UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

PARADIGMA ORIENTADO A OBJETOS COM MVC E COMUNICAÇÃO BASEADA EM AJAX. ESTUDO DE CASO: SISTEMA DE LOCADORA VIRTUAL

TRABALHO DE GRADUAÇÃO

Cleiton Cechin

Santa Maria, RS, Brasil
2007

PARADIGMA ORIENTADO A OBJETOS COM MVC E COMUNICAÇÃO BASEADA EM AJAX. ESTUDO DE CASO: SISTEMA DE LOCADORA VIRTUAL

por

CLEITON CECHIN
cechin@inf.ufsm.br

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria (UFSM)

Requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

Orientadora: Profª. Oni Reasilvia Sichonany

Trabalho de Graduação Nº 217
Santa Maria, RS, Brasil
2007

UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

A Comissão Examinadora, abaixo assinada, aprova o Trabalho de Graduação.

PARADIGMA ORIENTADO A OBJETOS COM MVC E COMUNICAÇÃO BASEADA EM AJAX. ESTUDO DE CASO: SISTEMA DE LOCADORA VIRTUAL

Elaborado por
CLEITON CECHIN
cechin@inf.ufsm.br
como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação

Comissão Examinadora:

_____
Profª. Oni Reasilvia Sichonany (Orientadora)

_____
Prof^{or}. Cláudio Rocha Lobato

_____
Profª. Roseclea Duarte Medina

Santa Maria, 28 de fevereiro de 2007.

AGRADECIMENTOS

Dedico este trabalho de graduação a todas as pessoas, em especial a minha família, que me apoiaram na trajetória de estudante, desde os tempos no primeiro grau, mas principalmente na época pré-vestibular e nos anos de universitário.

Agradeço em especial a minha família, Valcir, Lourdes e Emerson, pelo apoio dado em todas as fases de da minha vida de estudante.

À minha namorada, Gislaine, pelo apoio e pelo carinho nas horas de dificuldade.

Aos professores, principalmente à professora Oni, pelas orientações e ensinamentos transmitidos.

Aos colegas, pelo compartilhamento de conhecimentos e ajudas oferecidas durante todos os anos de faculdade, em especial ao Diego e ao Leandro.

Aos amigos, pelos momentos de descontração vividos durante o período de estudante.

RESUMO

UNIVERSIDADE FEDERAL DE SANTA MARIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TRABALHO DE GRADUAÇÃO

PARADIGMA ORIENTADO A OBJETOS COM MVC E COMUNICAÇÃO BASEADA EM AJAX. ESTUDO DE CASO: SISTEMA DE LOCADORA VIRTUAL

Autor: Cleiton Cechin
Orientadora: Profª. Oni Reasilvia Sichonany

O presente trabalho de graduação, versa no estudo da implementação de sistemas *WEB* usando a comunicação baseada em AJAX e programação orientada a objetos com a estrutura MVC.

A programação orientada a objetos baseada em MVC, é uma forma de dividir as partes de implementação de um objeto, separando a parte visual responsável pela construção da interface, da parte de controle e das propriedades do objeto.

A comunicação baseada em AJAX é uma nova tecnologia de troca de dados entre cliente e servidor, que é capaz de fazer requisições ao servidor sem a necessidade de recarregar todo o *site* novamente.

Um estudo de caso foi a implementação de uma locadora de vídeos virtual, a linguagem usada será PHP 5 orientada a objetos, o banco de dados usado é MySQL 5.1 e a tecnologia AJAX para a troca dos dados com o servidor.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AJAX	Asynchronous Javascript and XML
MVC	Module View Control
HTML	HyperText Markup Language
PHP	Hypertext Preprocessor
URL	Universal Resource Locator
WEB	World Wide Web (WWW)
XML	EXtensible Markup Language
DOM	Document Object Model
CSS	Cascating Style Sheet
XSLT	Extensible Stylesheet Language Transformation
API	Application Programming Interface
SQL	Structured Query Language
SGBD	Sistema de Gerenciamento de Banco de Dados
DER	Diagrama de Entidade Relacionamento

1 INTRODUÇÃO

O presente trabalho de graduação versa no estudo de duas novas tecnologias para implementação de sistemas *WEB*, AJAX [1] e programação orientada a objetos [5] baseada em MVC[12]. Todos os sistemas *WEB* têm como objetivo, ser o mais rápido possível, para qualquer tipo de usuário que venha a acessá-lo, sabe-se de fato que muitos *sites* são acessados por pessoas com internet de acesso limitado e com pouca banda[1]. Por isso a tecnologia AJAX foi estudada, pois é capaz de diminuir consideravelmente o consumo de banda, comparando com o modelo clássico dos *sites* de internet.

Outro objetivo de todo o sistema é que este possa ter o mínimo de custo de programação e a maior facilidade de entendimento. Por isso, o objetivo do estudo da orientação a objetos baseada em MVC, foi o de aprender a organizar uma aplicação *WEB*, de modo que esta se torne de fácil modificação, para que futuramente possam ser adicionadas novas funcionalidades, ou também editadas as funções presentes com o mínimo de custo de implementação.

Como estudo de caso para o projeto foi implementado um sistema de vídeo locadora virtual, no qual os clientes ganham um *login*[2] e uma senha de acesso ao "Portal do Cliente", onde este pode fazer buscas, encomendas e reservas das obras existentes do estabelecimento. Os funcionários da locadora têm acesso restrito às partes administrativas do sistema, ou seja, cadastros de clientes, filmes, álbuns musicais, jogos, atores, diretores, gêneros, além das operações de empréstimo, devolução, confirmação de encomenda, e configurações. O sistema de busca é por qualquer campo da obra, também existe um sistema de bônus para o cliente, sendo este notificado em seu portal, podendo retirar alguma obra gratuita ou com algum desconto. Os usuários que não possuem um *login* de acesso, podem apenas visualizar e fazer busca no acervo da locadora.

---

[1] Velocidade relacionada com uma conexão de internet.
[2] Conjunto de caracteres solicitado para os usuários que por algum motivo necessitam acessar um sistema computacional.

2 REVISÃO BIBLIOGRÁFICA

Neste tópico são demonstradas, a tecnologia AJAX que é responsável por tornar o sistema o mais parecido possível com um programa *desktop*[3], a linguagem de programação PHP em sua versão 5 e o modelo de orientação a objetos [5] com MVC usado nas aplicações *WEB*, o DBDesigner [15] que é a ferramenta que foi usada para modelagem dos dados e o banco de dados MySQL .

2.1 AJAX

O termo AJAX consiste em Javascript [16] e XML [17] assíncronos, utiliza Javascript juntamente com XML para tornar o navegador[4] mais interativo com o usuário, ou seja, fazendo o site na internet parecer um programa *desktop*. Para que isso aconteça o site em si é carregado apenas uma vez, e a cada nova interação do usuário é feita uma nova requisição ao servidor[5] que retorna apenas os dados necessários e estes são mostrados no navegador do cliente em tempo real, sem carregar todo site novamente como acontece na maioria dos *sites* na internet [1].

2.1.1 O assincronismo de AJAX

O assincronismo é a grande novidade do AJAX, pois podemos fazer conexões, baixar documentos do servidor, e realizar as operações sobre a árvore DOM [9], ou seja, modificar o conteúdo da página em tempo real, a qualquer momento [3].

O usuário carrega a página, e uma vez carregada, sem que haja necessidade de recarregá-la, pode modificá-la ao seu gosto baseado nas informações pedidas ao servidor, a qualquer instante [4], ao contrário do modelo síncrono, onde havendo uma interação do usuário, todos os dados seriam carregados novamente.

---

[3] Consiste de um ambiente gráfico adequado ao usuário, onde ele possa abrir algumas janelas de programas e efetuar operações básicas sobre as janelas abertas e sobre o ambiente em si.
[4] Programa capaz de demonstrar visualmente para o usuário o conteúdo de um site da internet.
[5] Computador onde são guardados os dados que podem ser acessados na rede.

A Figura 1 mostra as diferenças entre o sincronismo das aplicações clássicas e as baseadas em AJAX [2].

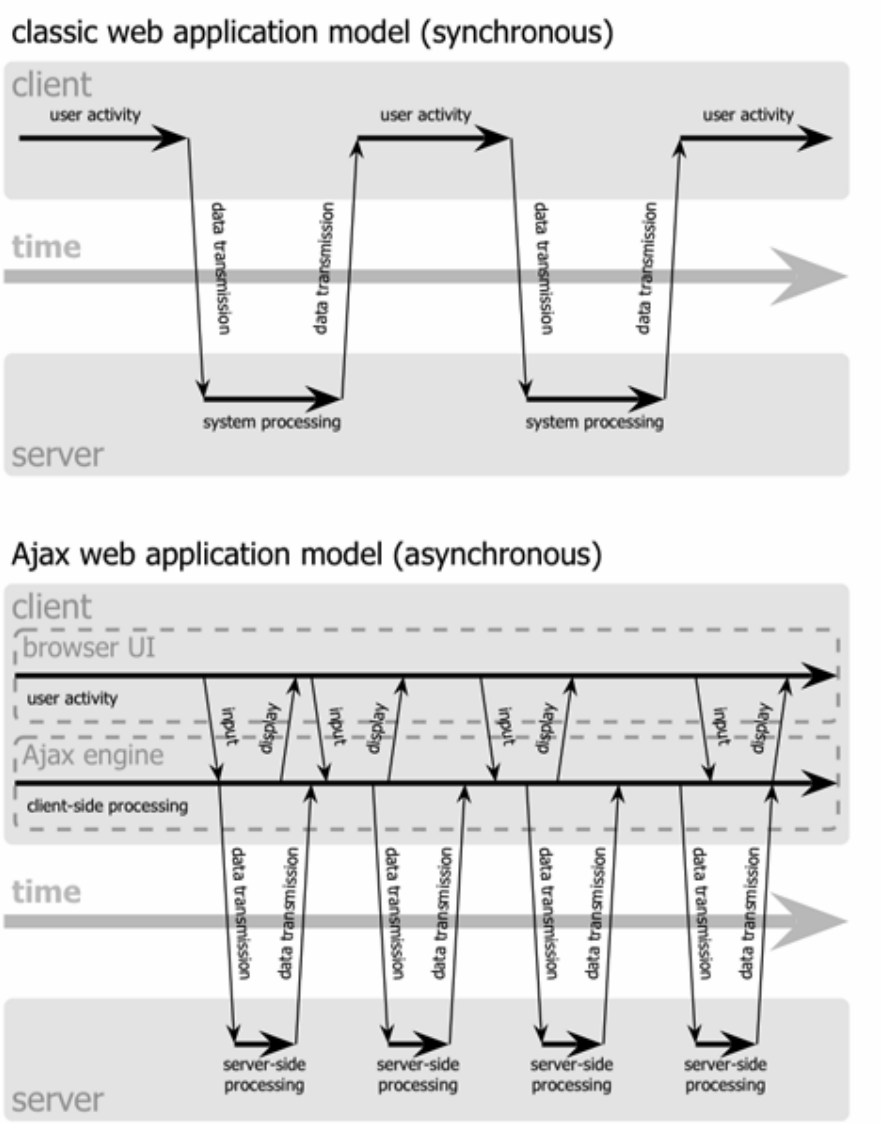

Figura 1 Sincronismo em aplicações *WEB* clássicas (acima) comparando com assincronismo das aplicações AJAX (abaixo)

2.1.2 O navegador hospeda[6] uma aplicação e não o conteúdo, e o servidor fornecem dados e não conteúdo.

[6] Neste caso tem o significado de possuir, armazenar, conter.

Essas diferenças são devido ao fato de que nas aplicações clássicas a cada interação do usuário, o navegador recebe uma página (conteúdo) nova do servidor e re-desenha todos os seus dados, mesmo os dados que já estavam no navegador são buscados no servidor e redesenhados a cada interação do usuário[1]. A Figura 2 demonstra o modelo clássico das páginas *WEB*, onde o cliente recebe a página inicial ao entrar com o endereço[7] *WEB* do servidor, a partir dai a cada novo pedido do usuário será devolvida outra página ao cliente, até chegarmos à página final [1].

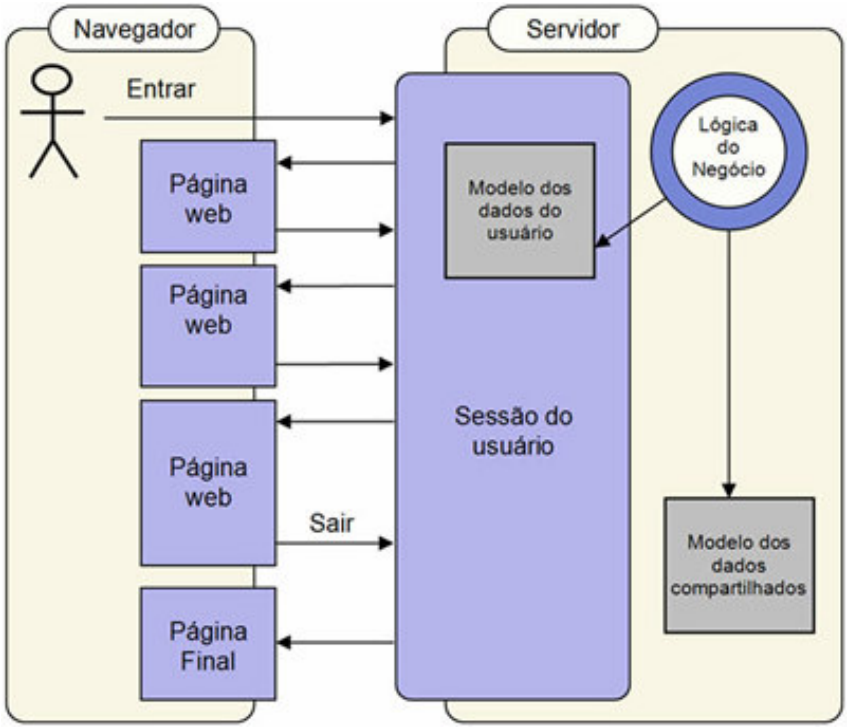

Figura 2 Modelo clássico das páginas web

Ao contrário disso, usando AJAX, a página inicial seria carregada apenas uma vez, com o cliente tornando-se uma aplicação responsável por tratar os pedidos feitos pelo usuário, buscando apenas os dados necessários e atualizando estes em tempo real no navegador [1]. A Figura 3 demonstra uma aplicação usando AJAX, na primeira requisição do cliente o servidor fornece uma aplicação ao cliente, esta contém a página inicial e a lógica programada baseada em AJAX, que irá tratar das

[7] Identificador único para que um computador possa ser acessado por outros em uma rede, principalmente na internet.

próximas interações do usuário [1]. Agora a cada novo *click* do usuário, essa lógica será responsável por requisitar apenas os dados necessários para o servidor e mostra-los na tela.

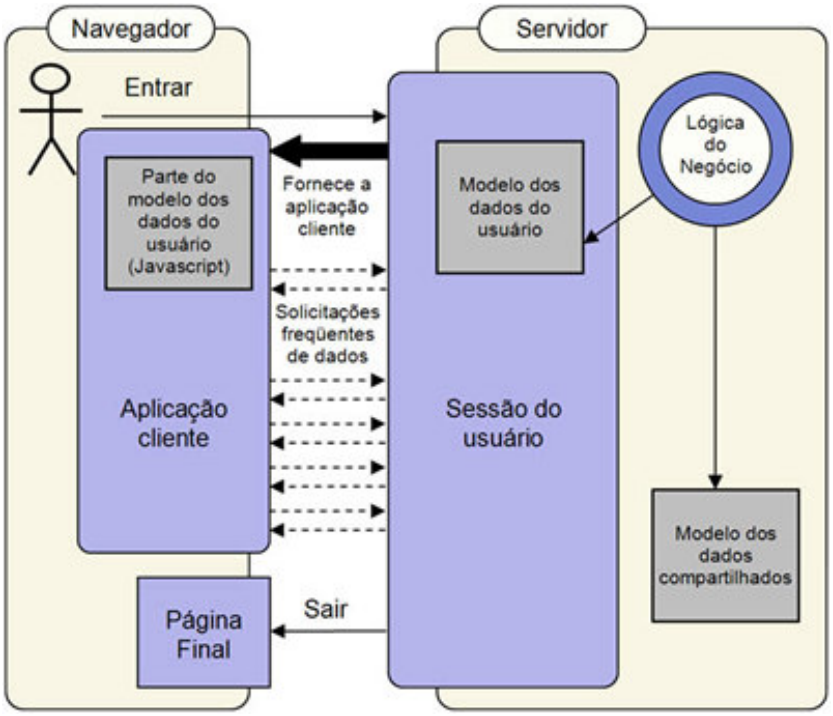

Figura 3 Modelo *web* baseado em AJAX

2.1.3 Comparando o tráfego o modelo clássico e o modelo baseado em AJAX

Em uma aplicação AJAX, o tráfego[8] tem sua maior intensidade no início, pois na primeira requisição o usuário recebe a aplicação inteira que é maior que a página inicial de uma aplicação *WEB* convencional. Mas à medida que a interações aumentarem, usando AJAX, a comunicação consumirá muito menos banda, pois a cada requisição apenas os dados necessários serão trafegados e no modelo clássico o consumo de banda cresce na proporção do tamanho das páginas que vão sendo carregadas [1]. Isso é demonstrado na Figura 4 [1].

[8] Quantidade de dados que são trocados em uma rede.

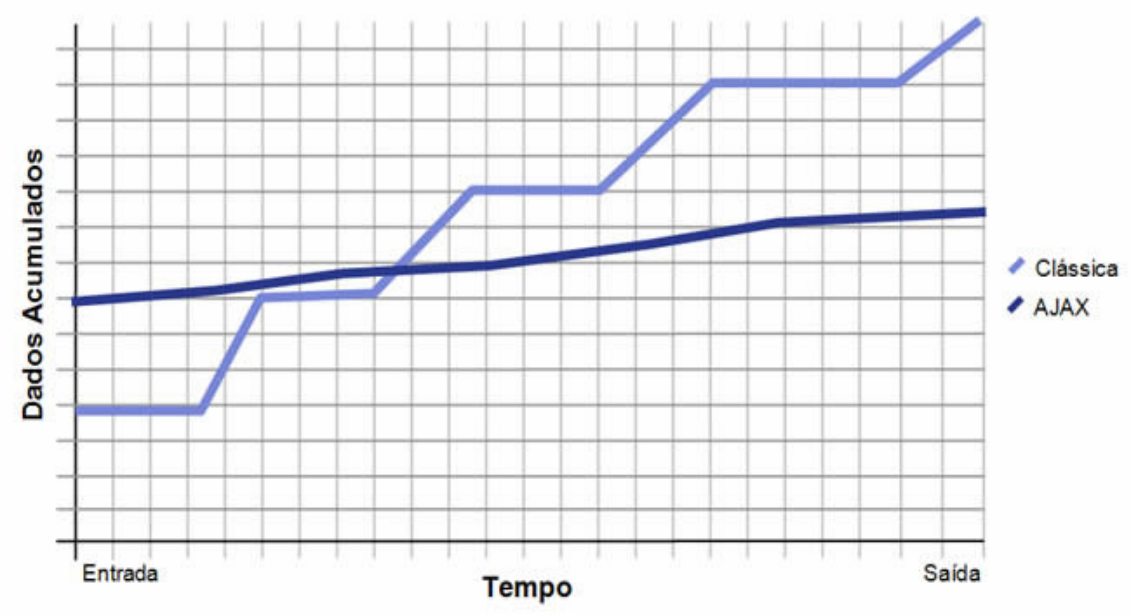

Figura 4 Comparação de consumo de banda do modelo clássico com o modelo baseado em AJAX.

2.1.4 AJAX: uma junção de tecnologias

As funcionalidades de AJAX são divididas em partes. A parte da recuperação assíncrona dos dados é feita pelo objeto XMLHttpRequest [3], para as alterações em tempo real usa-se a chamada árvore DOM, a apresentação dos dados usando HTML [18] e CSS[20] intercambio e manipulação dos dados usando XML [17] e XSLT[19], tudo em conjunto com javascript [2].

2.1.4.1 O Objeto XMLHTTPRequest

Este objeto Javascript torna possível a comunicação assíncrona com o servidor, sem a necessidade de recarregar a página por completo [2]. O objeto requisita dados de um endereço especificado.

Tabela 1 Os métodos do objeto XMLHTTPRequest

Método	**Descrição**
abort()	Cancela a requisição corrente.
getAllResponseHeaders()	Retorna todos os cabeçalhos, com os nomes e seus valores, como uma string.
getResponseHeader("*headerLabel*")	Retorna uma string para o cabeçalho especificado.
open("*method*", "*URL*"[, *asyncFlag*[, "*userName*"[, "*password*"]]])	Cria uma requisição pendente, com uma URL, um método, e outros atributos adicionais.
send(*content*)	Envia uma requsição, com a opção de postar uma string ou um objeto DOM.
setRequestHeader("*label*", "*value*")	Seta o valor do cabeçalho label com o valor value.

Fonte: Apple Developer Connection [3]

Na Tabela 1 mostra os métodos do objeto XMLHTTPRequest, os principais são, *open* e *send*, o primeiro constrói uma requisição ao endereço do parâmetro *URL*, o parâmetro *method* especifica o método da requisição, se GET ou POST e o parâmetro *asyncFlag* especifica uma comunicação assíncrona ou síncrona dos dados, e o método send envia a requisição [3].

Tabela 2 As propriedades do objeto XMLHTTPRequest

Propriedade	Descrição
Onreadystatechange	Função quando é disparado um evento, a cada mudança da variável readyState.
ReadyState	Status do Objeto: 0 = Não inicializado 1 = Carregando 2 = Carregado 3 = Interativo 4 = Completo
ResponseText	Resposta do servidor convertida em texto.
ResponseXML	Objeto DOM retornado do servidor em XML.
Status	Código retornado do servidor, 404 se “Não Achou” ou 200 se “”OK”
StatusText	String do Código de Status

Fonte: Apple Developer Connection [3]

As propriedades do objeto XMLHTTPRequest são explicadas na Tabela 2.A propriedade Onreadystatechange é uma função Javascript onde serão tratados os dados retornados da requisição, e é ativada na primeira mudança da propriedade readyState [2]. As propriedades responseText e ResponseXML são variáveis onde os dados retornados podem ser acessados. A primeira é onde retornam os dados em modo texto e a segunda é onde é retornado um objeto DOM do pedido de um documento XML.

2.1.4.2 DOM

O DOM [9], modelo de objeto de documentos, é uma API padrão para documentos HTML e XML, que define uma estrutura dos documentos para que

possam ser acessados e manipulados. Com o DOM, os programadores podem navegar pela estrutura de um documento, HTML ou XML, e alterar, apagar e adicionar elementos e conteúdo [7].

No padrão DOM os documentos podem ser visualizados como uma árvore onde as folhas desta árvore são objetos, cada um destes contendo seus atributos e funcionalidades.

Na Tabela 3 temos um exemplo de um documento HTML [7].

Tabela 3 Exemplo de um documento HTML

```
<TABLE>
<TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```

Fonte: AMTECHS, W3C Português [7]

Na Figura 5 está demonstrada a árvore DOM para o documento da Tabela 3 [7].

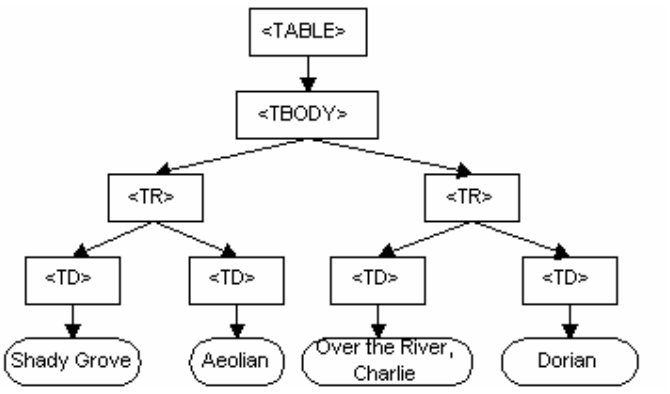

Figura 5 Representação DOM em forma de uma árvore

O DOM especifica interfaces[9] que podem ser usadas para gerenciamento de documentos HTML e XML, a Tabela 4 descreve a interface para a *tag*[10] table HTML [7].

Tabela 4 Interface HTMLTableElement

```
interface HTMLTableElement : HTMLElement {
  // modificado pelo DOM Nível 2:
           attribute HTMLTableCaptionElement caption;
                                        // raises(DOMException) on setting
  // modificado pelo DOM Nível 2:
           attribute HTMLTableSeção Element tHead;
                                       // raises(DOMException) on setting
  // modificado pelo DOM Nível 2:
           attribute HTMLTableSeção Element tFoot;
                                        // raises(DOMException) on setting
  readonly attribute HTMLCollection  rows;
  readonly attribute HTMLCollection  tBodies;
           attribute DOMString       align;
           attribute DOMString       bgColor;
           attribute DOMString       border;
           attribute DOMString       cellPadding;
           attribute DOMString       cellSpacing;
           attribute DOMString       frame;
           attribute DOMString       rules;
           attribute DOMString       summary;
           attribute DOMString       width;
  HTMLElement        createTHead();
  void               deleteTHead();
  HTMLElement        createTFoot();
  void               deleteTFoot();
  HTMLElement        createCaption();
  void               deleteCaption();
  // modificado pelo DOM Nível 2:
  HTMLElement        insertRow(in long índice )
                                         raises(DOMException);
  // modificado pelo DOM Nível 2:
  void               deleteRow(in long índice )
                                         raises(DOMException);
};
```

Fonte: AMTECHS, W3C Português [7]

Nota-se que a interface possui atributos e métodos, como o atributo rows que retorna todas as linhas da tabela, bem como os métodos insertRow e deleteRow, que respectivamente insere ou apaga uma linha da tabela.

[9] É um padrão para determinar a forma de comunicação entre componentes de software.
[10] São estruturas HTML que consistem em breves instruções, tendo uma marca de início e outra de fim, por exemplo, tudo oque estiver escrito dentro das tags , aparecerá em negrito no navegador.

2.1.4.3 Javascript

Esta linguagem é a responsável por atualizar os dados em tempo real no navegador sem necessidade de recarregar a página.

Isso só é possível, pois a linguagem JavaScript possui objetos capazes de realizar operações de manipulação da árvore DOM, manipulação XML e o objeto XMLHTTPRequest[4].

O DOM permite ao Javascript manipular o conteúdo da página *WEB*. Pode acrescentar, modificar ou finalizar etiquetas e atributos, através dos quais podemos operar sobre o documento de todas as formas possíveis [4].

Para acessar os elementos do documento basta seguir a estrutura na árvore[11]. Baseado no exemplo da Figura 5 são mostrados alguns exemplos de como percorrer numa estrutura DOM.

1) document.body.childNodes[0], isso faz referencia a raiz da árvore, ou seja a tabela (table).

2) document.body.childNodes[0].rows[0], faz referencia a primeira linha da tabela.

3) document.body.childNodes[0].rows[1].cells[2], faz referencia a terceira coluna da segunda linha da tabela.

2.2 Orientação a objetos

Esta técnica foi criada para que um problema fosse mais facilmente decomposto em subgrupos relacionados, podendo-se traduzir esses grupos em unidades auto-contidas, chamadas objetos [6].

2.2.1 Conceitos básicos

[11] Estrutura de dados muito comum em programação estruturada, a qual descreve uma hierarquia de memória onde dados podem ser temporariamente armazenados.

A orientação a objetos possui alguns conceitos básicos essenciais para o seu entendimento, dentre os quais podemos citar: objeto, abstração, abstração, encapsulamento e herança.

Um objeto é um tipo de dado construído pelo programador para atender um determinado propósito, é composto pelos seus dados e operações que serão capazes de manipulá-los [6], estas operações são frequentemente chamadas de métodos.

Abstração significa concentrar-se no que um objeto é e faz, antes de se decidir como ele será implementado [5]. Com isso o programador deve estar ciente das funcionalidades de cada objeto antes de começar a implementação. O uso apropriado de abstração permite que um mesmo modelo conceitual seja utilizado para todas as fases de desenvolvimento de um sistema, desde sua análise até sua documentação [5].

Encapsulamento consiste em separar os aspectos externos de um objeto, os quais são acessíveis a outros objetos, dos detalhes internos de implementação do objeto, os quais permanecem escondidos dos outros objetos [5], ou seja, um objeto que chama um método, não está interessado na implementação deste, mas apenas no resultado. O uso de encapsulamento permite que a implementação de um objeto possa ser modificada sem afetar as aplicações que usam este objeto [6].

Herança é uma técnica de orientação a objetos que promove compartilhamento de informações em diversos níveis distintos, ou seja, objetos podem adquirir propriedades de outros objetos. Herança de estrutura de dados e comportamento permite que estruturas comuns sejam compartilhadas entre diversas classes derivadas similares sem redundância [5]. A herança de código é uma maneira de economizar de código e é também capaz de nos fazer reconhecer que operações diferentes são na verdade a mesma coisa, o que reduz o número de casos distintos que devem ser entendidos e analisados.

Outra forma de herança é a generalização que nada mais é do que vários objetos herdando as características do mesmo objeto. A Figura 6 ilustra três objetos, filme, álbum e jogo, herdando as propriedades do objeto obra [5].

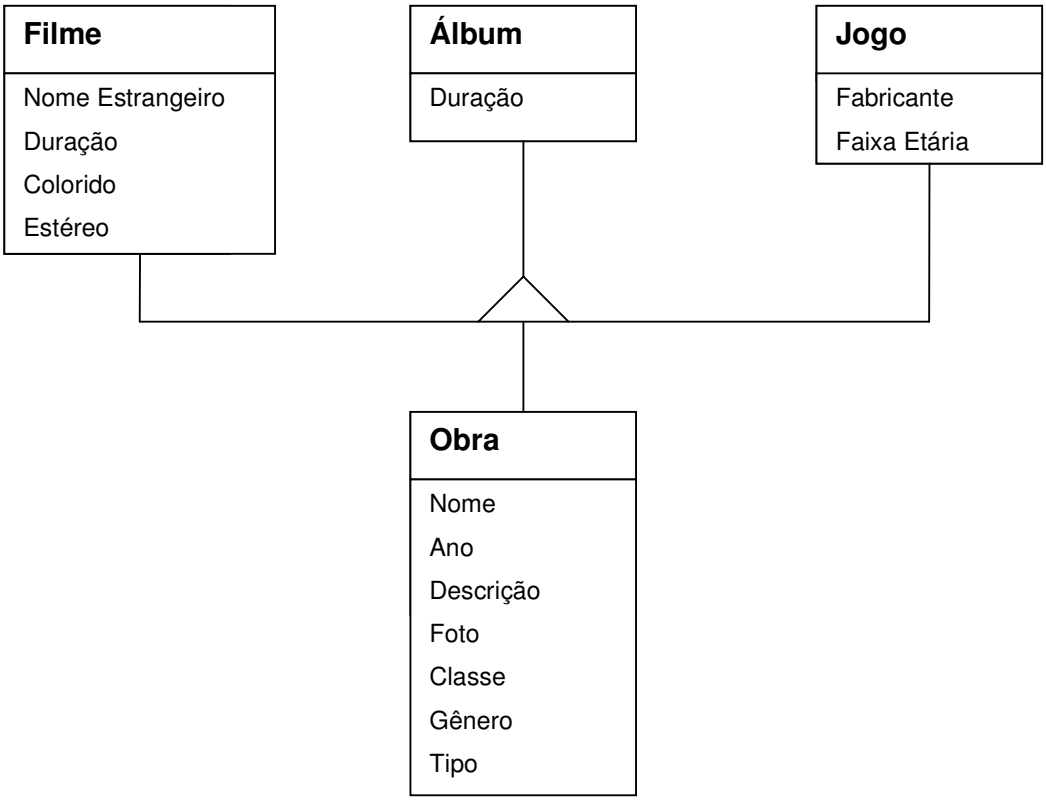

Figura 6 Representação diagramática para generalização

2.2.2 PHP como linguagem orientada a objetos

Na linguagem PHP a orientação a objetos foi introduzida na versão 4, mas no PHP 5 há um novo modelo de objeto. O tratamento de objetos do PHP foi completamente re-escrito, permitindo uma performance melhor e mais vantagens [14].

2.2.2.1 PHP com MVC

O MVC [12] é uma maneira de organizar o código de forma que separa a estrutura de uma aplicação em 3 partes: a regra de negócios(*Model*), a visão (*View*) e o controle de execução(*Control*) [21].

Dentre as vantagens de usar MVC temos: reaproveitamento de código, facilidade de manutenção, integração de equipes, divisão de tarefas, facilidade de alteração na interface, implementação de segurança.

Visão é a parte da interface com o usuário, onde é mostrado o que foi obtido do controle [21]. A Figura 7 representa um exemplo de visão para um cadastro de clientes. Nota-se que temos apenas a parte visual, o parâmetro $_REQUEST[“action”] diz onde vai ser postado o envio do formulário, este parâmetro é passado pelo controle [27].

```
<html>
    <head>
        <title>
            Cadastrar Cliente
        </title>
    </head>
    <body>

    <form action="<?=$_REQUEST["action"]?>" method="POST">

        Nome: <input type="text" name="nome" /><br/>
        Email: <input type="text" name="email" /><br/><br/>

        <input type="submit" value="Cadastrar">

    </form>

    </body>
</html>
```

Figura 7 MVC: Exemplo de Visão

Controle é onde serão processadas todas as requisições feitas através da interface (Visão). O controle também acessa o Modelo a fim de obter determinadas informações. Toda lógica da aplicação (validações, atribuições, etc) é feita no controle [21]. A Figura 8 representa um exemplo de controle referente à visão da Figura 7. Nota-se na figura abaixo que existem dois testes, o primeiro confere se a ação é cadastrar, com isso de alguma maneira deve ser mostrado o formulário de cadastro de clientes. No segundo teste é conferido se a ação é gravar, isto significa que o formulário de cadastro de clientes foi enviado pelo usuário e os dados devem ser gravados no banco de dados [27].

Modelo é onde esta o objeto propriamente dito, com suas propriedades e atributos a serem acessados pela parte de controle. A Figura 9 representa o modelo referente ao exemplo acima de cadastro de clientes [27].

```php
<?
    include_once('classes/cliente/cliente.php');

    class control    {
        function execute() {
            switch ($_GET['acao']) {
                case "cadastrar":
                    $_REQUEST['action'] = "controle.php?module=cliente&path=cadastrar&acao=gravar";

                    $return = "cadastrar";
                    break;

                case "gravar":
                    $cliente    = new Cliente();


                    $cliente->setNome($_POST["nome"]);
                    $cliente->setEmail($_POST["email"]);
                    Aqui insere o cliente no banco de dados

                    $return = "gravar";

                    break;
            }

            return $return;
        }
    }
?>
```

Figura 8 MVC: Exemplo de Controle

```php
<?

    class Cliente {
        var $idCliente;
        var $nome;
        var $email;

        public function setIdCliente($idCliente) {
            $this->idCliente = $idCliente;
        }
        public function getIdCliente() {
            return $this->idCliente;
        }

        public function setNome($nome)     {
            $this->nome = $nome;
        }
        public function getNome() {
            return $this->nome;
        }

        public function setEmail($email) {
            $this->email = $email;
        }
        public function getEmail() {
            return $this->email;
        }
    }

?>
```

Figura 9 MVC: Exemplo de Modelo

2.3 Mysql

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL[12] como interface. É atualmente um dos bancos de dados mais populares, com mais de 4 milhões de instalações pelo mundo, este sucesso se deve principalmente a fácil integração com a linguagem PHP [22].

As principais vantagens deste banco de dados são:

1) Portabilidade, suportando praticamente todas as plataformas existentes [22].

2) Compatibilidade com as mais diversas linguagens de programação [22].

3) A conectividade, velocidade, e segurança fazem com que o MySQL seja altamente adaptável para acessar bancos de dados na Internet[23].

[12] Structured Query Language - Linguagem de Consulta Estruturada é uma linguagem padrão usada para manipular os dados de um banco de dados.

4) É um software livre, ou seja, é possível para qualquer um usar e modificar o programa. Qualquer pessoa pode fazer *download* do MySQL pela Internet e usá-lo sem pagar nada [23].

5) Facilidade de uso.

6) Em suas versões mais recentes possui a maioria das funcionalidades existentes nos servidores de banco de dados comerciais, destas como, *views*, *stored procedures*, *triggers* e integridade referencial [24].

2.4 A ferramenta DBDesigner

É um software livre, multi - plataforma, permite engenharia reversa[13], faz a sincronia no banco de dados das alterações realizadas no DER, a interface com o usuário é muito bem elaborada, tornando o seu uso bastante simples, salva os arquivos em XML, gera relatórios em HTML, é muito bem documentado [25].

É feito especialmente para o MySQL, gerando tabelas e consultas levando em conta a sintaxe SQL desse banco de dados[26].

[13] A partir das tabelas do banco de dados é capaz de gerar o modelo visual.

3 ESTUDO DE CASO: SISTEMA DE LOCADORA VIRTUAL

Este capítulo demonstra o sistema propriamente dito, começando pelas suas funcionalidades e após explicando a modelagem do banco de dados, com as relações entre as entidades do sistema. Também está sendo mostrada a modelagem funcional do sistema com os casos de uso dos atores presentes e finalmente é demonstrada a implementação baseada em AJAX e orientação a objetos com a estrutura MVC.

3.1 O Sistema de Locadora Virtual

Este tópico refere-se ao sistema em si, com as funcionalidades que cada tipo de usuário tem permissão para acessar. São demonstradas algumas telas de exemplo de cadastro, listagem e busca que são padrões em todo o sistema.

3.1.1 Portal do Administrador

No portal do administrador ou funcionário, é onde todas as operações administrativas da locadora são realizadas, quem tiver um login de funcionário, terá acesso a todos os cadastros. Os cadastros principais são os de cliente, autorizado, jogo, álbum, filme, cópia de jogo, cópia de álbum e cópia de filme. Existem também os cadastros extras que são: ator, cantor, classe, diretor, fabricante de jogo, gênero, endereço, mídia, música, faixa etária e idioma. Dentre outras operações administrativas estão: empréstimo, devolução, confirmação de reserva e encomendas, relatórios dos mais diversos tipos, configurações referentes à locadora, locações e reservas.

Os cadastros são padronizados como o cadastro de jogo demonstrado na Figura 10, cada cadastro também possui uma pesquisa e uma listagem, ambas padronizadas.

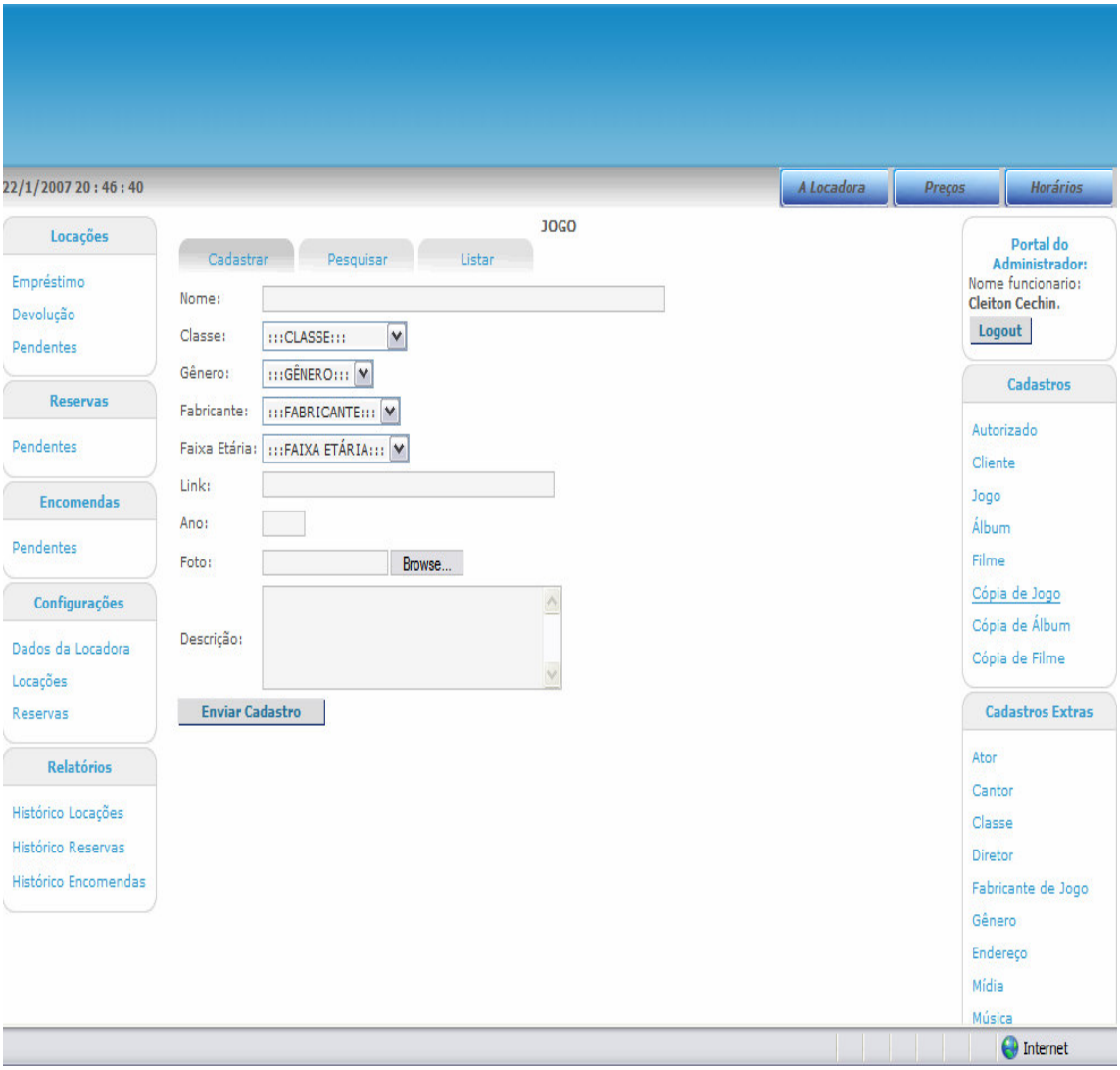

Figura 10 Tela padrão de cadastros com cadastro de jogo

Na Figura 11 está sendo demonstrada a pesquisa de jogo, como exemplo da pesquisa padrão. A busca pelos dados é feita por qualquer campo do cadastro conforme a decisão do usuário.

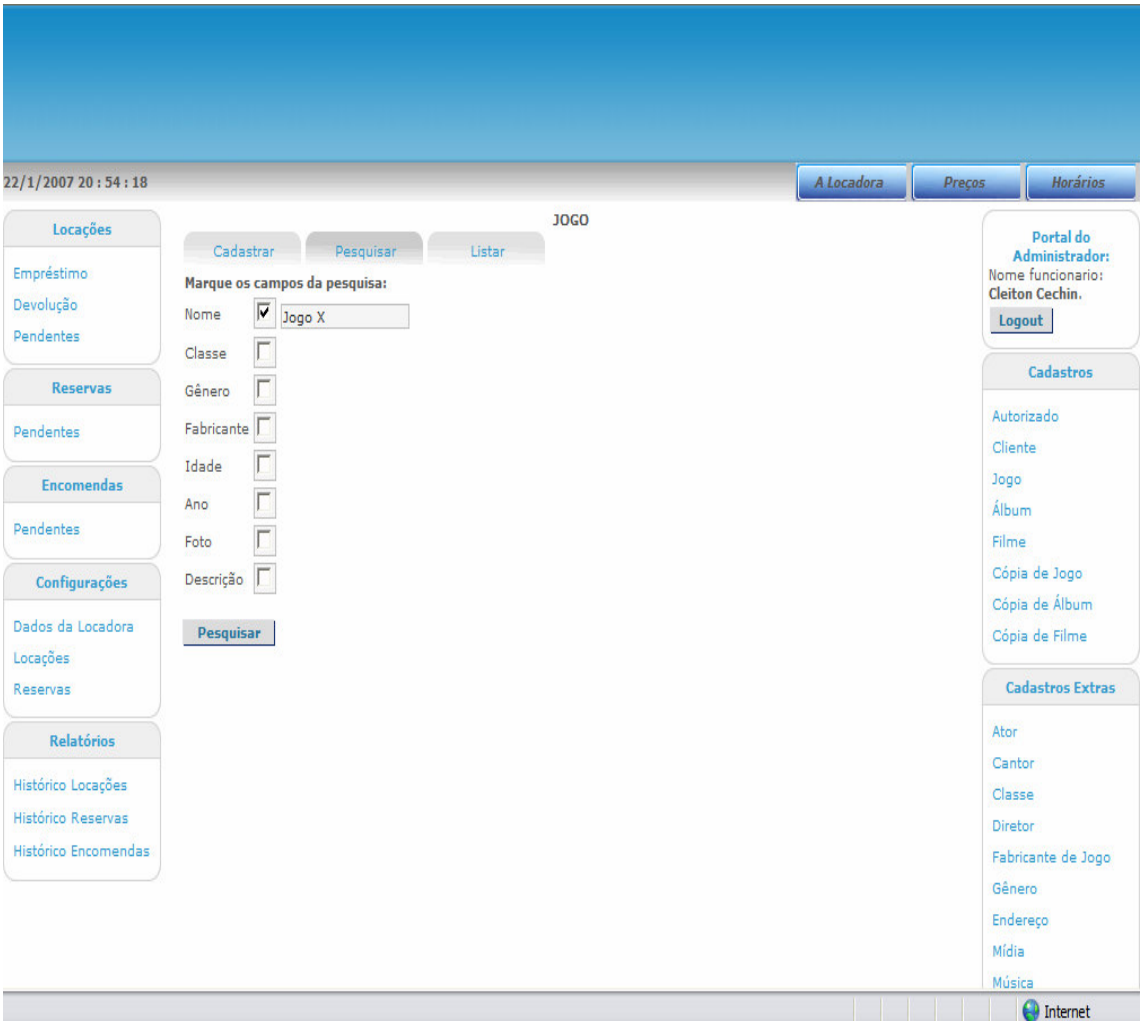

Figura 11 Tela de pesquisa padrão dos cadastros com pesquisa de jogo

Na Figura 12 pode-se visualizar a listagem de jogo como exemplo das listagens padrões, a amostragem dos dados é feita por qualquer campo do cadastro conforme a seleção dos campos feita pelo usuário Também possui a possibilidade da geração de documento para impressão dos dados que estão presentes na lista.

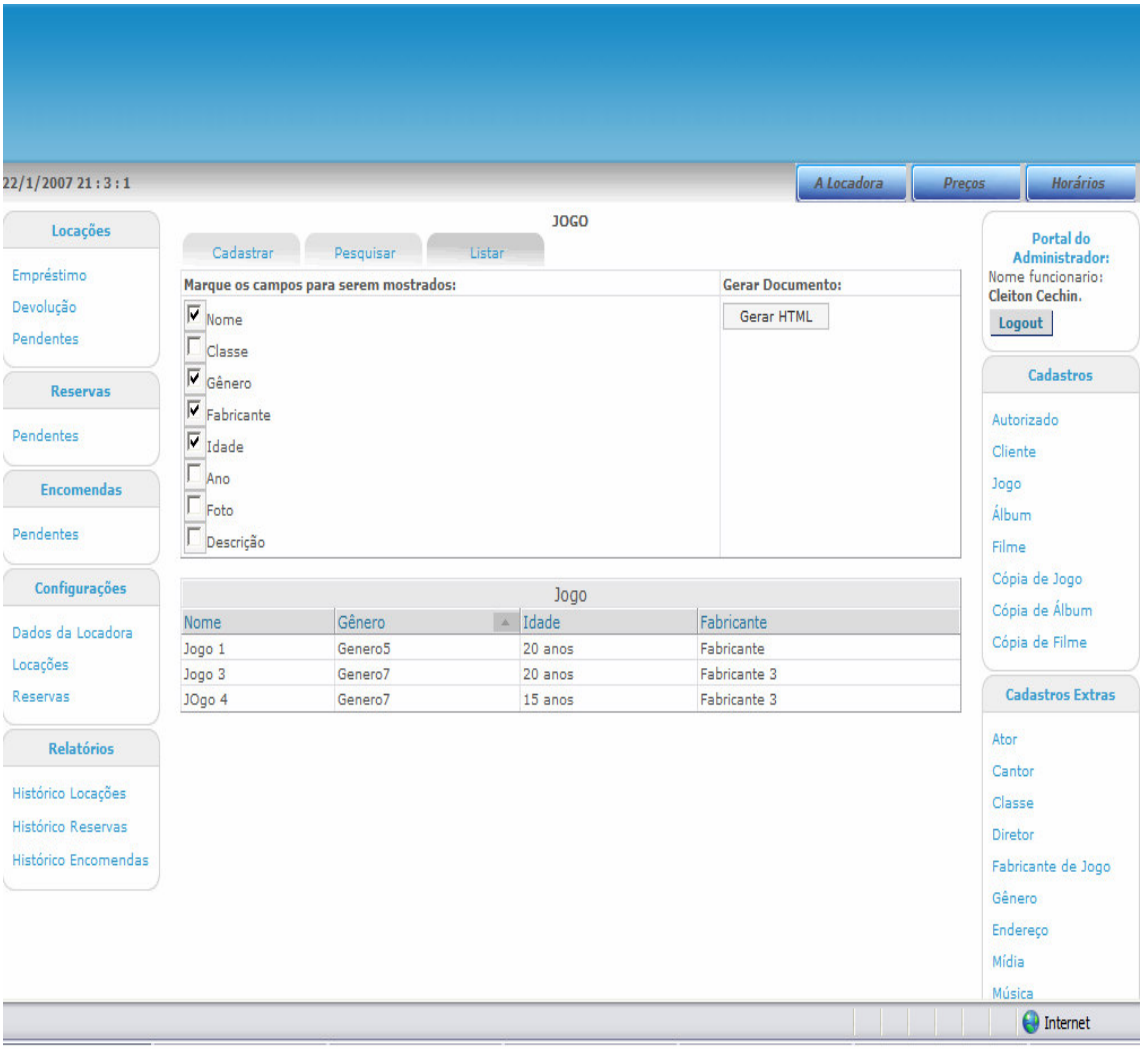

Figura 12 Tela de listagem padrão dos cadastros com listagem de jogo

A Figura 13 demonstra a tela de empréstimo e a Figura 14 a devolução no portal do administrador.

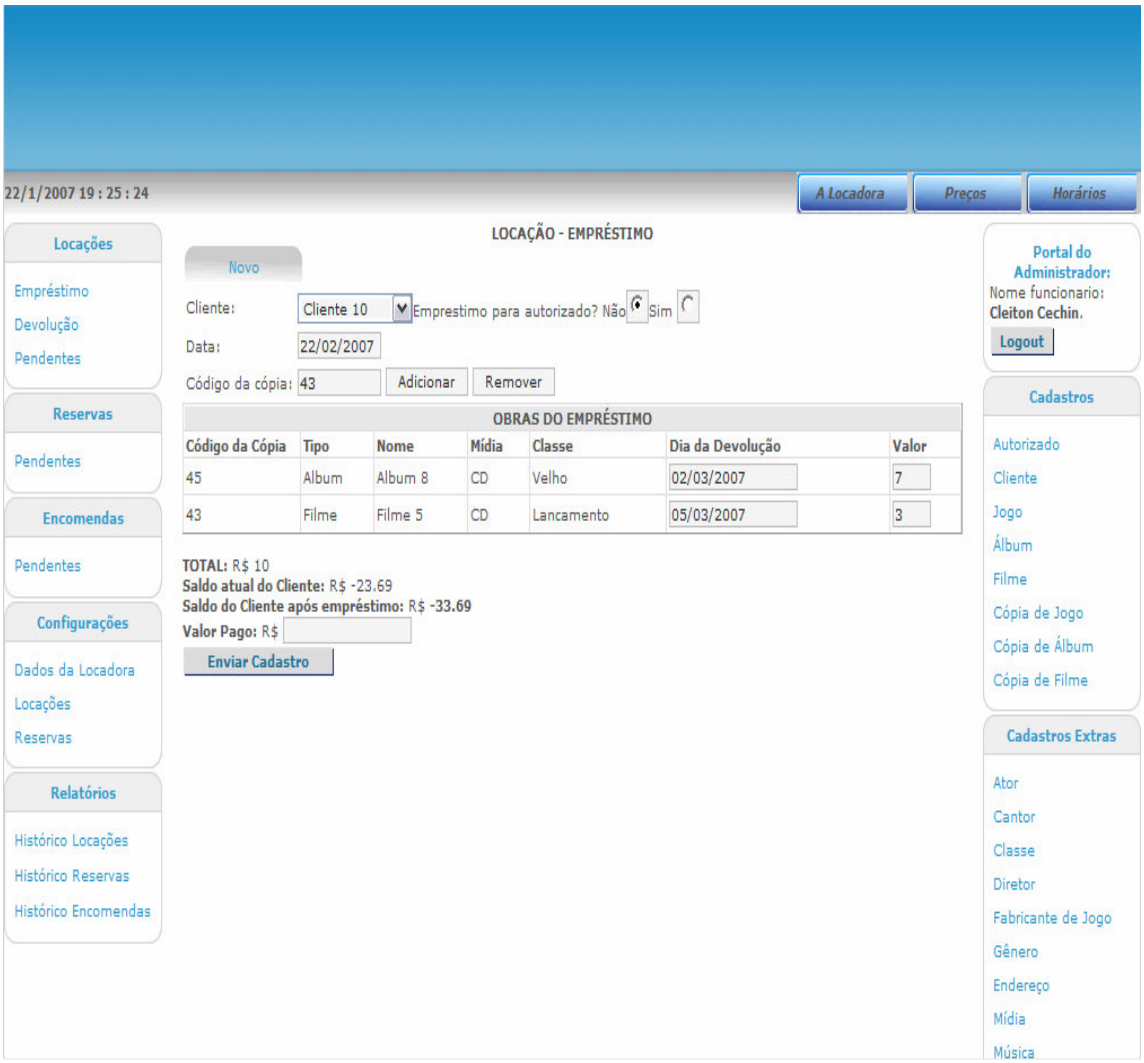

Figura 13 Portal do Administrador com tela de Empréstimo

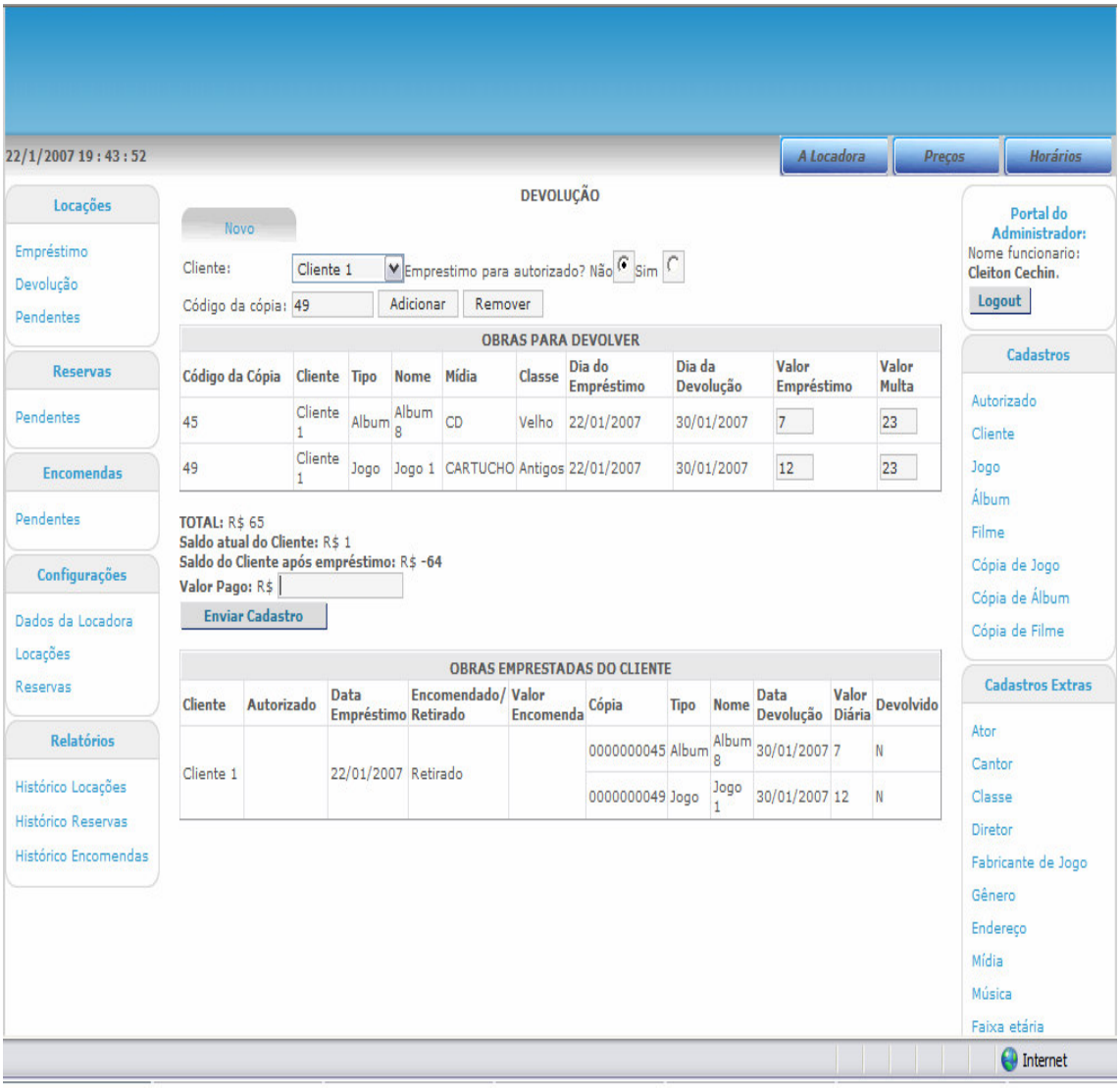

Figura 14 Portal do Administrador com tela de Devolução

3.1.2 Portal do Cliente

No Portal do Cliente, este tem acesso ao acervo das obras e cópias pertencentes à locadora, sendo capaz de listar e procurar no mesmo. Também tem permissão para reservar e encomendar cópias de obras, além de visualizar relatórios sobre as suas reservas, encomendas e empréstimo pendentes. A busca e visualização de obras e de cópias podem ser feitas detalhadamente no acervo, através do procurar e listar padrões dos cadastros, ou por alguns campos na pesquisa rápida.

A Figura 15 representa o Portal do Cliente.

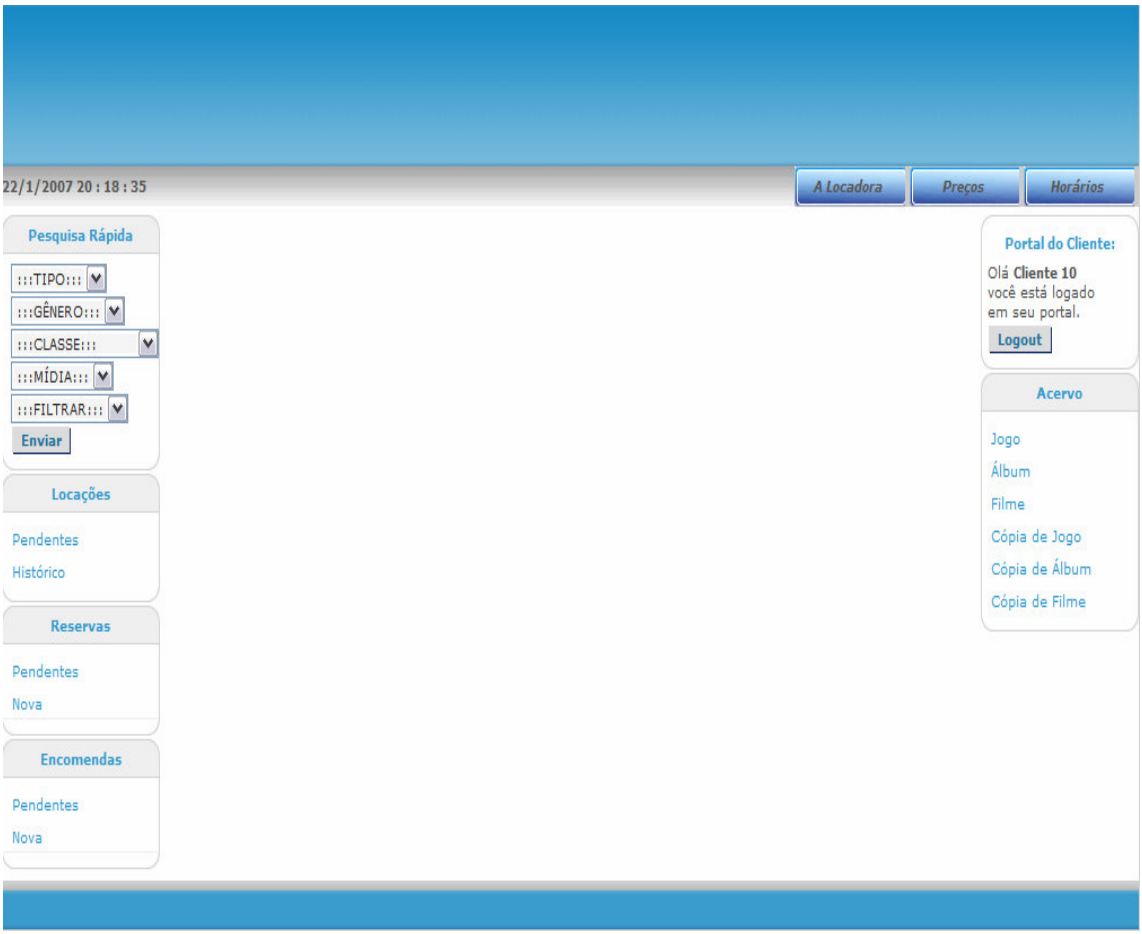

Figura 15 Portal do Cliente

3.1.3 Portal da Locadora

Aqui é a parte do site para os usuários que não tem um *login* no sistema. Estes apenas têm permissões para procurar e visualizar o acervo das obras e cópias pertencentes à locadora. Como no portal do cliente, a busca e visualização de obras e de cópias podem ser feitas detalhadamente no acervo através do procurar e listar padrões dos cadastros, ou por alguns campos na pesquisa rápida.

A Figura 16 ilustra o Portal da Locadora com a pesquisa rápida sendo utilizada.

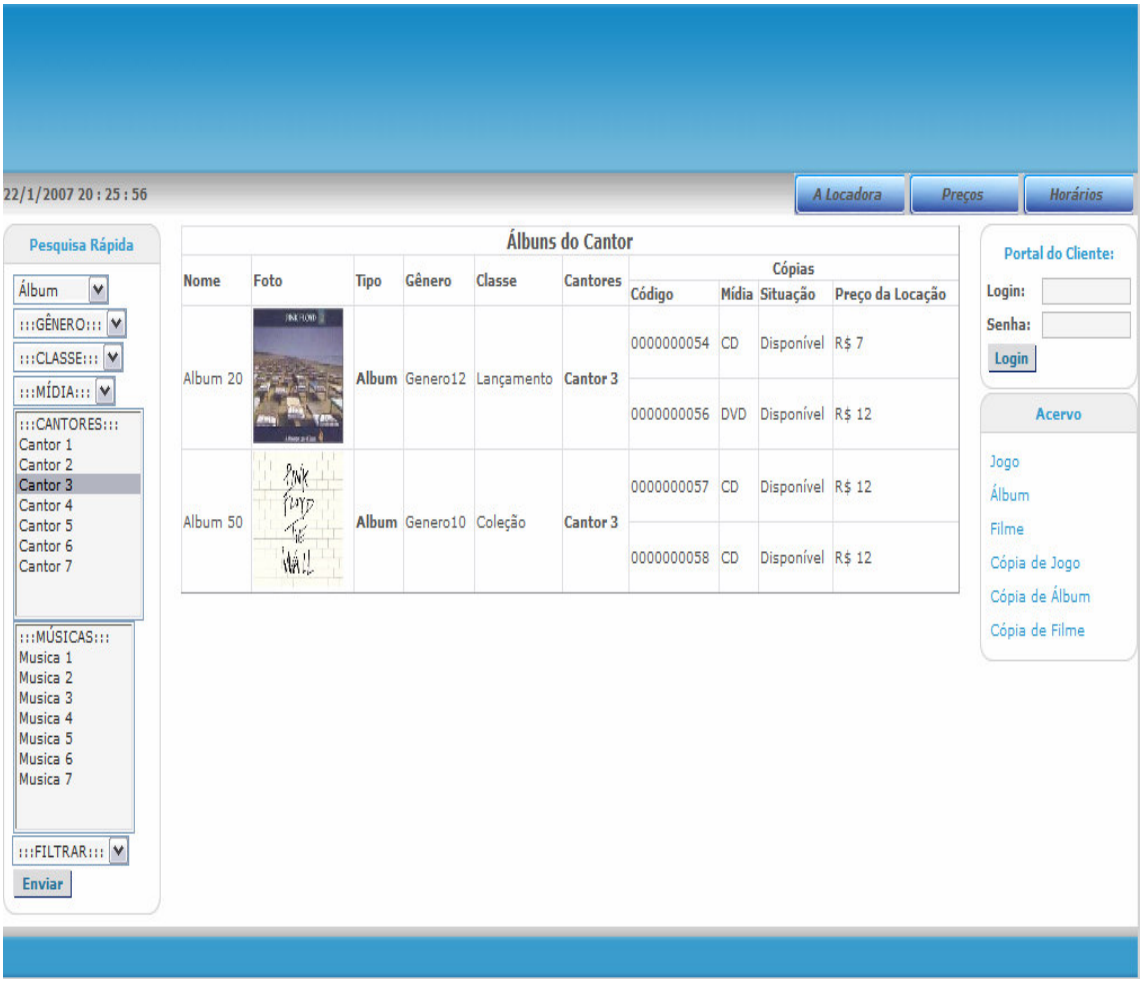

Figura 16 Portal da Locadora com pesquisa rápida de álbuns do Cantor

3.2 Modelagem Funcional

Neste capitulo são demonstrados os casos de uso dos atores presentes no sistema, que são: Usuário *WEB*, Cliente e Funcionário.

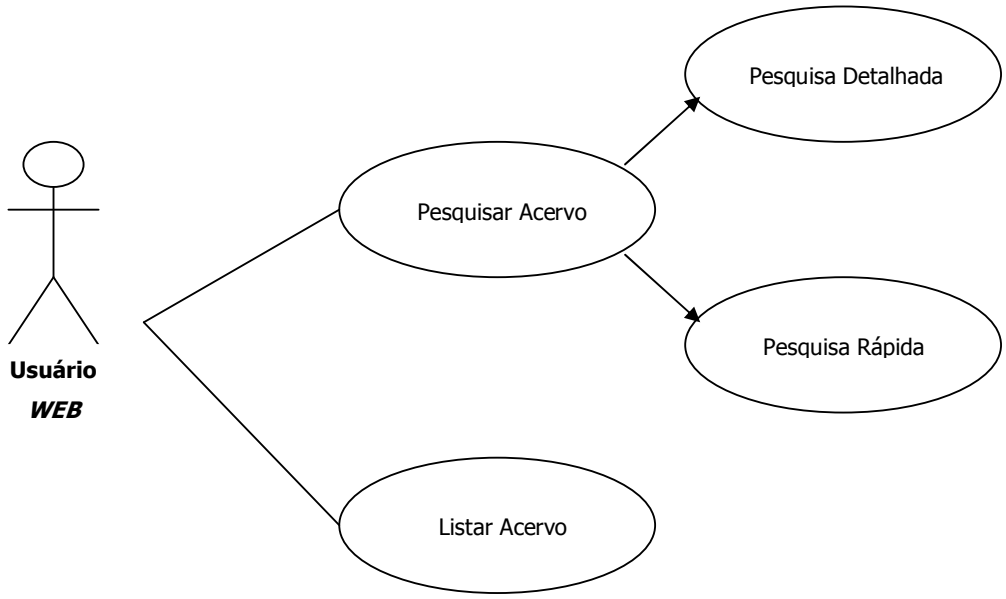

Figura 17 Casos de uso do ator Usuário *WEB*

Os casos de uso do ator Usuário *WEB*, que é o usuário que acessará o Portal da Locadora apenas, pois não tem um *login* no sistema, estão descritos na Figura 17 e são os seguintes:

- Pesquisar Acervo (Jogo, Álbum, Filme e suas cópias): divide-se em Pesquisa Detalhada e Pesquisa Rápida que são pesquisas realizadas sobre os dados do acervo.

- Listar Acervo: listagem dos dados do acervo.

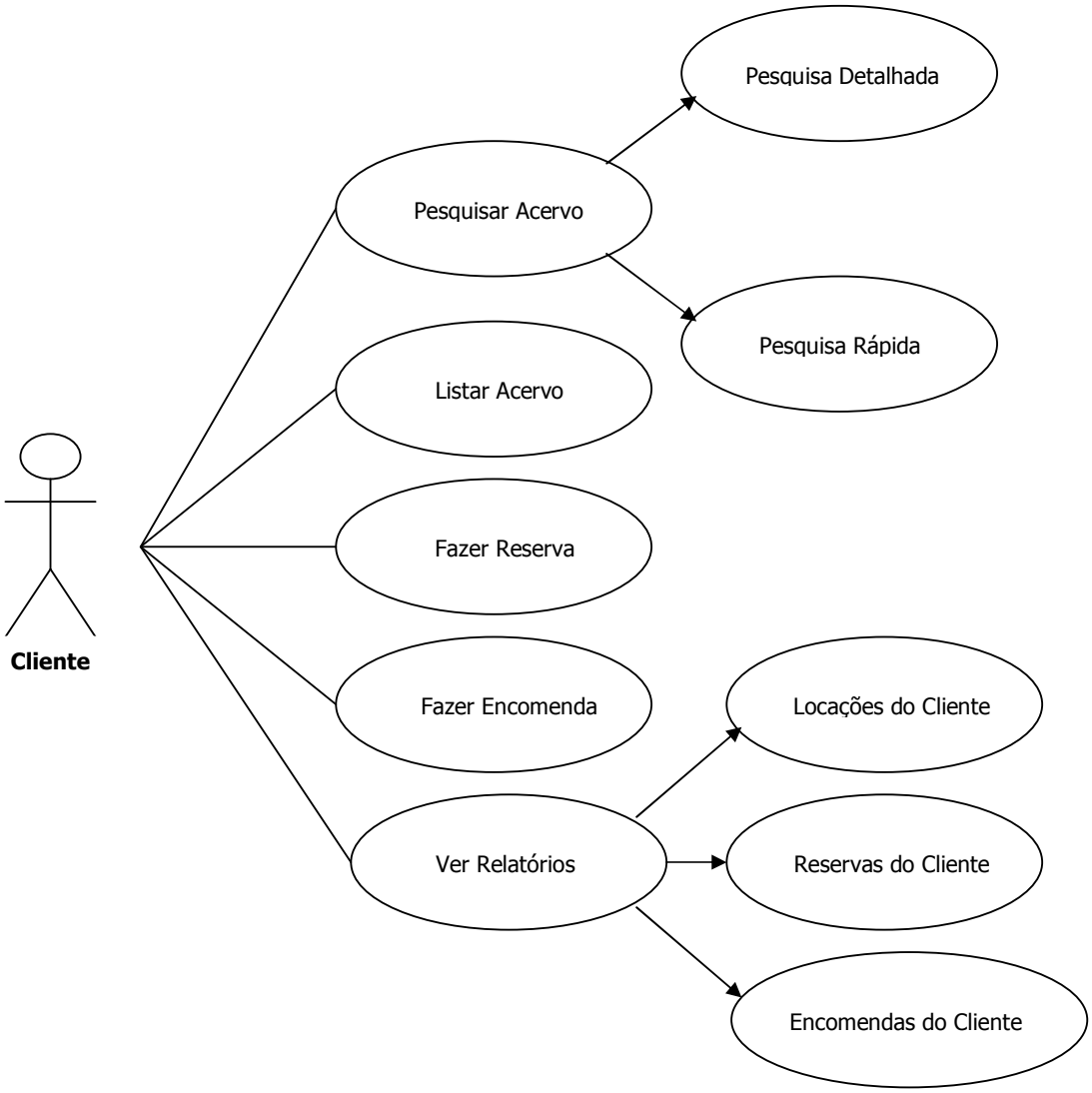

Figura 18 Casos de uso do ator Cliente

Os casos de uso do ator Cliente, que é o usuário que tem um *login* para acessar o Portal do Cliente, estão descritos na Figura 18 e são os seguintes:

- Pesquisar Acervo (Jogo, Álbum, Filme e suas cópias): divide-se em Pesquisa Detalhada e Pesquisa Rápida que são pesquisas realizadas sobre os dados do acervo.

- Listar Acervo: listagem dos dados do acervo.

- Fazer Reserva: reservar cópias do acervo.

- Fazer Reserva: encomendar cópias do acervo.

- Ver Relatórios: mostrar relatórios de locações, reservas e encomendas do cliente.

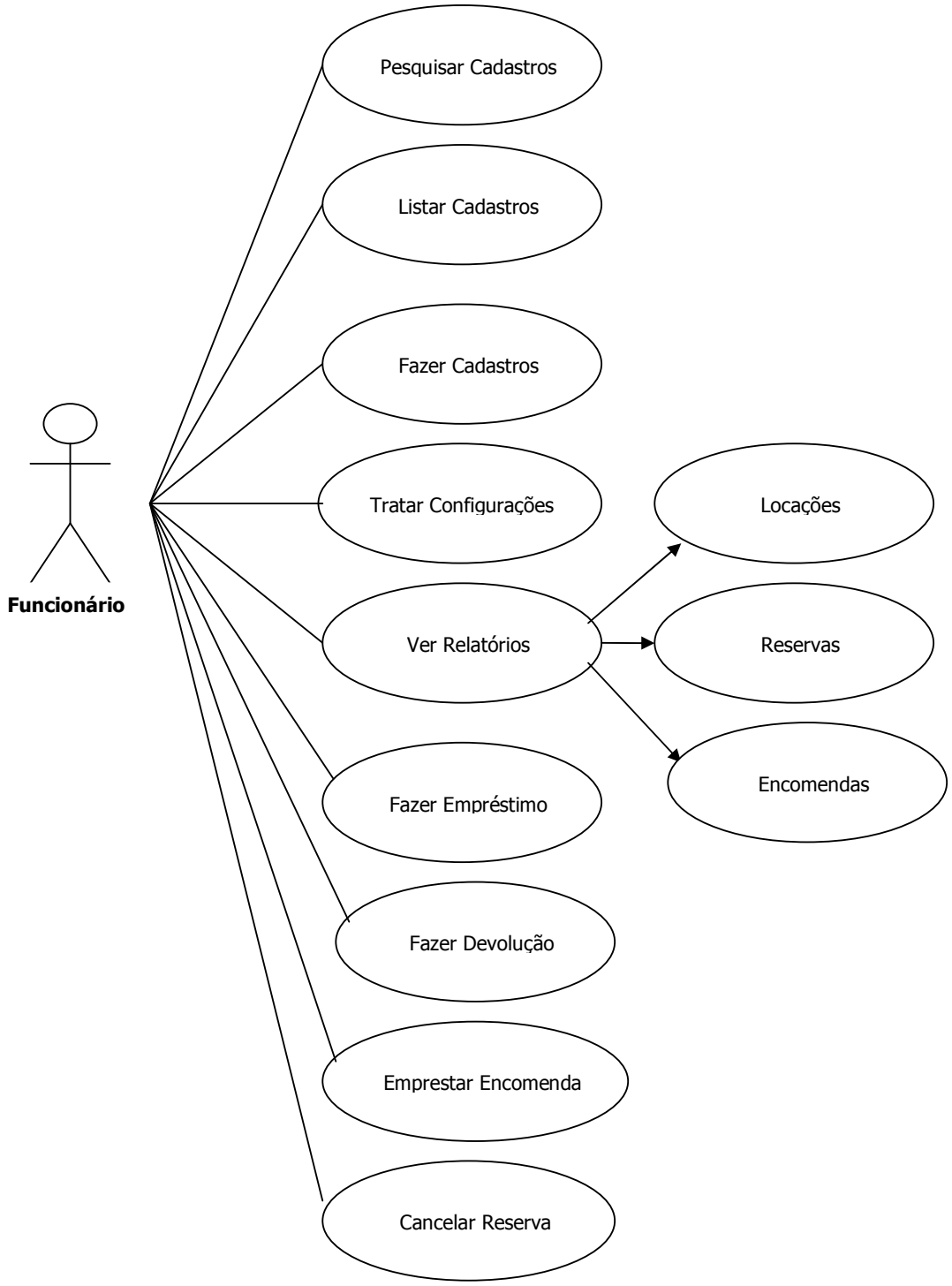

Figura 19 Casos de uso do ator Funcionário

Os casos de uso do ator Funcionário, que é o usuário que tem um *login* para o Portal do Administrador, estão descritos na Figura 19 e são os seguintes:

- Pesquisar Cadastros (Cliente, Autorizado, Jogo, Álbum, Filme, Cópia de Jogo, Cópia de Álbum, Cópia de Filme, Ator, Cantor, Classe, Diretor, Fabricante de Jogo, Gênero, Endereço, Música, Faixa Etária e Idioma): divide-se em Pesquisa Detalhada e Pesquisa Rápida que são pesquisas realizadas sobre os dados.

- Listar Cadastros: listagem dos dados de algum dos cadastros.

- Fazer Cadastros: realizar cadastro sobre algum dos cadastros.

- Tratar Configurações: tratar das configurações da locadora, locações e reservas.

- Ver Relatórios: mostrar relatórios de pendências e histórico de locações, reservas e encomendas.

- Fazer Empréstimo: fazer empréstimo de cópias para um cliente ou autorizado.

- Fazer Devolução: fazer devolução de cópias de um empréstimo de um cliente ou autorizado.

- Emprestar Encomenda: realizar um empréstimo de uma encomenda feita por um cliente.

- Cancelar Reserva: cancelar uma reserva feita por um cliente.

3.3 Modelagem do banco de dados

Neste tópico são demonstradas graficamente as relações entre as entidades do banco e o significado de alguns dos campos mais importantes de cada entidade. As figuras foram geradas no DBDesigner 4.

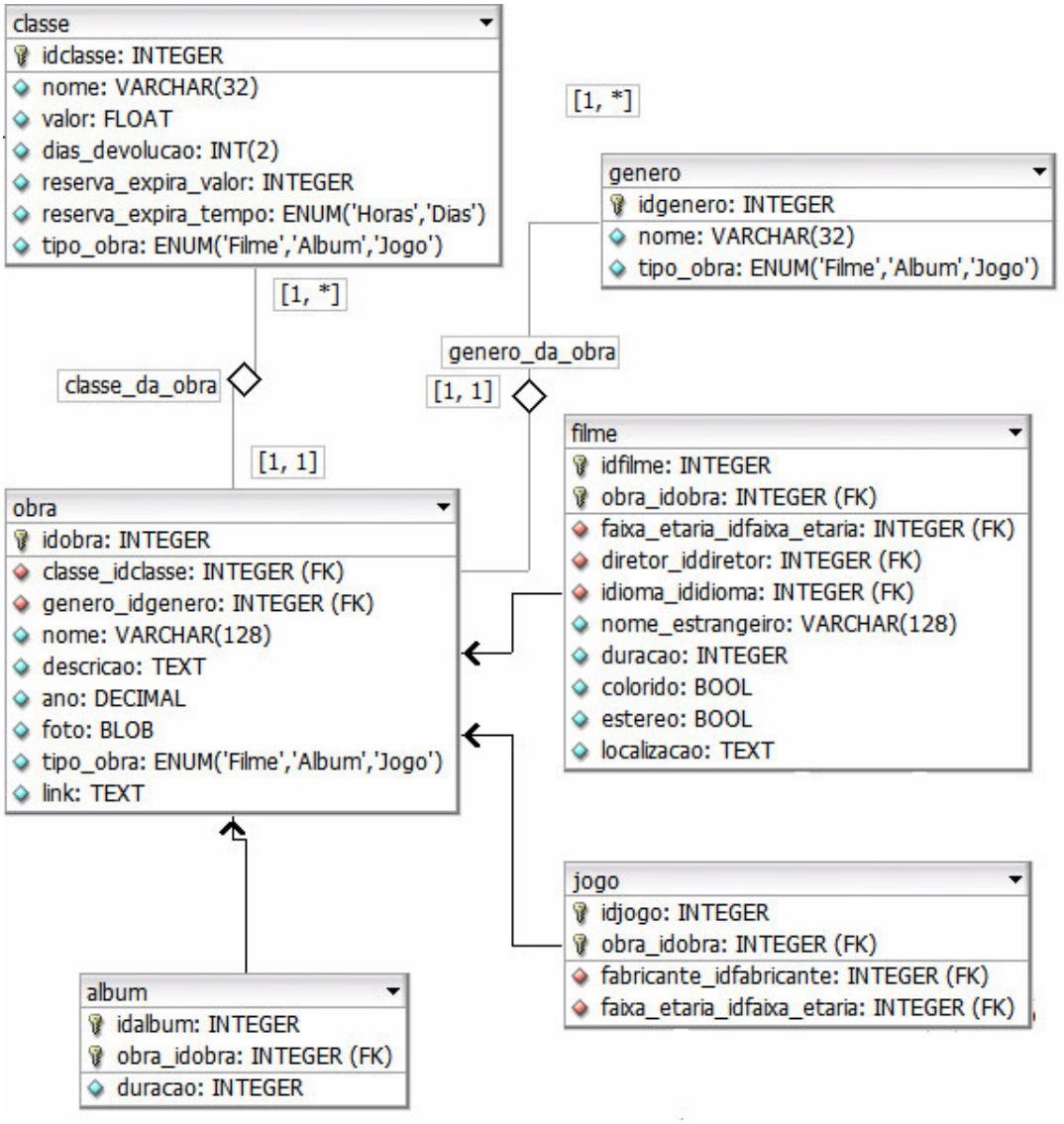

Figura 20 ER da entidade Obra

Como demonstrado na Figura 20 existe uma herança de filme, jogo e álbum com relação à obra, pois os dados da obra serão comuns a estas três entidades.

Uma obra possui um gênero, que depende do tipo da obra, ou seja, por exemplo, um gênero de filme é diferente de um gênero de álbum, a obra também possui uma classe que significa, se a obra é um lançamento, super-lançamento, etc. A Classe também depende do tipo da obra, é a classe quem determina qual é o valor do empréstimo da obra, em quantos dias uma obra deve ser devolvida e o tempo em que uma reserva expira, ou seja, quando um cliente da locadora reserva uma obra, ele tem um tempo para retirá-la, se este tempo expirar a obra não estará mais reservada.

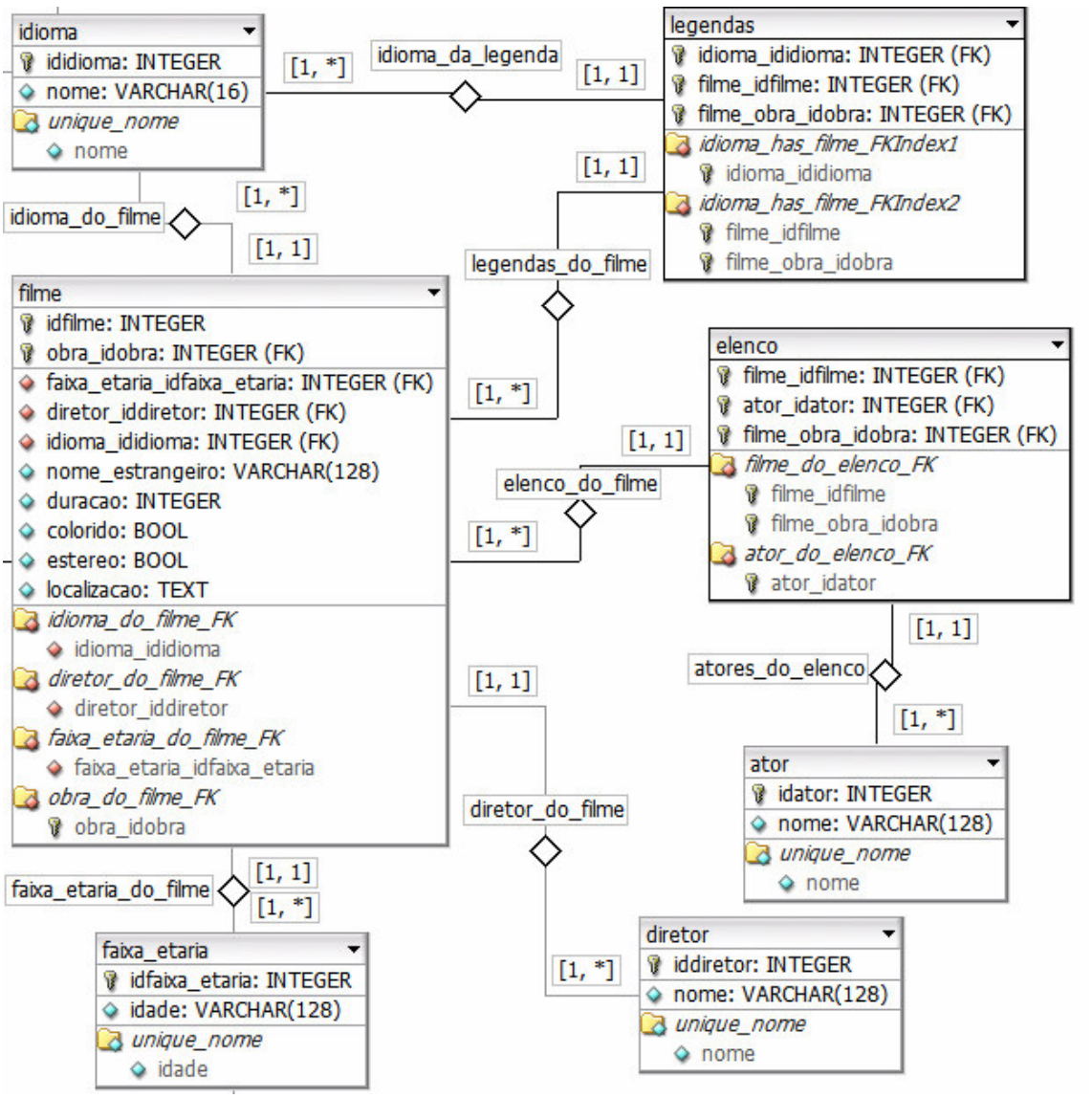

Figura 21 ER da entidade Filme

Um filme, que está sendo representado na Figura 21, além de herdar os campos da obra terá, ainda, uma faixa etária, um diretor, um elenco que é composto por vários atores, um idioma e várias legendas.

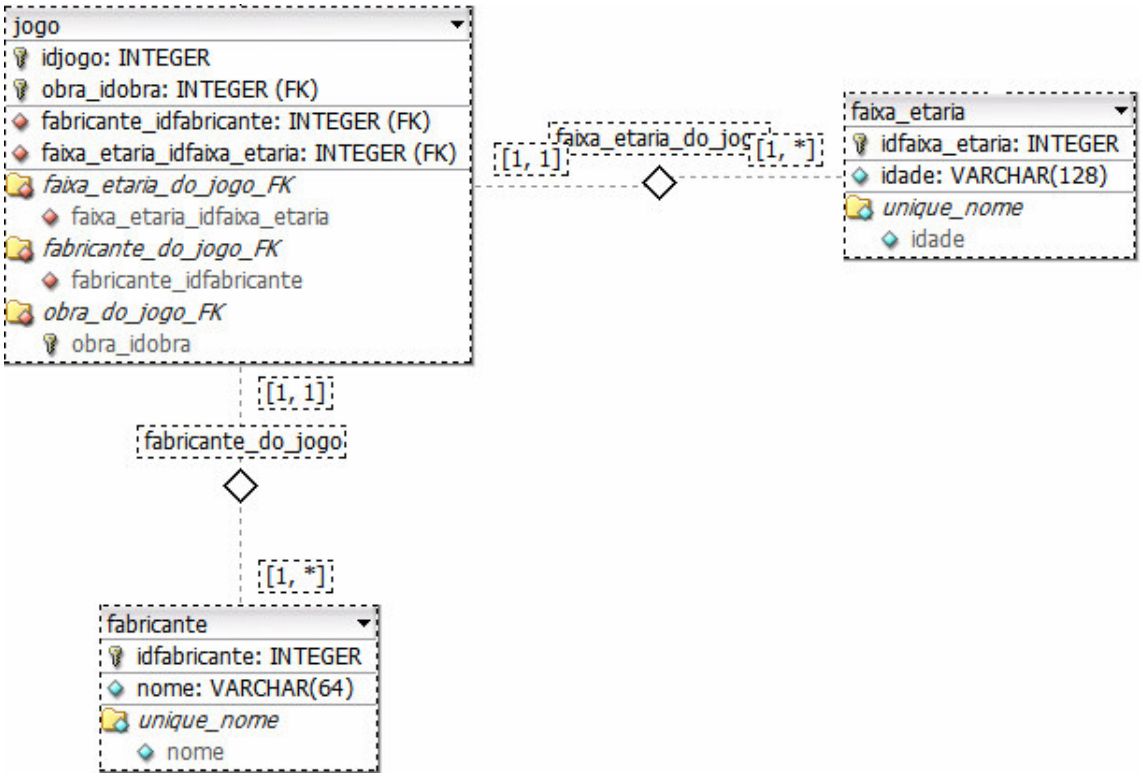

Figura 22 ER da entidade Jogo

Um jogo, que é mostrado na Figura 22, além de herdar a entidade Obra, ainda vai possuir uma faixa etária e um fabricante.

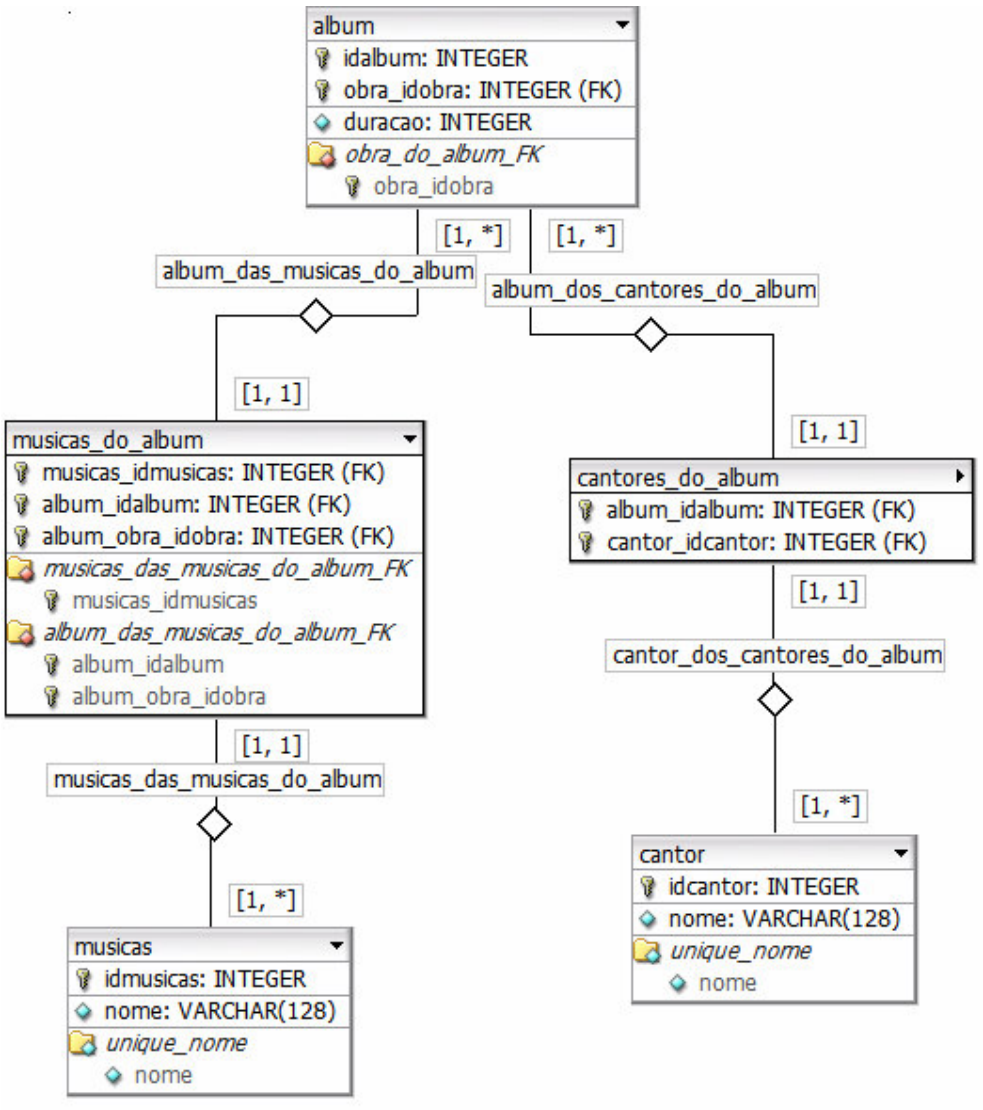

Figura 23 ER da entidade Álbum

A entidade álbum, com sua representação na Figura 23, herda os dados da Obra, e ainda pode possuir vários cantores e várias músicas.

Uma cópia é mostrada na Figura 24, refere-se ao produto propriamente dito, por isso cada copia faz referencia a uma obra e a uma mídia, por exemplo, um DVD, CD, etc.

Um cliente, representado na Figura 25, possuirá além dos seus dados um endereço. Este endereço possuirá um valor de entrega que é o custo que a locadora cobrará caso este cliente queira receber uma obra em sua casa. O endereço é uma entidade a parte, pois o valor de entrega pode ser um padrão para todas as pessoas

da mesma cidade, do mesmo bairro e da mesma rua. Autorizadas, são as pessoas que não necessitam ter um cadastro, mas podem retirar obras no nome do seu cliente.

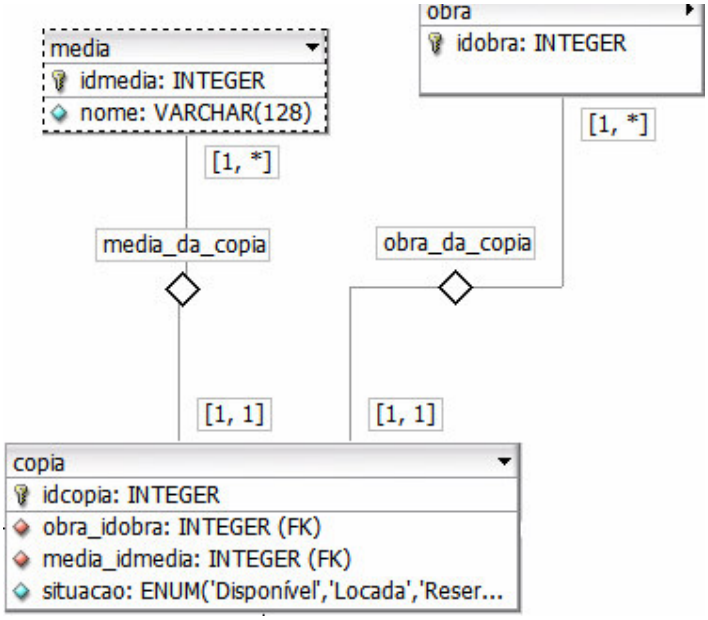

Figura 24 ER da entidade Cópia

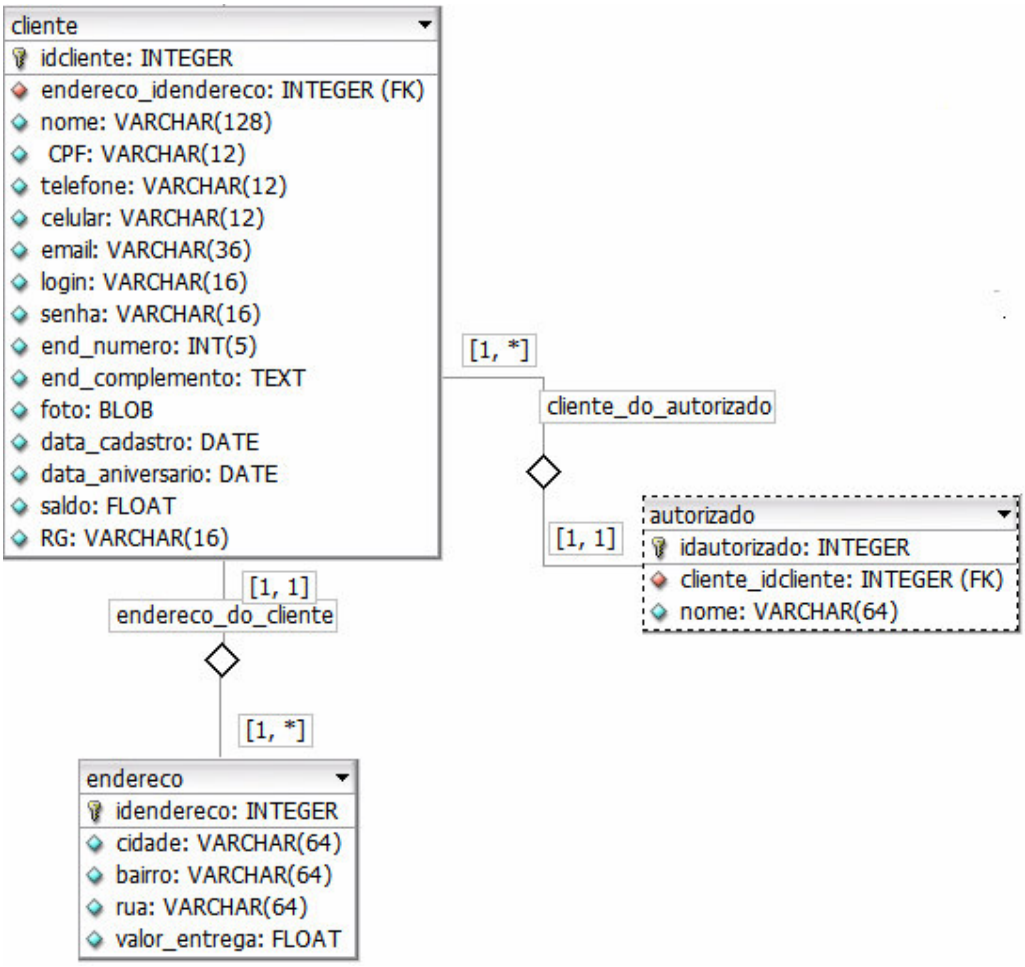

Figura 25 ER da entidade Cliente

O Empréstimo, mostrado na Figura 26, possui um cliente, pode possuir várias cópias e caso o empréstimo seja para um autorizado deve possuir uma referência para este. O empréstimo também pode ser referente a uma encomenda que foi feita pelo usuário.

Uma reserva possui um cliente e uma ou várias cópias, pode ser visualizada na Figura 27.

Uma encomenda, que é mostrada na Figura 28, possui um cliente e uma ou várias cópias. A encomenda também uma situação para informar se esta já foi emprestada para o cliente.

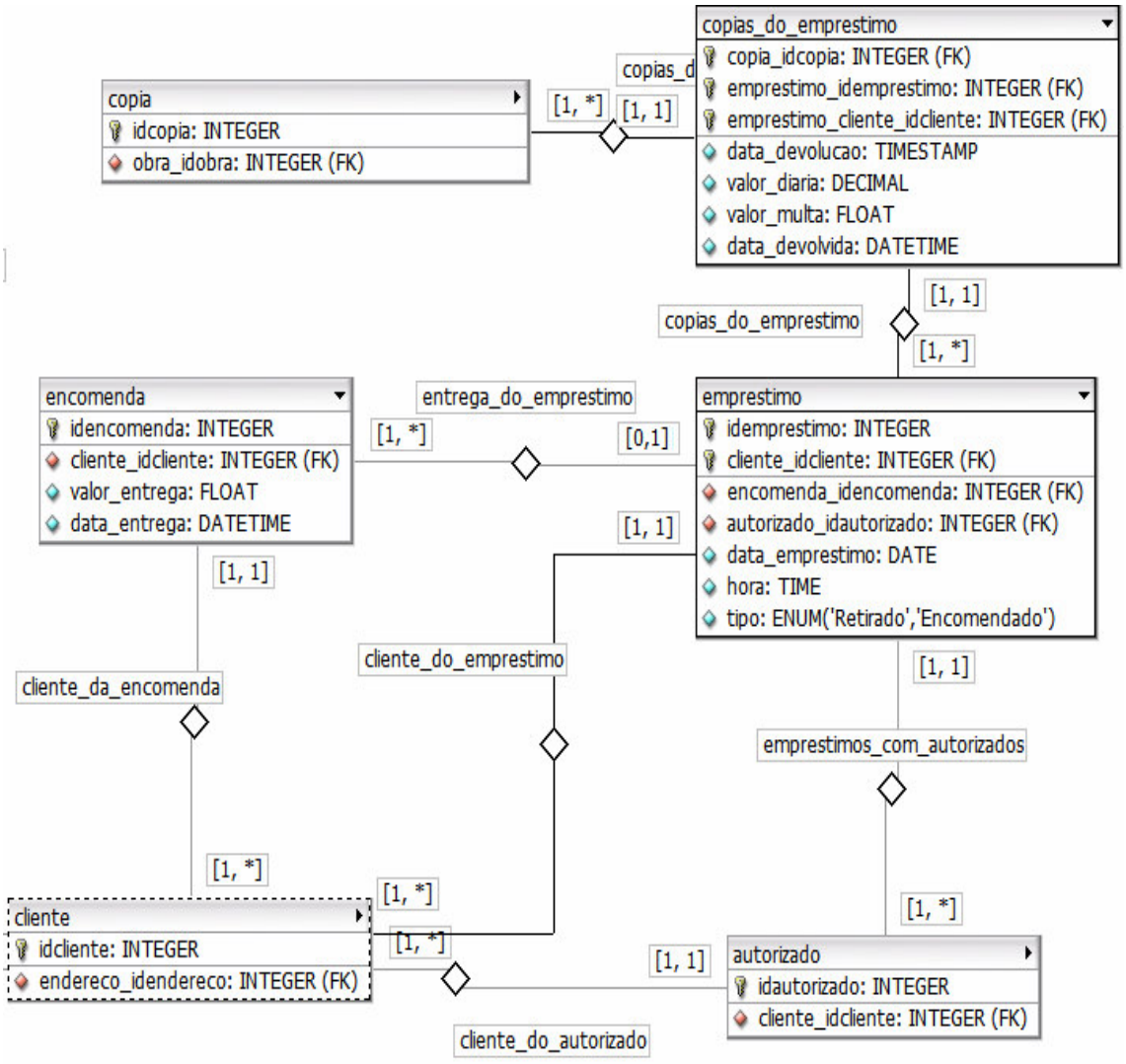

Figura 26 ER da entidade Empréstimo

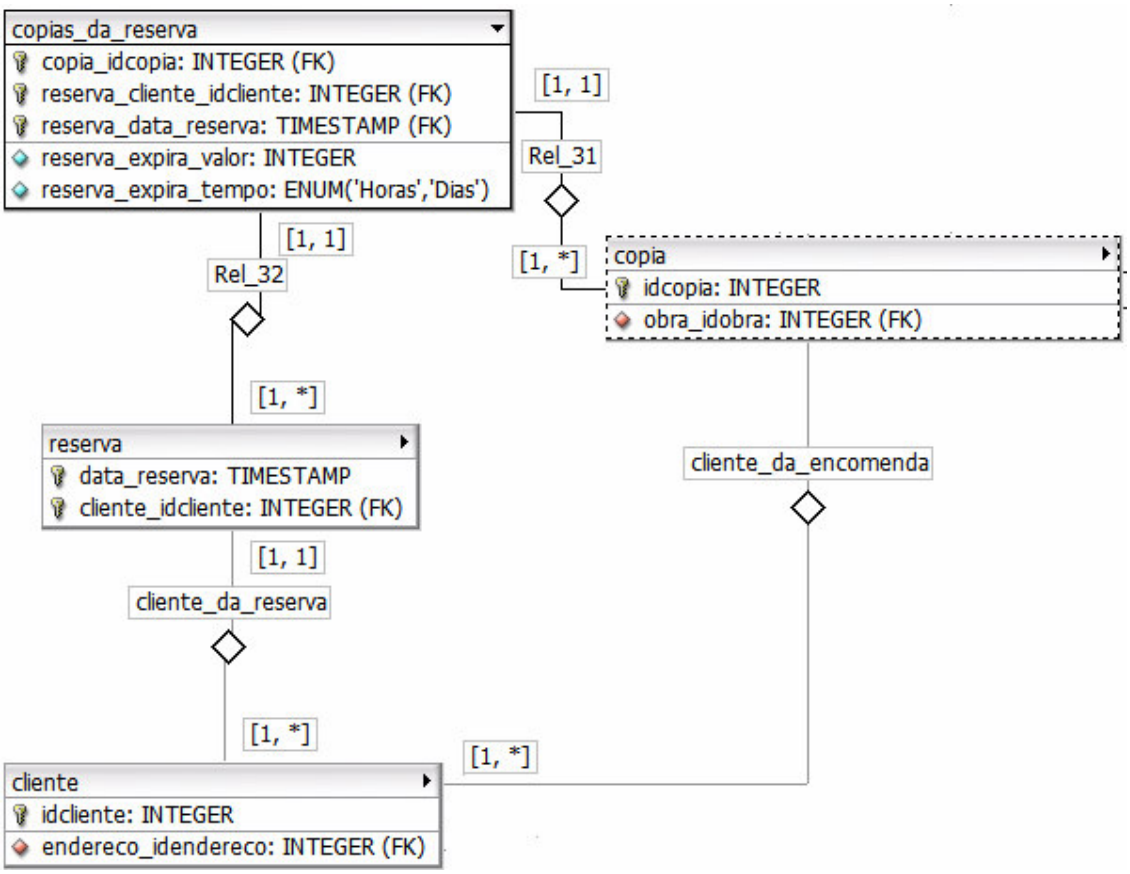

Figura 27 ER da entidade Reserva

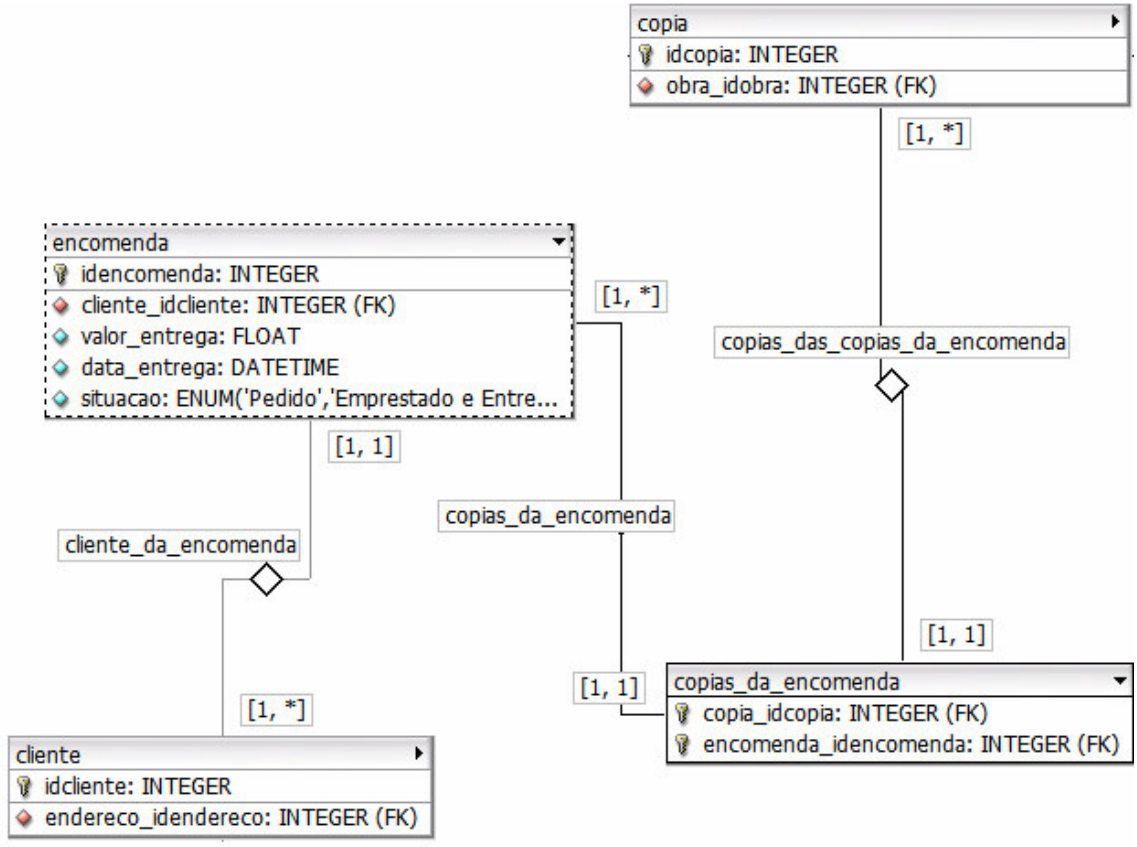

Figura 28 ER da entidade Encomenda

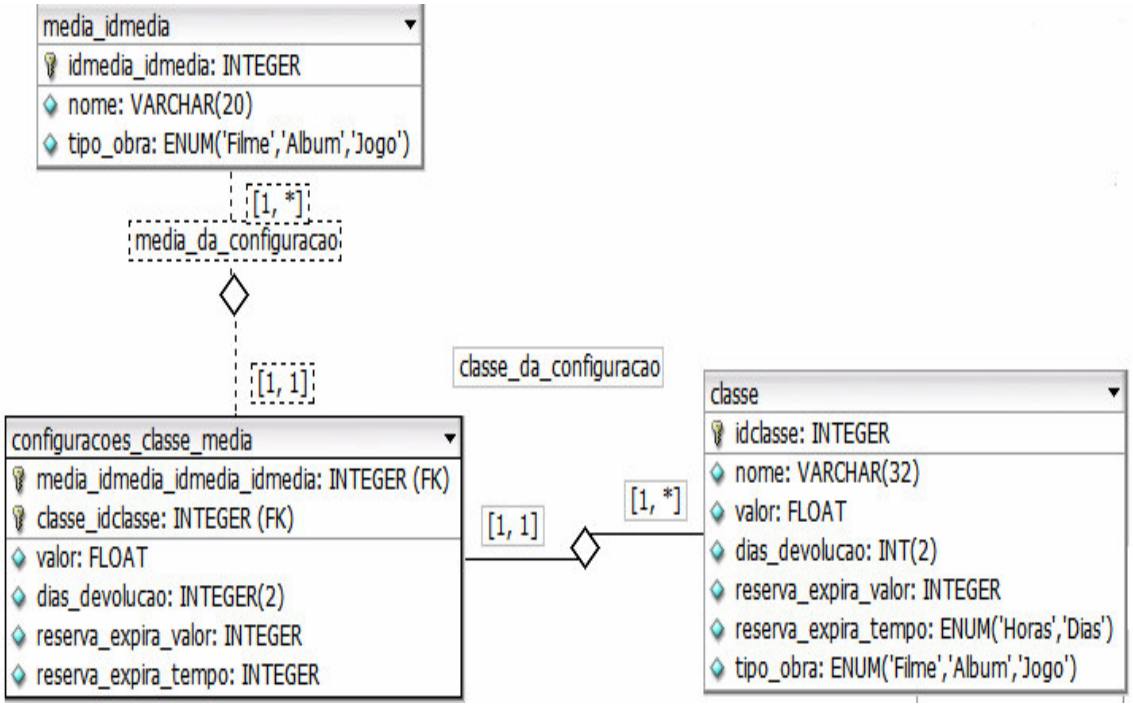

Figura 29. Tabela de Configuração da classe com a mídia

A Figura 29 consiste nas configurações de preços, dias de devolução e dados de tempo de validade de reserva da relação entre uma classe e uma mídia. Ou seja, o funcionário quem configura o preço de uma cópia, conforme a sua classe e sua mídia.

Existem mais algumas tabelas no banco de dados, como, administrador que guarda os funcionários da locadora, tabela de configuração das permissões de acesso, para definir as partes do sistema que cada tipo de usuário, cliente ou funcionário, poderá ter acesso, e mais algumas tabelas auxiliares.

3.4 Implementação

Neste tópico serão descritos exemplos de como foi implementado o sistema usando as tecnologias AJAX e orientação a objetos baseado em MVC.

3.4.1 O uso de orientação a objetos com MVC no sistema da locadora virtual

Para exemplificar o uso de orientação a objetos com as estruturas MVC, está sendo usado o cadastro de gênero. A Figura 30 representa a classe de controle MVC da entidade gênero, onde são tratadas algumas das requisições. Os testes ilustrados na Figura 30 são responsáveis respectivamente por mostrar o formulário de cadastro de gêneros (método visual_cadastrar) e por gravar os dados enviados do formulário no banco de dados (método cadastrar).

```php
<?include "genero.php";
include "genero_visual.php";
class genero_controle
{
    var $genero;
    var $genero_visual;
    function genero_controle()
    {
        if($_GET["acao"]=="visual_cadastrar")
        {
            $this->genero_visual = new genero_visual();
            $this->genero_visual->visual_cadastrar();
        }
        if($_GET["acao"]=="cadastrar")
        {
            if(!$_GET["nome"])
            {
                    msg_erro("O Nome do Gênero não pode ficar em branco");
                    $this->genero_visual = new genero_visual();
                    $this->genero_visual->visual_cadastrar();
            }else{
                $ok = mysql_query("insert into genero(nome,tipo_obra)
             values('$_GET[nome]','$_GET[tipo_obra]')");

            if($ok) msg_cadastrar_ok();
            else msg_cadastrar_erro();

            }
        }
    }
}?>
```

Figura 30 MVC: Exemplo de Controle para a entidade gênero

A Figura 31 mostra o Modelo MVC da entidade gênero com seus atributos que serão utilizados pelo Visual e pelo Controle.

```php
<?class genero
{
    var $campos;
    var $campos_selecionados;
    var $titulo_cabecalho;
    var $titulo_listagem;
    var $classe;
    var $ID_SQL;
    var $orelhas;
    function genero()
    {

        $this->campos["nome"][0] = "nome";
        $this->campos["titulo"][0] = "Nome";
        $this->campos["nome"][1] = "tipo_obra";
        $this->campos["titulo"][1] = "Tipo";
        $this->campos_selecionados[0] = "nome";
        $this->campos_selecionados[1] = "tipo_obra";
        $this->titulo_cabecalho = "GÊNERO";
        $this->titulo_listagem = "Gêneros";
        $this->classe = "genero";
        $this->ID_SQL = "generos";
        $this->orelhas["URL"][0]="trata_requisicoes.php?
        func=$this->classe&acao=visual_cadastrar";
        $this->orelhas["titulo"][0]="Cadastrar";
        $this->orelhas["titulo"][1]="Procurar";
        $this->orelhas["titulo"][2]="Listar";
    }
}
?>
```

Figura 31 MVC: Exemplo de Modelo para a entidade gênero

A Figura 32 mostra o visual da estrutura MVC para a entidade gênero, onde estão os métodos ou objetos que irão mostrar os dados na tela, como, cadastros (método visual_cadastrar), lista de gêneros (objeto listagem, chama uma objeto padrão para desenhar uma tabela de listagem) e busca por gênero (objeto procurar, chama um objeto padrão para mostrar o formulário de busca).

```
<?class genero_visual
{
    var $genero;
    var $listagem_controle;
    var $pesquisa_controle;
    function genero_visual()
    {
        $this->genero = new genero();
        $this->listagem_controle = new listagem_controle($this);
        $this->pesquisa_controle = new pesquisa_controle($this);
    }

    function visual_cadastrar()
    {?>
    <table>
    <form action="trata_requisicao.php" method="GET">
    <tr><td class="fonte_normal">Tipo de Obra:</td>
    <td><select name="tipo_obra" class="form">
    <option value="Filme">Filme</option>
    <option value="Album">Album</option>
    <option value="Jogo">Jogo</option>
    </select></td></tr>
    <tr><td class="fonte_normal">Nome:</td><td>
    <input type="text" name="nome" class="form" size=30>
    </td></tr>
    </table>
    <input type="button"  class="enviar" value="Enviar Cadastro"
    onclick="document.getElementById('espaco_central').innerHTML=requisicao('trata_requisicoes.php?
     func=genero&acao=cadastrar&nome='+document.getElementById('nome').value+'&tipo_obra='+
      document.getElementById('tipo_obra').value,false,'espaco_central');">

    </form>
  <?}
}
?>
```

Figura 32 MVC: Exemplo de Visual para a entidade gênero

3.4.2 O uso de AJAX no sistema da locadora virtual

O trecho de código destacado na Figura 32, exemplifica o uso de AJAX, este trecho significa que ao clicar em “Enviar Cadastro”, é chamada a função responsável por enviar os dados, que é a função “requisicao (URL, sincr, obj)”, esta faz a requisição de uma URL através do objeto XMLHTTPRequest e retorna os dados em HTML. A URL especificada no trecho acima irá enviar a ação cadastrar com os dados do gênero, isso será tratado pela classe “genero_controle” dentro do teste “$_GET [acao]==cadastrar”, que é onde o gênero é gravado no banco de dados. O resultado da requisição é retornado em HTML, por isso deve ser gravado em algum lugar pré-definido, este lugar é a parte central da página, a qual se pode acessar

usando a expressão Javascript document.getElementById (‘espaco_central’).innerHTML, sendo que document.getElementById (‘espaco_central’) é um objeto DOM que já estava especificado anteriormente e só serve para demarcar a área central da página. Com isso, o cadastro é enviado e a única parte de código que muda é a parte central, aonde virão os novos dados, nas figuras, Figura 33 e Figura 34, são mostradas as telas geradas pelo cadastro de gênero antes e depois de ser enviado, a parte destacada nas figuras é o única que é recarregada.

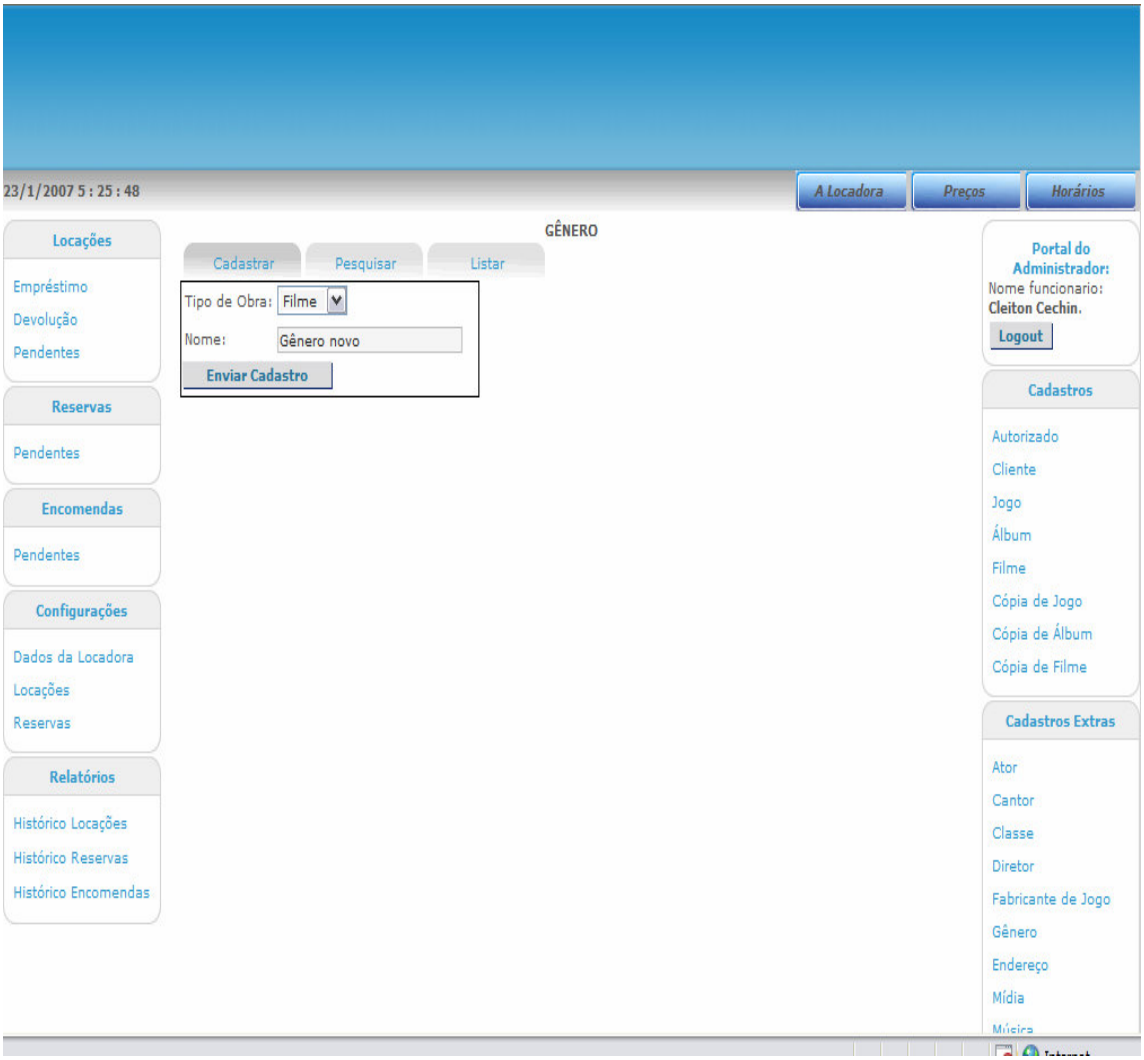

Figura 33 Tela de cadastro de gênero

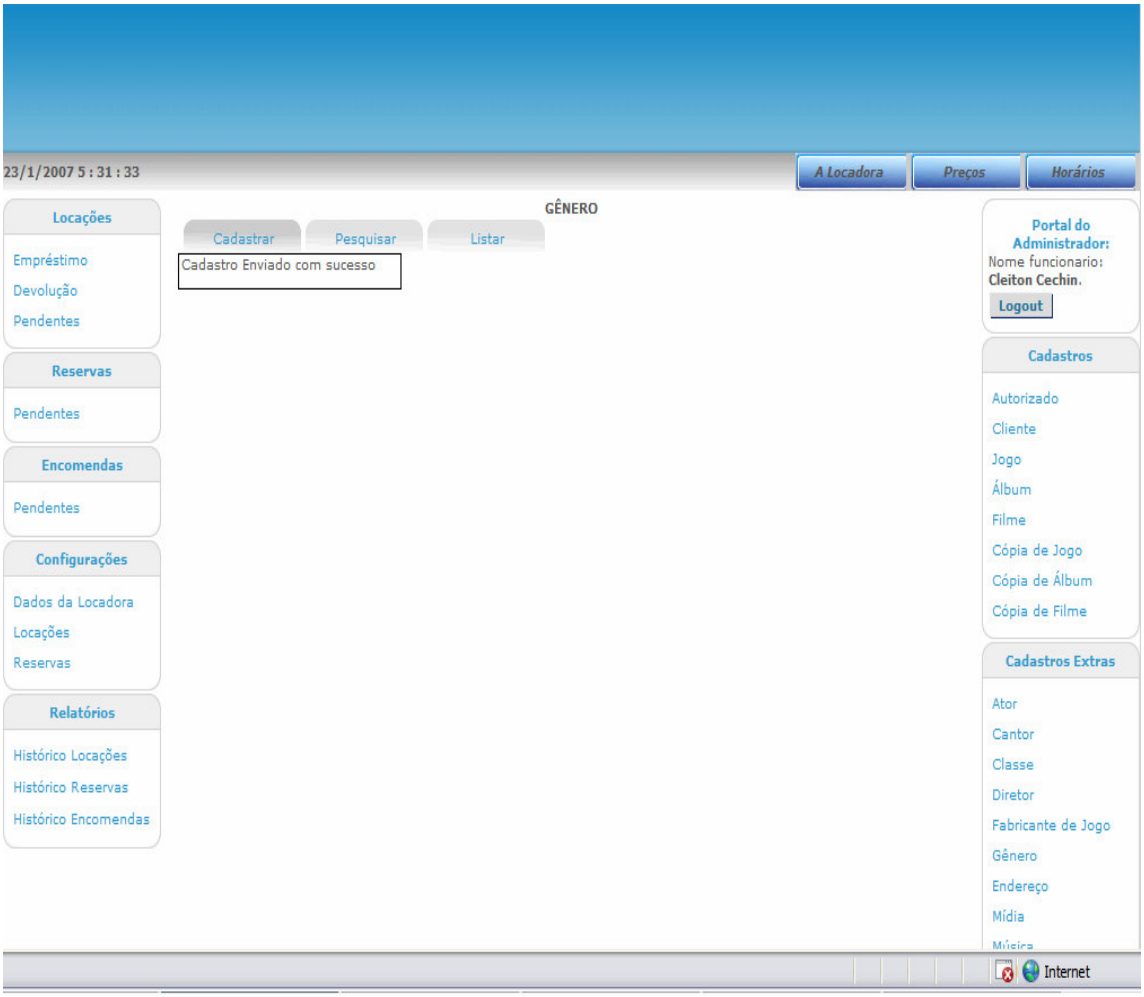

Figura 34 Tela de cadastro de gênero após envio de cadastro

Observa-se que no exemplo citado acima, as requisições retornam os dados em HTML. Uma parte do sistema onde são retornados os dados em XML é das telas de listagem padrão. O sistema de listagem funciona da seguinte maneira: o funcionário marca os campos para serem mostrados na tabela de listagem. Ao fazer isso, é enviada uma nova requisição usando AJAX, quando são buscados os dados em XML do campo marcado pelo usuário e inseridos na coluna respectiva ao campo da tabela, sem a necessidade de modificar as outras colunas já existentes na lista.

A Figura 35 é a listagem de gêneros apenas com o nome sendo selecionado pelo usuário.

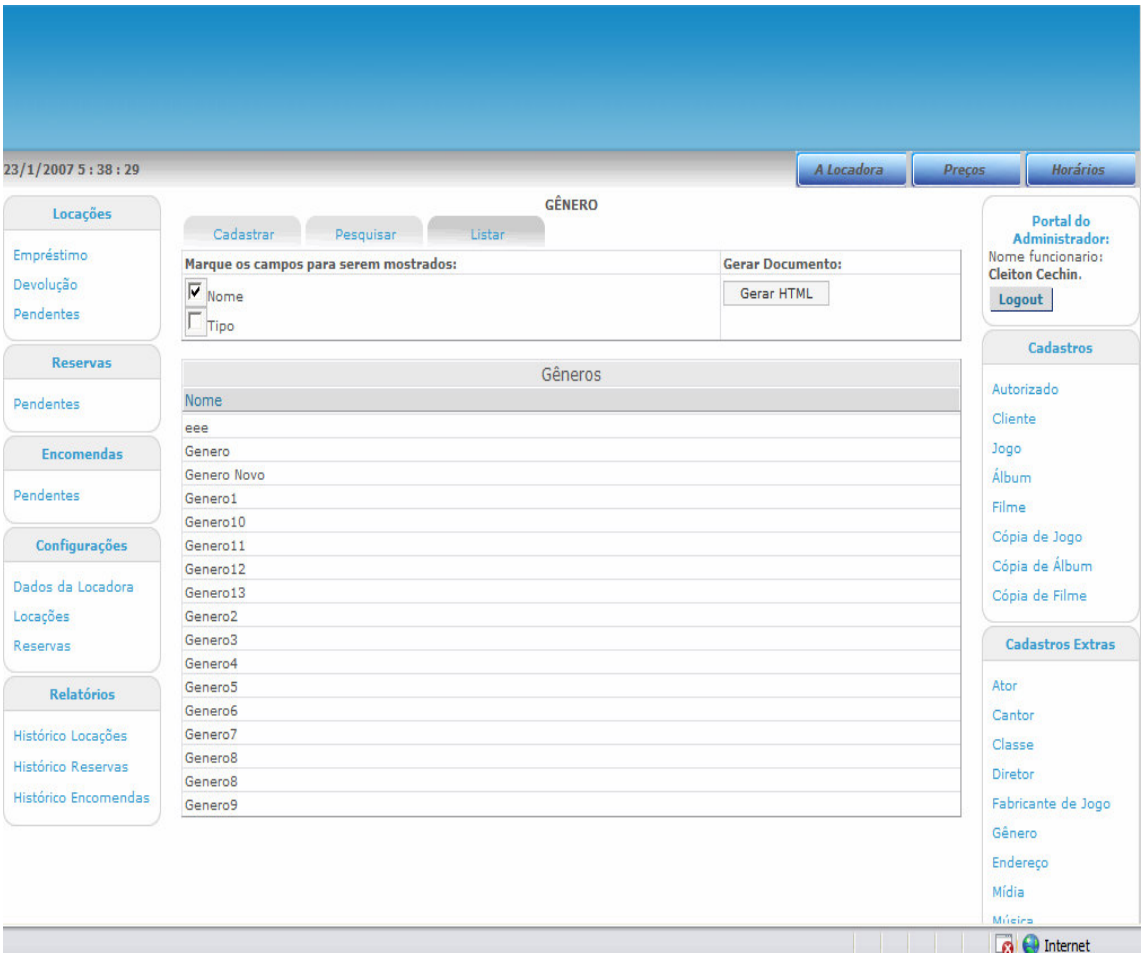

Figura 35 Listagem de gênero com o campo nome selecionado

Na Figura 36 é visualizada a nova listagem de gêneros após o novo campo –tipo- ter sido selecionado pelo usuário, seus dados serem retornados em XML e inseridos na sua coluna em tempo real pelo AJAX.

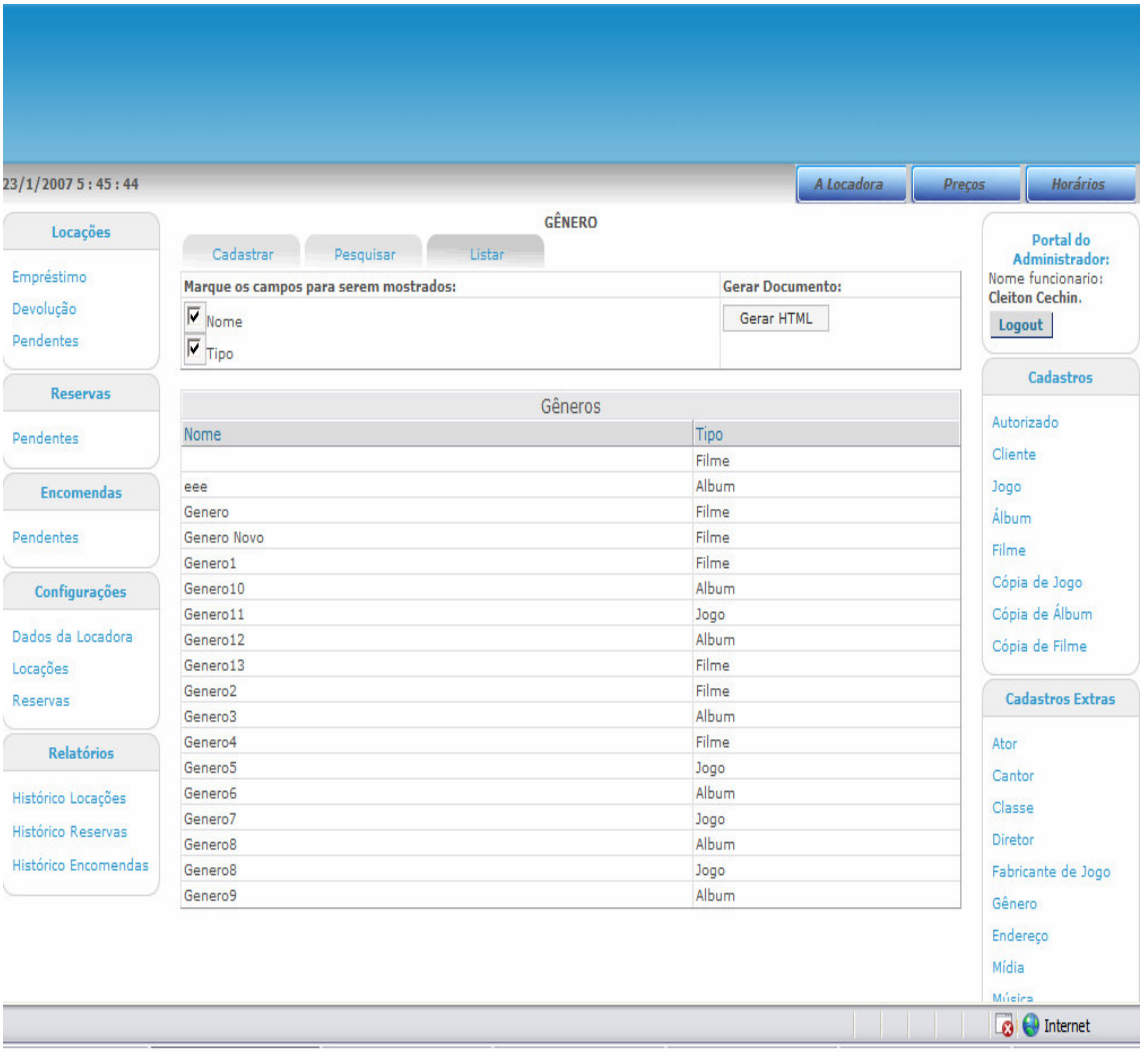

Figura 36 Listagem de gênero após o campo tipo ser selecionado

4 RESULTADOS

Este capítulo mostra alguns resultados do uso da tecnologia orientada a objetos baseada em MVC e da comunicação com AJAX usados no estudo de caso da locadora virtual implementado.

4.1 Orientação a objetos baseada em MVC

Com o uso desta forma de estruturar e modelar o sistema, este ficou organizado em módulos, pois, desde a fase de modelagem do banco de dados já foi projetado com o pensamento na implementação orientada a objetos, todas as partes do sistema foram organizadas em entidades (objetos) que implementam suas funções especificas.

Sendo assim cada cadastro, relatório, listagens e buscas, têm seus objetos próprios que implementam apenas as suas funções especificas independente dos outros. Como exemplo disso está mostrado na Figura 37 os objetos de Filme, Álbum e Jogo, herdando as propriedades e métodos de uma de uma obra. Os objetos e métodos herdados demonstrados na Figura 37 são, respectivamente, listagem e cadastrar. Observa-se que estes também estão presentes no Filme, Jogo e Álbum, isto porque estes devem cadastrar e listar somente seus atributos próprios e chamar os métodos herdados para cadastrar e listar os atributos herdados da obra.

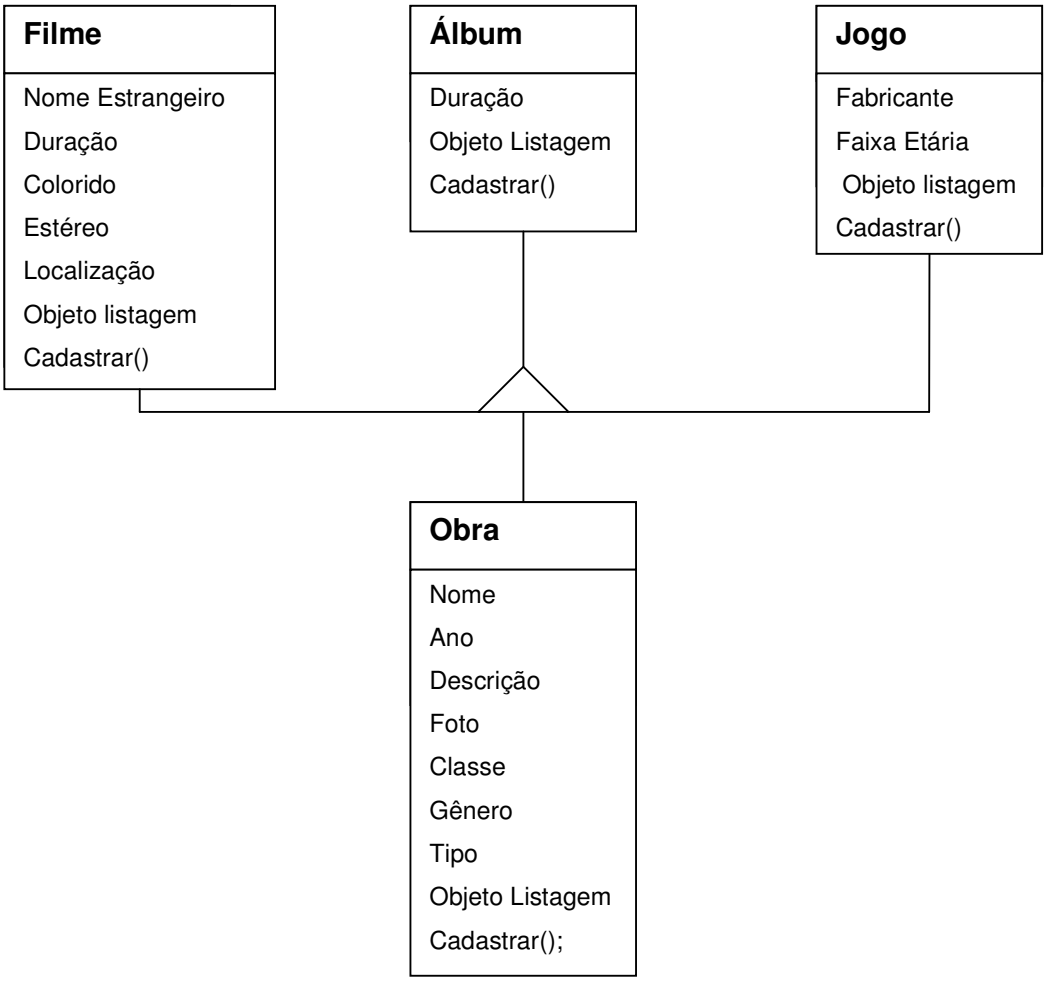

Figura 37 Filme, álbum e jogo herdando propriedades de Obra

Todas as entidades do sistema também ficaram organizadas conforme os princípios estruturais da orientação a objetos com MVC. Com isso, cada entidade do sistema possui sua classe modelo, controle e visual.

Na Figura 38 está sendo demonstrada à organização da entidade obra, com suas classes controle, modelo e visual. Observa-se que a classe visual da obra faz referência à entidade listagem, que é a listagem padrão para todos os objetos, que também é estruturada na forma MVC.

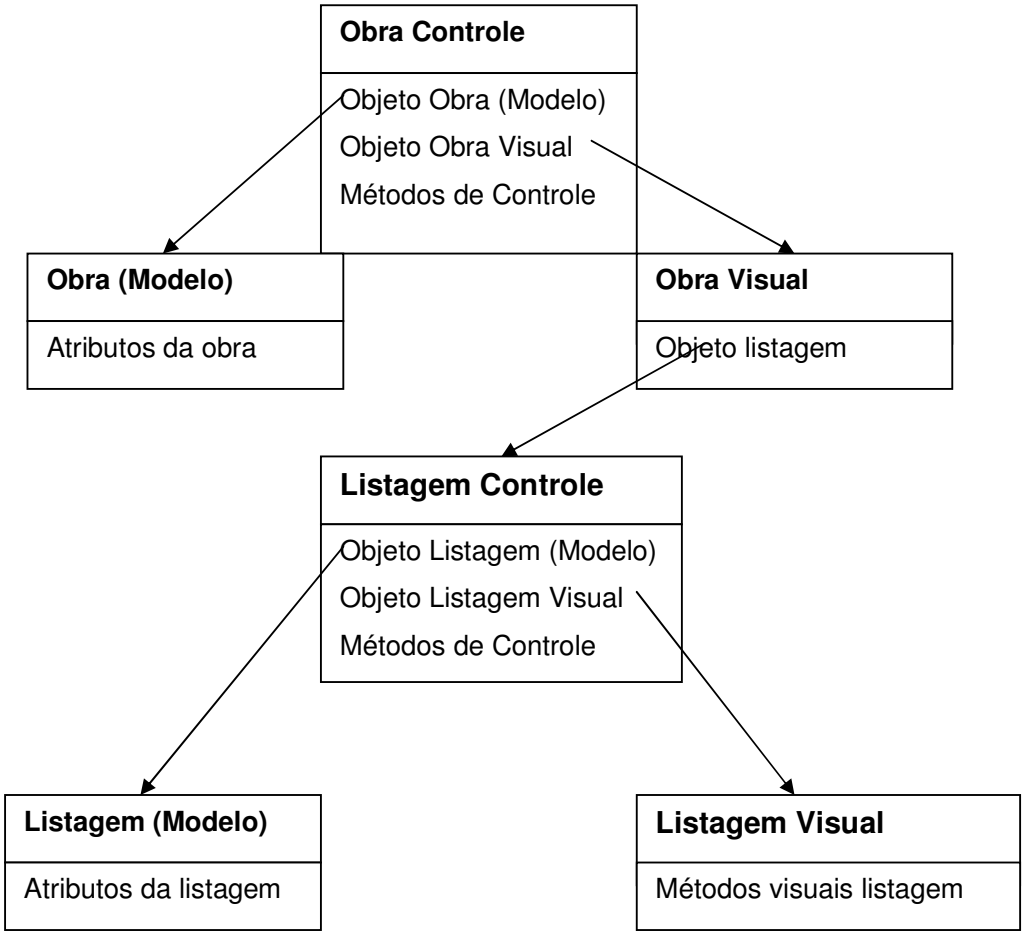

Figura 38 MVC da entidade obra

Todas as outras entidades do sistema estão organizadas da mesma forma, ou seja, cada uma destas é responsável somente e especificamente por suas funções. Cada entidade está dividida em 3 camadas: visual, controle e modelo, onde cada uma destas tem suas funcionalidades especificas para poder realizar em conjunto todas as funções de responsabilidade de sua entidade.

Com isso o sistema ficou de fácil entendimento e modificações, pois, ao ser necessário a inserção de uma nova funcionalidade, apenas será criada uma nova entidade. Se for necessária a modificação do sistema é preciso somente modificar as classes MVC da entidade responsável.

4.2 AJAX

Com a implementação da troca de dados entre cliente e servidor baseada em AJAX, todo o sistema está consumindo apenas a banda necessária a cada requisição, ou seja, somente os dados necessários para obtenção do resultado desejado são buscados no servidor. Na primeira requisição feita ao sistema, toda a lógica AJAX é carregada, e nas próximas requisições, apenas as partes necessárias do site estão sendo redesenhadas.

Como exemplos, são citadas as tabelas de listagens padronizadas, ilustradas nas Figura 35 e Figura 36. Se estas fossem implementadas da maneira convencional de programação WEB, todos os dados da tabela listagem, além de toda a página, seriam carregados novamente a cada pedido de novo campo pelo usuário. Com o uso de AJAX, apenas os dados necessários de cada novo campo selecionado são buscados no servidor. Com isso, os dados da tabela que estavam sendo mostrados não necessitam ser buscados novamente.

5 CONCLUSÕES

O presente trabalho de graduação consistiu no estudo do uso de duas novas tecnologias de programação para *WEB,* o paradigma de orientação a objetos com MVC e a comunicação baseada em AJAX. Como estudo de caso sobre estas tecnologias foi implementado um sistema de locadora virtual.

Com base na implementação do sistema da locadora virtual com as duas tecnologias estudadas, orientação a objetos com MVC e comunicação AJAX, ficaram confirmadas as características de cada uma delas. A primeira aumenta o poder de compreensão da implementação e organização do sistema. A outra consegue minimizar a quantidade de dados trocados entre o cliente e o servidor, tornando os *sites WEB* mais rápidos e acessíveis.

Com a implementação de um sistema baseado na metodologia de programação orientada a objetos, esse fica dividido em módulos que se relacionam para formar o sistema como um todo, facilitando sua compreensão e funcionamento, pois, cada um destes módulos tem funções específicas para o conjunto da implementação. O sistema fica ainda de mais fácil entendimento quando é usada à estrutura baseada em MVC, pois com esta forma de organizar os módulos, podem-se dividi-los em 3 partes de uma maneira simples: as partes visuais, os atributos (modelo) e os controles de cada entidade, facilitando com isso a implementação e a modificação das funcionalidades de cada objeto.

Com o uso da tecnologia AJAX para o tratamento das requisições, é possível diminuir consideravelmente a troca de dados desnecessários entre o cliente e servidor, deixando o site com o máximo possível de rapidez de acesso, pois somente são retornados os dados necessários a cada requisição.

O sistema de locadora virtual ainda não está totalmente finalizado, para que este possa se usado em um estabelecimento necessita algumas atualizações, como: melhoria de layout, adição de um sistema de ajuda para todas as telas e a implementação de uma maneira de editar todos os cadastros. Outras funções que

podem ser adicionadas ao sistema é a geração dos relatórios em PDF, pois atualmente estes somente podem ser gerados em HTML, e a elaboração de um sistema de bônus para o cliente, em que o próprio funcionário da locadora possa configurá-lo.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Wikipédia, a enciclopédia livre. **AJAX (programação)**. Disponível em http://pt.wikipedia.org/wiki/AJAX. Acessado em 10 de outubro de 2006.

[2] Garret, James Jessé, **AJAX: A New Approach to *Web* Applications**. Disponível em http://www.adaptivepath.com/publications/essays/archives/000385.php. Acessado em 10 de outubro de 2006.

[3] Apple Developer Connection. **Dynamic HTML and XML: The XMLHttpRequest Object**. Disponível em http://developer.apple.com/internet/*web*content/xmlhttpreq.html. Acessado em 10 de outubro de 2006.

[4] Ruiz, José Maria; Orantes, Pedro. **AJAX A nova tecnologia da *Web***. Revista Linux Magazine. P 73-78. Setembro 2006.

[5] Ricarte, Ivan Luiz Marques. **Programação Orientada a Objetos em C++**. Disponível em http://www.dca.fee.unicamp.br/cursos/POOCPP/POOCPP.html. Acessado em 20 de dezembro de 2006

[6] Berg, Cruz Alexandre; Figueiró, Joice Pavek. **Lógica de Programação**. Ed. Editora da Ulbra.1996.

[7] AMTECHS, W3C Português. **Modelo de Objeto de Documentos DOM – Nível 1**. Disponível em http://www.amtechs.com/w3c/rec-dom-level-1.html. Acessado em 25 de outubro de 2006

[8] AMTECHS, W3C Português**. Modelo de Objeto de Documentos DOM – Especificação HTML Nível 2**. Disponível em http://www.amtechs.com/w3c/rec-dom-level2/. Acessado em 25 de outubro de 2006.

[9] W3C**. Document Object Model,W3C**. Disponível em http://www.w3.org/DOM/. Acessado em 26 de outubro de 2006

[10] ALECRIM, Emerson. **Linguagem XML**. Disponível em http://www.infowester.com/lingxml.php. InfoWester, 03 de janeiro de 2001. Acessado em 20 de outubro de 2006.

[11] ALVAREZ, Miguel A. **O que é PHP**. Disponível em http://www.criarweb.com/artigos/202.php. <criarweb>. Acessado em 20 de outubro de 2006

[12] Wikipédia, a enciclopéia livre. **MVC**. Disponível em http://pt.wikipedia.org/wiki/MVC. Acessado em 13 de dezembro de 2006.

[13] **Site oficial MySQL**. Disponível em http://mysql.com/.

[14] **Manual Completo da linguagem PHP**. Disponível em http://www.php.net/manual/pt_BR/.

[15] **Site oficial da ferramenta DBDesigner 4**.Disponível em http://fabforce.net/dbdesigner4/

[16] Wikipédia, a enciclopéia livre. **Javascript**. Disponível em http://pt.wikipedia.org/wiki/Javascript. Acessado em 13 de dezembro de 2006

[17] Wikipédia, a enciclopédia livre. **XML**. Disponível em http://pt.wikipedia.org/wiki/XML. Acessado em 13 de dezembro de 2006

[18] INTERney. **O que é HTML? Para que serve?** Disponível em http://www.interney.net/blogfaq.php?p=6541494. Acessado em 11 de outubro de 2006.

[19] AMTECHS, W3C Português**. Transformações XSL(XSLT)**. Disponível em http://www.amtechs.com/w3c/rec-xslt-19991116.html. Acessado em 15 de dezembro de 2006

[20] **Manual CSS desenho programação**. Disponível em http://www.criarweb.com/css/. Acessado em 13 de dezembro de 2006

[21] Cavalcanti, Marcus. **Arquitetura MVC**. Disponível em http://www.revistaphp.com.br/artigo.php?id=50. Acessado em 5 de janeiro de 2007

[22] Wikipédia, a enciclopédia livre. **MySQL**. Disponível em http://pt.wikipedia.org/wiki/MySQL. Acessado em 10 de janeiro de 2007

[23] Costa, Henry Franklin Dualibe. **MySQL**.Disponível em http://www.henry.eti.br/pagina.php?idPagina=127. Acessado em 10 de janeiro de 2007

[24] Cunningham, Larkin. **MySQL truque com dados**. Revista Linux Magazine. P 36-42. Setembro 2006.

[25] Dicas- L. **DBDesigner - Software livre para modelagem de dados.** Disponível em http://www.dicas-l.com.br/dicas-l/20030922.php. Acessado em 10 de janeiro de 2007.

[26] Info Online. **DBDesigner**. Disponível em http://info.abril.com.br/download/.shtml. Acessado em 10 de janeiro de 2007.

[27] Neto, Almir. **Desenvolvimento em três camadas usando MVC e PHP 5**. Disponível em http://fgsl.aslgo.org.br/fgsl3/images/atividades/mvc-php5.pdf. . Acessado em 13 de dezembro de 2006.