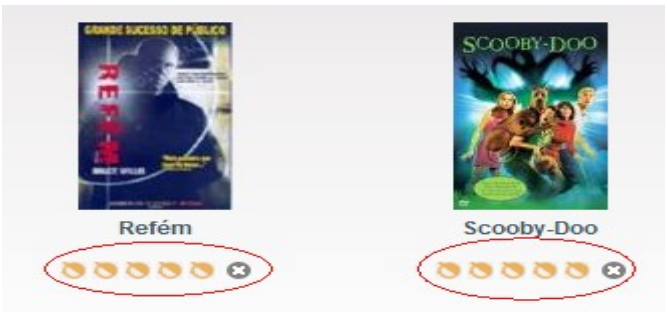

Figura 74: Avaliação de Filmes

- **Lista de Desejo**

Através dessa interface o associado poderá reservar filmes que deseja alugar através do próprio site. O associado adiciona os filmes que deseja ver e assim os usuários do sistema de Gerenciamento, quando o filme que o associado deseja ver estiver disponível, contactam o associado e alugam por telefone o filme, entregando em domicílio o filme desejado.

Figura 75: Adicionar Filmes a Lista de Desejo

Para o associado adicionar um filme a sua lista de desejo, ele deve somente escolher o filme clicando no botão "adicionar" que aparece abaixo dos filmes. Para ver a sua lista de desejo o associado deve clicar no botão "lista"que aparece no menu de navegação.

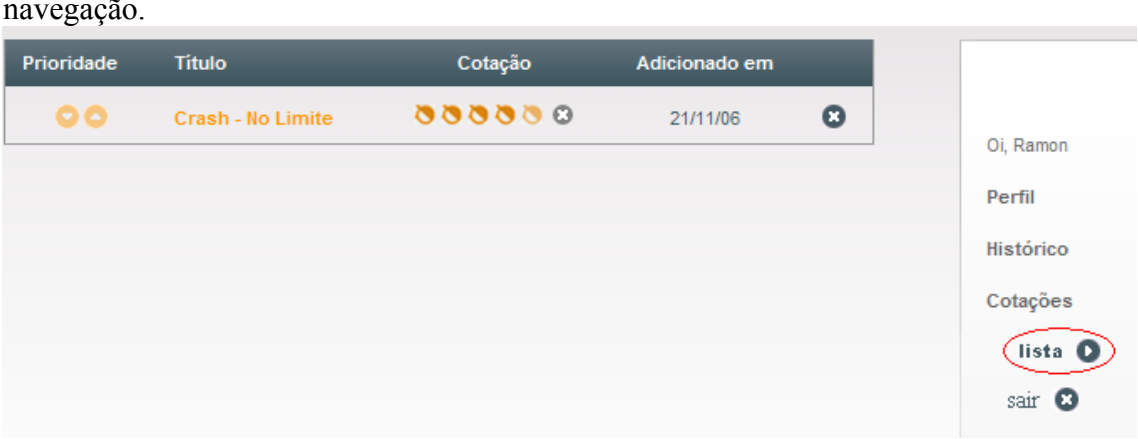

Figura 76: Filmes na Lista de Desejo

- **Histórico**

Essa interface é responsável pela exibição do histórico de locações do associado. O histórico de locações contém informações sobre as locações que o associado efetuou na loja e assim o associado pode verificar os filmes que foram alugados pela sua família.

Para o associado acessar o seu histórico, ele deve clicar no link "Histórico" do *menu* e assim o sistema o redireciona para a tela de histórico onde exibe todos os filmes que o associado e sua família alugaram.

Figura 77: Histórico de Locações do Associado

- **Pesquisa**

Essa interface contém os principais fluxos do sistema de divulgação que são as pesquisas do sistema. Esta interface por não depender das informações do associado é um módulo que pode ser acessado por qualquer visitante do site.

Para permitir bastante flexibilidade nas pesquisas, foi desenvolvida duas diferentes formas de pesquisa de filmes no sistema.

A primeira forma de pesquisa é a pesquisa de filme por categoria. Essa pesquisa é baseada nas categorias dos filmes que são cadastrados no sistema de Gestão. Existem diversos tipos de categorias de filmes que são: Lançamento, Ação e Aventura, Clássicos, Comédia, Documentário, Drama, Esportes, Estrangeiro, Família, Ficção Científica, Independente, Infantil/Animação, Música e Musicais, Romance, Série, Suspense e Terror.

Foram adicionados também duas formas de categorias em que a lista de filmes é gerada dinamicamente e é baseada nas avaliações dos associados em relação aos filmes da loja. Essas duas categorias são *Top 16* e *Top 32*, onde os filmes são listados conforme as maiores pontuações dadas pelos associados.

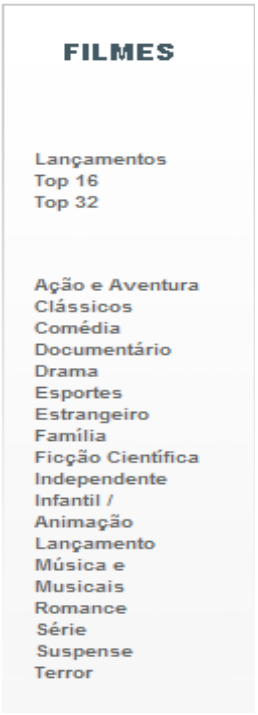

Figura 78: Pesquisa de Filmes por Categoria

Outra forma de pesquisa desenvolvida foi a busca de filmes pela palavra que o associado deseja. Essa busca exibirá os resultados de todos os filmes que contém essa palavra no título, dos atores que possuem essa palavra no nome e dos diretores que possuem essa palavra no nome.

Esse tipo de busca permite que os associados e visitantes possam encontrar os filmes que desejam de maneira rápida e assim verificarem os dados dos filmes, dos atores e dos diretores existentes no site.

Figura 79: Lista de Filmes Pesquisados por palavra

3.2.2 Interfaces de Software

Não existirão interfaces entre o software e outro software aplicativo.

3.2.3 Interfaces de Comunicação

O software usará protocolo TCP/IP implementando assim uma arquitetura cliente/servidor via Web.

3.3 Requisitos de Desempenho

Não há grandes requisitos de desempenho para este software. Mas, por ser uma aplicação, para a Web o tempo de resposta é sempre importante, portanto temos que procurar diminuir o tempo de acesso ao banco de dados que é o gargalo principal do sistema pela necessidade de acesso a disco.

Além disso, tem-se como objetivo máximo deste sistema ter uma alta amigabilidade. Pois, sendo um sistema para Web considerarmos que o usuário foco é imprevisível, podendo ser leigo em diversos aspectos.

3.4 Restrições de Projeto

Como restrições básicas para este sistema impomos a utilização do padrão de projeto MVC (*Model View Controller*), através da *framework* WebWork e da modelagem conceitual orientada a objeto utilizando padronização UML.

3.5 Atributos

Os atributos principais para a garantia de qualidade do Sistema de Gerenciamento de Locadora - SisLoc são:

- Amigabilidade: o sistema é amigável, tendo uma interface limpa, clara e objetiva que visa uma maior automação das operações diárias da locadora.
- Consistência de dados: sistema é consistente, ou seja, os dados do sistema não são inseridos sem que toda transação já tenha sido efetuada.
- Segurança e Confiabilidade: o sistema é confiável, pois o sistema é o responsável pela administração e do correto funcionamento das locadoras.
- Usabilidade: este atributo nos levou a utilização da análise orientada a objeto que nos permite maior reutilização de código.
- Escalabilidade: o sistema tem uma fácil adaptação para uma demanda maior de usuários.

3.6 Outros Requisitos

Não há outros requisitos de software além dos que já foram citados no documento.