UNICEUB – CENTRO UNIVERSITÁRIO DE BRASÍLIA

FATECS – FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS

CURSO DE ENGENHARIA DA COMPUTAÇÃO

ALAN RICARDO SARAIVA MARTINS

TOTEM INTERATIVO

BRASÍLIA-DF

2° SEMESTRE DE 2008

ALAN RICARDO SARAIVA MARTINS

TOTEM INTERATIVO

Monografia de conclusão apresentada à
Faculdade de Tecnologia e Ciências Sociais
Aplicadas, do Centro Universitário de Brasília
UniCeub, curso de Engenharia da Computação.

Orientador: M.C. Professor Claudio Penedo

BRASÍLIA-DF

2° SEMESTRE DE 2008

Agradecimento

Agradeço a Deus pela sabedoria, persistência e por ter chegado até aqui.
Agradeço a minha amada noiva, Cynthia, pela atenção e paciência.

Agradeço a minha mãe, Emília, pelo suporte oferecido durante toda essa jornada e principalmente porque sem ela não estaria aqui.

Agradeço ao meu orientador, M.C. Claudio Penedo, pela atenção e suporte no desenvolvimento deste projeto. Ao coordenador do curso de Engenharia de Computação, professor Abiezer Amarília Fernandes, pela seriedade e profissionalismo. Ao professor Francisco Javier De Obaldia, por acreditar e incentivar o desenvolvimento deste trabalho. E aos demais professores que contribuíram com esta jornada, ampliando a minha visão sobre o mundo, reforçando a idéia que o caminho correto é o da humildade e perseverança.

Agradeço aos meus colegas do trabalho pela atenção e tempo dispensados.

E a todos que de alguma forma me ajudaram com este trabalho.

“Às vezes, quando tudo dá errado
acontecem coisas tão maravilhosas
que jamais teriam acontecido
se tudo tivesse dado certo.”

Resumo

O Totem Interativo é um protótipo desenvolvido com equipamentos de hardware habitualmente usados no dia-a-dia das pessoas, e que tem a função de oferecer auxilio na localização em shopping centers, verificação de conteúdos relacionados, controle de avisos internos, interação dos usuários com os horários dos cinemas e visualização de trailers dos filmes em cartaz. Essas funcionalidades são disponibilizadas por um software desenvolvido para a Web na linguagem PHP, de acordo com os conceitos de MVC, fazendo uso da Zend Framework e especificado exclusivamente para esse fim conforme os padrões da UML. Acoplado a uma tela Touchscreen, possibilita a interatividade do usuário com o protótipo de uma forma simples, rápida e clara.

Neste trabalho desenvolve-se um totem com a motivação de difundir a tecnologia no cotidiano das pessoas e incluir um meio de publicidade em grandes centros comerciais, abrindo portas para exploração pelos usuários.

Palavras Chave: Totem Interativo. PHP. Touchscreen.

Abstract

Totem interactive is a prototype developed with equipments of hardware normally used in people's daily life, and it has the function of helping find the indicates in the shopping centers, verification of related content, control of the internal warnings, interaction of the movies schedule and movies visualization. These functions are available by software to Web in the PHP language in the accordance of the concepts of MVC, using Zend Framework and exclusively for this purpose according the patterns of UML, attached to a screen Touchscreen that enables the user's interactivity with the prototype on a simple, fast and clear way.

This Totem was developed for spread the technology in the daily lifes of people, including a kind of advertising in major shopping centers, and opening doors of exploration for this kind of operation.

Keywords: Totem interactive. PHP. Touchscreen.

Sumário

Lista de Figuras

Glossário

AJAX – *Asyncronous Javascript and XML* – JavaScript Assíncrono e XML

CSS – *Cascanding Style Sheets* – Folha de Estilos em Cascata

DAEMON - *Disk And Execution Monitor* - Monitor de Execução e de Disco

DNS – *Domain Name System* – Sistema de Nomes de Domínio

GNU - *General Public License* - Licença Pública Geral para software livre

HTML – *Hypertext Markup Language* – Linguagem de Marcação de Hipertexto

HTTP – *HyperText Transfer Protocol* – Protocolo de Transferência de Hipertexto

HTTPS – *Hyper Text Transfer Protocol Secure* – Protocolo de Transferência Seguro

IP – *Internet Protocol* – Protocolo de Internet

MVC – *Model, View, Controller* - Padrão de Projeto Modelo, Visão e Controle

PC – *Personal Computer* – Computador Pessoal

PHP – *PHP Hypertext Preprocessor* – Personal Home page

POO – *Programação Orientada a Objetos*

SERVER-SIDE – Sistemas com código executado no servidor

SGBD – Sistema Gerenciador de Banco de Dados

SQL – *Structure Query Language* – Linguagem Estrutura de Consulta

USB – *Universal Serial Bus* – Barramento Serial Universal

UML – Unified Modeling Language-Linguagem para Especificação e Documentação

W3C – *World Wide Web Consortium* – Consórcio da Rede de Alcance Mundial

WWW – *World Wide Web* – Rede de Alcance Mundial

XHTML – *eXtensible Hypertext Markup Language* – Linguagem Extensível para Marcação de Hypertexto

XML – *eXtensible Markup Language* – Linguagem Extensível de Marcação

Introdução

Atualmente a tecnologia é aplicada das mais diversas formas e nos mais variados setores da sociedade. De acordo com a evolução da tecnologia, nos próximos anos, uma onda de conectividade mudará radicalmente a forma como as pessoas trabalham, vivem e se divertem. Essas mudanças terão importantes impactos na qualidade de vida das pessoas.

Utilização de tecnologias no interior de lojas, para oferecer comodidade e facilidade – este é o intuito do presente trabalho, por meio do desenvolvimento de um programa que acoplado a um totem tenha a função de orientar os usuários e divulgar propagandas e informações às pessoas que freqüentam determinados locais.

No decorrer da graduação em engenharia de Computação, varia disciplinas foram aplicadas no contexto do projeto, entre elas pode-se citar:

- Arquiteturas de Computadores.
- Lógica digital.
- Linguagem técnica de programação.
- Instalações Elétricas, dentre outras.

1.1 Motivação

Inicialmente, a idéia para desenvolver um totem surgiu de forma espontânea durante o primeiro contato com um desses equipamentos. Os totens podem ser desenvolvidos para os mais diversos tipos de funções: informativos, promocionais, de propaganda e outros. Daí surgiu o projeto Totem Interativo. O primeiro contato que o autor deste projeto teve com um totem foi ao fazer um *check-in* no quiosque de uma companhia aérea. O equipamento era interativo, e os próprios passageiros faziam seu *check-in*, agilizando assim a fila.

Assim, pesquisando e observando, levantou-se a necessidade de localização em *shopping centers*, e procurou-se idealizar uma interface que de forma fácil e interativa pudesse colaborar com essa localização. Hoje em dia podemos verificar que várias empresas utilizam em seu ambiente algum tipo de totem, a ser utilizado conforme a necessidade dos clientes, tudo dependendo do software desenvolvido.

Também as formas de visualização podem ser adaptadas às necessidades de quem utilizará os totens. Existem os que fazem uso de teclados e os que utilizam o recurso de *touchscreen* para interação com seus usuários – neste caso, a tela é inicialmente mais cara, o que torna o equipamento com teclado mais viável; porém, totens com *touchscreen* levam vantagem quando o assunto é interatividade e durabilidade.

Em Março de 2008, época da concepção da idéia não existia tantos totens como podemos ver hoje nas ruas com grande movimento, onde a intenção é fazer pesquisas de opinião e até simulados de votação. Nesses casos, faz-se uso de telas de *touchscreen* para que os usuários tenham maior facilidade na hora de escolher seus candidatos ou a opção que quiserem.

Grandes lojas utilizam certo tipo de totem para ler o código de barras e informar o preço para o usuário, mostrar quais produtos há na loja e outras funções correlatas. Existem totens nos campus de faculdades que disponibilizam de forma fácil e rápida a impressão de documentos. Tudo só depende de criatividade para fundir a tecnologia ao cotidiano das pessoas, e essa é a proposta do presente projeto.

1.2 Metodologia

Este projeto segue uma linha de pesquisa bibliográfica baseada em consultas a materiais publicados em livros, periódicos e sites da Internet. Com base nestas pesquisas o projeto tem finalidade aplicada, ou seja, prevê aplicação prática e experimental como objeto de estudo.

1.3 Objetivo

O objetivo deste projeto é a inclusão de um meio eletrônico em locais públicos com grande fluxo de pessoas, onde seja necessário ou recomendável oferecer auxilio na localização, verificação de conteúdos relacionados e interação dos usuários com os horários dos cinemas e a visualização dos trailers dos filmes em cartaz.

Este trabalho está dividido em cinco capítulos:

- Capítulo 1: é apresentada a introdução, motivação, metodologia e objetivo.
- Capítulo 2: descreve os componentes de hardware utilizados.
- Capítulo 3: detalha as tecnologias utilizadas para desenvolver o software.
- Capítulo 4: trata sobre as metodologias empregadas.
- Capítulo 5: descreve a implementação do projeto.

2 Hardware do totem

O Totem Interativo é o acoplamento do *hardware* necessário para a perfeita execução do projeto, com um *software* desenvolvido exclusivamente para esse fim. O *hardware* utilizado foi determinado com a seguinte premissa: o projeto segue o melhor custo/benefício possível (Apêndice B), pois assim pode-se viabilizá-lo para uso comercial.

Partindo deste ponto foi determinado que a placa mãe, processador, memória e disco rígido são apontados com vista à maximização do custo/beneficio e que não serão determinantes para a seqüência do trabalho. Apenas deverão seguir uma pré-configuração estabelecida, preservando a integridade do projeto e considerando o desempenho esperado.

Configuração necessária:

- Processador
  - Intel Pentium IV 631, Barramento de 800Mhz.
- Placa Mãe
  - Asus P5GC-MX Barramento de 800Mhz, Socket 775, DDR2.
- Disco rígido
  - Capacidade 80 GB, Velocidade de rotação 7200 rpm. Interface Serial ATA II (3,0Gbps).
- Memória
  - Kingston DIMM (SDRAM-DDR2, 1.8V) 1GB 667MHz – 240 pinos.

Determinou-se que três partes do *hardware* do projeto são de fundamental importância, pois tratam de comunicação e interatividade. Assim será dada ênfase ao adaptador de rede, a impressora e à tela *touchscreen*.

2.1 Adaptadores de rede

Adaptadores de rede, ou comumente chamadas placas de rede, são dispositivos necessários para que o computador possa se comunicar com uma rede. Através dessas placas o computador pode trocar informações com outros computadores conectados. (TORRES, 1998)

A função do adaptador de rede é basicamente controlar o fluxo de dados, seja transmissão, seja recepção entre o computador e a rede. (TORRES, 1998)

A palavra *Wireless* provém do inglês: *wire* (fio, cabo) e *less* (sem), ou seja, sem fios. Uma conexão *Wireless* é qualquer forma de conexão entre dois sistemas transmissores e receptores de dados que não requeira o uso de fios. Para tanto são utilizadas freqüências de rádio ou sinais luminosos, geralmente na faixa de infravermelho. Utiliza como meio de transmissão o ar ou o vácuo. Sistemas de comunicação *Wireless* podem permitir o tráfego de voz, dados ou ambos. (SOUZA, 2002)

2.2 Impressora

O dispositivo de impressão é um periférico que, quando conectado a um computador ou a uma rede de computadores, tem a função de dispositivo de saída, imprimindo textos, gráficos ou qualquer outro resultado de uma aplicação.

As impressoras térmicas são mais rápidas, mais econômicas e mais silenciosas do que outros modelos de impressoras. São muito comuns em aparelhos de fax e máquinas que imprimem cupons fiscais e extratos bancários. (WIKIPEDIA.ORG, 2008)

2.3 Tela Touchscreen

O *touchscreen* é um painel transparente que acompanha interna ou externamente o monitor e é ligado a uma porta serial ou USB.

Um computador *touchscreen* é um equipamento em que o visor é sensível ao toque humano, permitindo que o usuário possa interagir com a máquina tocando em imagens ou palavras na tela. São dispositivos e sistemas concebidos para ajudar as pessoas que têm dificuldade em manipular o mouse ou teclado. Essa tecnologia pode ser utilizada como uma alternativa de interface do usuário com os aplicativos que normalmente exigem um mouse, como um *Web browser*. Algumas aplicações são concebidas especificamente para tecnologia *touchscreen*, tendo ícones maiores que os típicos de aplicação em PC. (JUNJA, 2001)

Existem três tipos de tecnologia *touchscreen*:

2.3.1 Sistema resistivo

O sistema resistivo consiste de um painel de vidro normal, recoberto por uma camada metálica condutora e outra resistiva. Estas duas camadas são mantidas afastadas por espaçadores e uma camada resistente a riscos é colocada por cima de todo o conjunto. Uma corrente elétrica passa através das duas camadas enquanto o monitor está operacional. Quando um usuário toca a tela, as duas camadas fazem contato exatamente naquele ponto. A mudança no campo elétrico é percebida, e as coordenadas do ponto de contato são calculadas pelo computador. Logo que as coordenadas são conhecidas, um *driver* especial traduz o toque em algo que o sistema operacional possa entender, parecido com o que faz o *driver* do mouse do computador ao traduzir os movimentos em uma operação de clicar ou arrastar.

2.3.2 Sistema capacitivo

No sistema capacitivo, uma camada que armazena carga elétrica é colocada no painel de vidro do monitor. Quando um usuário toca o monitor com seu dedo, parte da carga é transferida para o usuário, de modo que a carga na camada capacitiva diminui. Esta diminuição é medida nos circuitos localizados em cada canto do monitor. Considerando as diferenças relativas de carga em cada canto, o computador calcula exatamente onde ocorreu o toque e então envia esta informação para o *software* do *driver* da tela. Uma vantagem que o sistema capacitivo apresenta sobre o resistivo é que ele transmite quase 90% da luz do monitor, enquanto o sistema resistivo transmite apenas 75%. Isso dá ao sistema capacitivo uma imagem muito mais clara do que o sistema resistivo.

2.3.3 Sistema de onda acústica superficial

No monitor de um sistema de onda acústica superficial, dois transdutores (um receptor e um emissor) são posicionados ao longo dos eixos x e y da placa de vidro do monitor. Também instalados sobre o vidro, encontram-se refletores que enviam de volta um sinal elétrico proveniente de um transdutor para o outro. O transdutor receptor é capaz de informar se a onda foi perturbada por um evento de toque em qualquer instante e localizá-lo. A configuração por onda acústica não possui camadas metálicas sobre a tela, permitindo a passagem de 100% da luz e uma claridade perfeita da imagem. Isso torna o sistema de onda acústica ideal para exibição de gráficos detalhados. Já os dois outros sistemas apresentam uma degradação significativa da claridade.

Outra área na qual os sistemas diferem é quanto aos estímulos que serão registrados como um evento de toque. Um sistema resistivo registra um toque

enquanto as duas camadas estiverem em contato, o que significa que não haverá diferença se você tocar com seu dedo ou com uma bola de borracha. Por outro lado, um sistema capacitivo precisa de uma entrada condutora, geralmente o dedo, para registrar um toque. O sistema de onda acústica superficial funciona parecido com o sistema resistivo, permitindo o toque com quase qualquer objeto, exceto objetos duros e pequenos, como a ponta de uma caneta. (JUNJA, 2001)

Neste projeto faz-se uso de placa Ethernet PCI de 10 Mbps / 100 Mbps com encaixes RJ 45 para cabos de par trançado. E a conexão *wireless* é feita através de placa USB 54Mbs, essa forma de comunicação é essencial para o desenvolvimento do projeto, pois ele funciona exclusivamente na *Web*. A Impressora utilizada é de pequeno porte da marca Sipix, que faz uso da tecnologia térmica para impressão de documentos e que faz comunicação por interface infre-vermelho e porta serial COM1. A tela de touchscreen utilizada é do tipo capacitiva, é acoplada por meio da porta USB, tem boa durabilidade e um bom preço.

3 Tecnologias aplicadas no software

Neste capítulo serão tratadas as tecnologias utilizadas para o desenvolvimento do sistema T I 1.0. Ele faz uso de tecnologias *open-source* em todo o seu contexto, sendo várias as opções de linguagens de programação que se pode usar no desenvolvimento do sistema.

Muito se discute a escolha da linguagem a ser utilizada. Tal escolha dependerá de vários fatores: Onde o sistema vai “rodar” (infra-estrutura, arquitetura, sistema operacional, entre outras variáveis), quais as necessidades “vitais”, e em determinada tarefa qual oferece a relação desempenho/segurança mais competitiva.

Para um sistema funcionar na *web* é necessário um servidor – neste caso faz-se uso do servidor *web* Apache. O servidor de banco de dados utilizado é o MySQL, as linguagens de programação são PHP e JavaScript e é implementado com o auxilio da Zend Framework. Todas estas tecnologias estão de acordo com os padrões da web 2.0, baseadas nos métodos Tableless e formatações com CSS.

3.1 Servidor Apache

O servidor *web* é a ferramenta responsável pela distribuição de documentos HTML através do protocolo HTTP. Um Servidor *Web* é um *software*, geralmente implementado na forma de *daemon*, que controla a recuperação e disponibilização de arquivos no formato de hypertexto em um sistema de arquivos. (DIETZ, 1998 *apud* SOUZA S, 2003)

O *Apache Web Server* é um Servidor *Web, open-source*, multi-plataforma (BSD, GNU/Linux, UNIX systems, Microsoft Windows e outras), e hoje é o mais utilizado em todo mundo. As principais características do Apache são flexibilidade, altamente configurável, robustez, escalabilidade, pode ser configurado para

diferentes funções e é composto de módulos separados. Com o crescimento da disponibilização de documentos na *Web*, o aumento da complexidade destes servidores e da grande demanda por parte dos usuários, os Servidores *Web* necessitam atender em tempo hábil a todas as solicitações de recuperação, decorrendo o aumento da necessidade de poder computacional nos Servidores *Web*. (DIETZ, 1998 *apud* SOUZA S, 2003)

3.2 MySQL

MySQL é um sistema de gerenciamento de banco de dados relacionais que utiliza a linguagem padrão SQL e é largamente utilizado em aplicações para a internet. É o mais popular entre os bancos de dados com código-fonte aberto. Há mais de cinco milhões de instalações do MySQL no mundo todo, inclusive em sites com alto volume de dados e de tráfego, como *Google, Nasa* e *Suzuki.* (PRATES, *et al.* 2006)

O MySQL está disponível através de um esquema de licenciamento duplo, tanto como um *software* livre quanto como um *software* comercial. Todo o *software* é disponibilizado sob o licenciamento GNU *General Public License (GPL).* No entanto, também se poderá adquiri-lo em situações que prevejam licença comercial. É uma alternativa atrativa porque mesmo possuindo uma tecnologia complexa de banco de dados, seu custo é bastante baixo.

Trata-se de um *software* robusto e ao mesmo tempo flexível, com destaque para suas características de velocidade, escalabilidade e confiabilidade, o que vem fazendo com que ele seja adotado por departamentos de TI (Tecnologia da Informação), desenvolvedores *Web* e vendedores de pacotes de *softwares*. (PRATES, *et al.* 2006)

A velocidade sempre foi um aspecto importante no projeto do MySQL. Novos recursos são adicionados apenas quando isso pode ser feito sem prejudicar o desempenho da aplicação. Esse procedimento tem assegurado que o MySQL mantenha suas características de rapidez e eficiência.

Para o desenvolvimento de sites dinâmicos, o MySQL forma uma excelente dupla com a linguagem PHP, tanto para *web* sites pequenos como para grandes portais. Ele suporta a grande maioria dos recursos considerados importantes pelos usuários e administradores de banco de dados, como por exemplo, transações, *lock* de registros, chaves estrangeiras, subconsultas, e pesquisa *full-text*.

Devido à facilidade de instalação e uso, o MySQL é uma ótima ferramenta de aprendizado sobre bancos de dados. Somam-se a isso os baixos requisitos de recursos de *hardware*, como disco rígido e memória.

3.3 PHP

PHP, que significa "*Hypertext Preprocessor*", é uma linguagem de programação (de *script Open Source* de uso geral), interpretada, do lado servidor ('*server-side*'), e incorporada dentro do código HTML, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links. É a linguagem mais utilizada para desenvolvimento *Web* com a criação de páginas dinâmicas.

Como muitos projetos de Software Livre, o PHP nasceu despretensiosamente como uma ferramenta pessoal e cresceu rapidamente para o uso corporativo.

O PHP é multiplataforma, aceita vários sistemas operacionais, como o Windows, Unix e Linux. Também, possibilita a conexão direta com uma grande quantidade de Banco de Dados relacionais, como Oracle, Sybase, Informix, MySQL.

É suportado pela maioria dos servidores *WEB* que existem no mercado, como o APACHE, IIS e PWS.

A quantidade de aplicações já existentes em PHP, a sua robustez, a rapidez na codificação e a facilidade de aprendizagem tornam o PHP uma escolha excelente para o desenvolvimento deste projeto e para demais aplicações de pequeno e médio porte para a *web*.

Além disso, PHP é melhor por seguir o padrão da linguagem C e ter mais material disponível na literatura técnica. A diferença de PHP com relação a linguagens semelhantes, tais como JavaScript, é que o código PHP é executado no servidor, sendo enviado para o cliente apenas HTML puro. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente

A utilização do PHP no desenvolvimento de sistemas para Internet utilizando o paradigma de Orientação a Objeto tem crescido muito. Juntamente com isso, vários *frameworks* (conjunto de classes com objetivo de reutilização de um projeto) que implementam o padrão MVC em PHP.

3.4 JavaScript

Javascript é uma linguagem de *script* orientada a objetos com uma sintaxe bastante similar a C, C++, Pascal e Delphi. É uma linguagem dirigida por eventos, no sentido de que é projetada para reagir quando um evento ocorre. É utilizada para desenvolver aplicações cliente para Internet.

A linguagem Javascript foi projetada para manipular e apresentar informação através de um navegador. Ela não é capaz de recuperar informações de outro arquivo ou salvar dados em um servidor da *Web*, ou no computador do usuário. Isto significa que não é possível escrever um programa Javascript que, por

exemplo, varra os diretórios de um computador, lendo ou apagando arquivos do usuário.

Javascript é uma linguagem independente de plataforma, ou seja, o código escrito nesta linguagem não depende de uma plataforma específica (Windows, Macintosh, UNIX) depende apenas do navegador que a interpreta. Dessa forma, quer o usuário tenha um navegador para Windows, Macintosh ou UNIX, o código Javascript será executado sem que nenhuma adaptação seja necessária.

3.5 CSS

Cascading Style Sheets (CSS) são descrições de como os documentos devem ser apresentados e determinam a forma do HTML.

Uma folha de estilos consiste de uma ou mais definições de estilo (tamanho de fonte, estilo da fonte, alinhamento de texto, cor de texto e do fundo, margens, altura da linha) para elementos HTML que podem ser *linkados* ou embutidos em documentos HTML. Esta funcionalidade foi criada para propiciar aos projetistas e desenvolvedores *Web* a possibilidade de contar com estilos e posicionamentos consistentes no documento.

3.6 Tableless

Table (Tabela) *less* (sem) ou seja sem tabelas. É um método de construir sites usando os Padrões *Web* (Web Standards) como guia. É utilizado CSS para a formatação das informações apresentadas nos arquivos de marcação de texto, ou arquivos HTML.

Não são utilizadas as famosas Tabelas para fazer a estruturação do site para estruturação, é utilizado exclusivamente o CSS. Porém as tabelas não foram

abolidas, elas são somente indicadas para a apresentação de dados e não são utilizadas para a estruturação de *layout*. (TABLELESS.COM, 2008)

3.7 Framework

Framework é uma estrutura de suporte definida em que outro projeto do *software* pode ser organizado e desenvolvido. Tipicamente, um *framework* pode incluir programas de apoio, bibliotecas de código, linguagens de *script* e outros *softwares* para ajudar a desenvolver e juntar diferentes componentes do seu projeto.

Especificamente em orientação a objetos, *framework* é um conjunto de classes com objetivo de reutilização de um *design*, provendo um guia para uma solução de arquitetura em um domínio específico de *software*.

Framework se diferencia de uma simples biblioteca, pois esta se concentra apenas em oferecer implementação de funcionalidades, sem definir a reutilização de uma solução de arquitetura. (WIKIPEDIA.ORG, 2008)

Framework é uma técnica de reuso orientado a objeto que compreende tanto *design* quanto código. Eles compartilham muitas características com técnicas de reuso de maneira geral e com técnicas de reuso orientado a objetos em particular. (RUBIN, 2003)

O *framework* tenta capturar o fluxo de controle de aplicação dentro de um domínio. Este fluxo pode ser especializado mais tarde por uma aplicação específica. Os *frameworks* são adaptáveis por customização e extensão da estrutura que eles provêem. (RUBIN, 2003)

3.8 Zend Framework

A Zend Framework fornece os alicerces para construção de *websites* baseados no padrão de projeto Modelo-Visão-Controlador (MVC).

O sistema foi projetado para ser leve, modular e extensível. É um design minimalista cujo objetivo é oferecer flexibilidade e alguma liberdade aos usuários, ao mesmo tempo em que fornece uma estrutura que permite aos sistemas construídos com base na Zend Framework compartilharem algumas convenções comuns e padrões similares de código.

O fluxo de trabalho da Zend Framework é implementado por alguns componentes. Não é necessário entender completamente a orquestração destes componentes para usar o sistema, sendo mais útil compreender a maneira como os processos trabalham. (ZEND.COM, 2008)

As principais tecnologias utilizadas neste projeto foram citadas no capítulo 3. O *software* foi desenvolvido para funcionar na *web*, portanto o servidor é importante para controlar a recuperação e disponibilização dos arquivos.

O Servidor de Banco de dados utilizado é o MySQL, um servidor *open-source* e que tem uma capacidade de interagir com a linguagem PHP, formando uma dupla perfeita para desenvolvimento *web*. Tudo isso foi feito com base na Zend Framework, que fornece o suporte para implementação mais rápida e adequada.

4 Metodologias aplicadas ao desenvolvimento de software

Neste capítulo são comentadas as metodologias que foram utilizadas na elaboração e especificação do sistema. A UML propõe uma especificação e documentação de sistemas orientados a objetos, e conforme esses padrões documentou-se o sistema T I 1.0. A arquitetura foi elaborada conforme o padrão MVC, o qual propõe uma divisão do sistema em três camadas com o intuito de reutilização de códigos, segurança e fácil manutenção de suporte.

4.1 UML

A UML (*Unified Modeling Language*) é uma linguagem para especificação, documentação, visualização e desenvolvimento de sistemas orientados a objetos. Sintetiza os principais métodos existentes, sendo considerada uma das linguagens mais expressivas para modelagem de sistemas orientados a objetos.

Por meio de seus diagramas é possível representar sistemas de *softwares* sob diversas perspectivas de visualização. Facilita a comunicação de todas as pessoas envolvidas no processo de desenvolvimento de um sistema gerentes, coordenadores, analistas, desenvolvedores por apresentar um vocabulário de fácil entendimento. (Marcos, 2001)

Na figura 4.1 é exemplificada a estrutura proposta pela UML. É basicamente identificado como são separados os diagramas de acordo com sua proposta, em um grupo são colocados os diagramas de estrutura e no outro grupo os diagramas de comportamento.

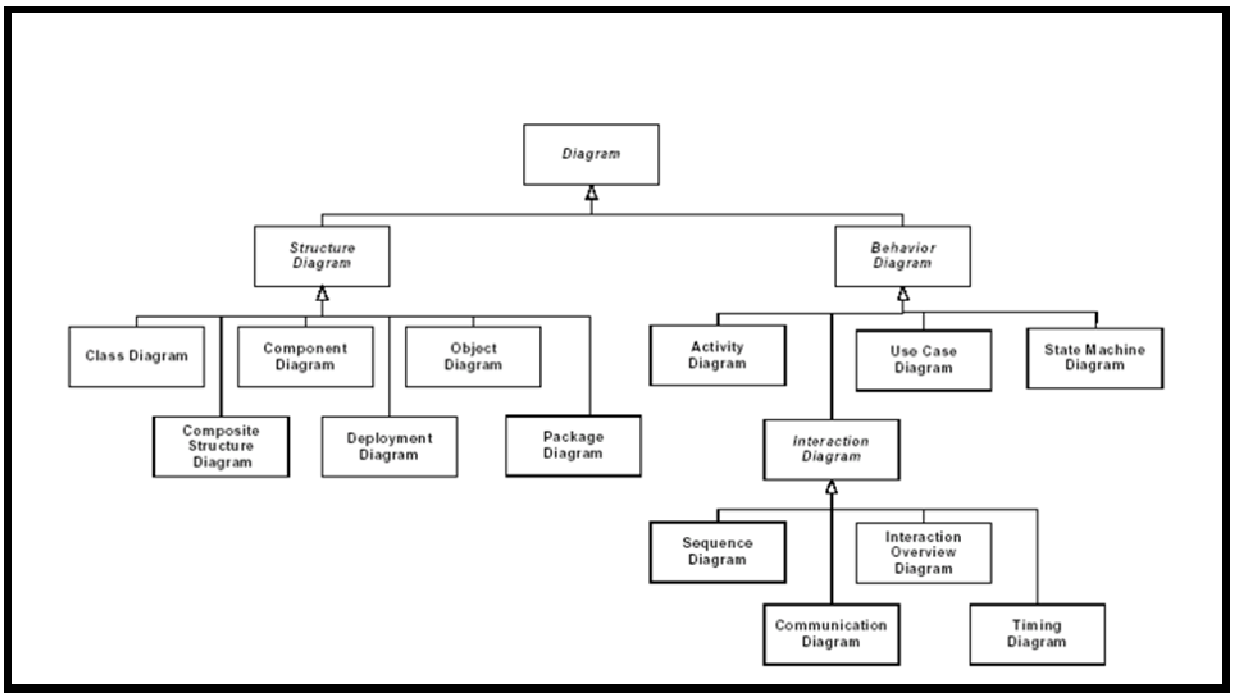

Figura 4.1 Exemplo de Diagramas da UML.

4.1.1 Diagramas de caso de uso.

Pode-se dizer que um caso de uso é um *"documento narrativo que descreve a seqüência de eventos de um ator que usa um sistema para completar um processo".* (JACOBSON, 2000)

Um caso de uso é uma técnica de modelagem usada para descrever o que um novo sistema deve fazer. Ele é construído através de um processo interativo no qual as discussões entre o cliente e os desenvolvedores do sistema conduzem a uma especificação do sistema onde todos estão de acordo. (JACOBSON, 2000)

Os casos de uso têm por objetivo:

- decidir e descrever os requisitos funcionais do sistema,
- fornecer uma descrição clara e consistente do que o sistema deve fazer,

- permitir descobrir os requisitos funcionais das classes e operações do sistema. (*Casos de uso NÃO são requisitos.*)

Os componentes de um modelo de casos de uso são:

- *Ator* - é um papel que tipicamente estimula/solicita ações/eventos do sistema e recebe reações. Cada ator pode participar de vários casos de uso.
- *Casos de uso* - documento narrativo que descreve a seqüência de eventos feitos por um ator no uso do sistema.
- *Sistema* - O sistema a ser modelado.

Na figura 4.2 é exemplificado um modelo de diagrama de casos de uso, que segundo os modelos da UML consistem de casos de uso que mostram uma versão macro das funcionalidades do sistema através dos atores e seus relacionamentos.

O usuário se relaciona com as funcionalidades “Emprestando Material” e “Pesquisando Assunto” e o gerente se relaciona com as funcionalidades “Castrando Usuário”, “Cadastrando Material” e “Analisando Empréstimos”.

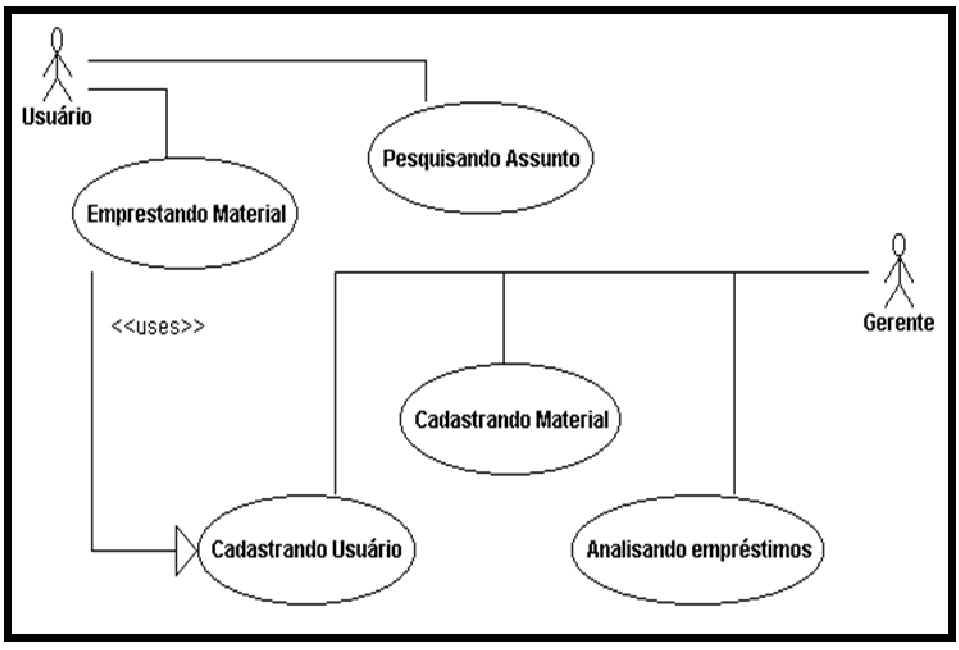

Figura 4.2 – Exemplo de Diagramas da UML (Casos de Uso)

4.1.2 Diagramas de classes

Diagramas de classe descrevem as classes que formam a estrutura do sistema e suas relações. As relações entre as classes podem ser associações, agregações ou heranças. As classes possuem, além de um nome, os atributos e as operações que desempenham para o sistema. Uma relação indica um tipo de dependência entre as classes, e essa dependência pode ser forte (como no caso da herança ou da agregação) ou mais fraca (como no caso da associação), mas indicam que as classes relacionadas cooperam de alguma forma para cumprir um objetivo para o sistema. (FURLAN, 2003)

Sendo uma linguagem de descrição, a UML permite diferentes níveis de abstração aos diagramas, dependendo da etapa do desenvolvimento do sistema em que se encontram. Assim, os diagramas de classe podem exibir nas fases iniciais da análise apenas o nome das classes, e em uma fase seguinte os atributos e operações (como é exemplificado na figura 4.3). Finalmente, em uma fase avançada do projeto pode exibir os tipos dos atributos, a visibilidade, a multiplicidade das relações e diversas restrições. Existem elementos na UML para todas estas representações.

O diagrama de classes, ao final do processo de modelagem, pode ser traduzido em uma estrutura de código que servirá de base para a implementação dos sistemas. Observa-se, no entanto, que não existe no diagrama de classes uma informação sobre os algoritmos que serão utilizados nas operações, e também não se pode precisar a dinâmica do sistema porque não há elementos sobre o processo ou a seqüência de processamento neste modelo. (FURLAN, 2003)

Na figura 4.3 é exemplificado um modelo de diagrama de classes, que detalha como deverá ser a comunicação interna dos objetos do sistema e seus métodos.

A tabela usuários mantém um relacionamento de 1 para *, ou seja, um usuário para * materiais, e a tabela materiais mantém um relacionamento de 1 material para * reservas.

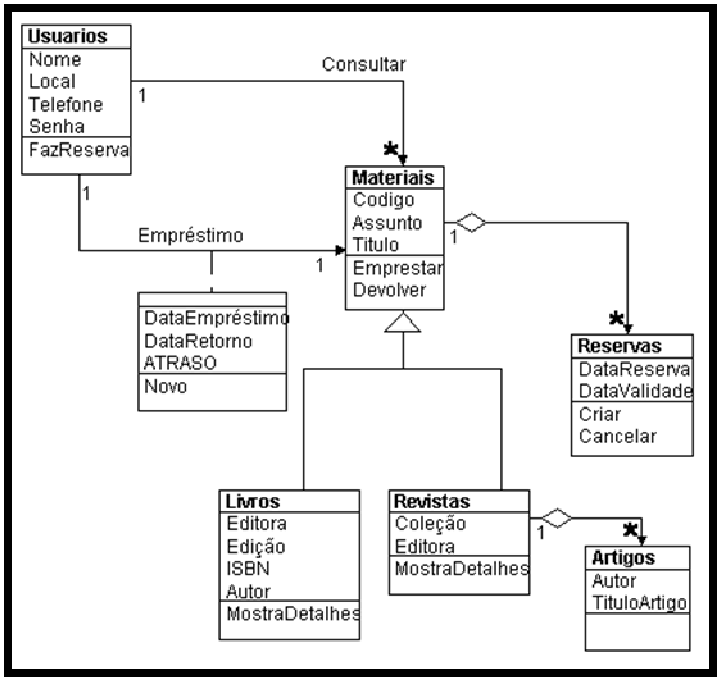

Figura 4.3 – Exemplo de Diagramas da UML (Diagrama de Classes)

4.2 Orientação a Objetos

A orientação a objetos, também conhecida como Programação Orientada a Objetos (POO), é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de *software* chamadas de objetos.

A análise e projeto orientados a objetos têm como meta identificar o melhor conjunto de objetos para descrever um sistema de *software*. O funcionamento deste sistema se dá através do relacionamento e troca de mensagens entre estes objetos. (MARCOS, 2001)

Hoje existem duas vertentes no projeto de sistemas orientados a objetos. O projeto formal, normalmente utilizando técnicas como a notação UML, processos

de desenvolvimento como o RUP e a programação extrema, que utiliza pouca documentação, programação em pares e testes unitários.

Na programação orientada a objetos, implementa-se um conjunto de classes que definem os objetos presentes no sistema de *software*. Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento com outros objetos.

Serão listados alguns exemplos do que compões a orientação a objetos;

- **Classe** representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos, através de métodos, e quais estados ele é capaz de manter, através de atributos. Exemplo de classe: os seres humanos.
- **Objeto** é uma instância de uma classe. Um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos. Exemplo de objetos da classe humanos: João, José, Maria.
- **Mensagem** é uma chamada a um objeto para invocar um de seus métodos, ativando um comportamento descrito por sua classe. Também pode ser direcionada diretamente a uma classe (através de invocação a um método dinâmico).
- **Herança** (ou generalização) é o mecanismo pelo qual uma classe (sub-classe) pode estender outra classe (super-classe), aproveitando seus comportamentos (métodos) e estados possíveis (atributos). Há herança múltipla quando uma sub classe possui mais de uma super classe. Essa relação é normalmente chamada

de relação "é um". Um exemplo de herança: mamífero é super classe de humano. Ou seja, um humano é um mamífero.

- **Associação** é o mecanismo pelo qual um objeto utiliza os recursos de outro. Pode tratar-se de uma associação simples "usa um" ou de um acoplamento "parte de". Por exemplo: um humano usa um telefone. A tecla "1" é parte de um telefone.
- **Encapsulamento** consiste na separação de aspectos internos e externos de um objeto. Este mecanismo é utilizado amplamente para impedir o acesso direto ao estado de um objeto (seus atributos), disponibilizando externamente apenas os métodos que alteram estes estados. Exemplo: você não precisa conhecer os detalhes dos circuitos de um telefone para utilizá-lo. A carcaça do telefone encapsula esses detalhes, provendo a você uma interface mais amigável (os botões, o monofone e os sinais de tom).
- **Abstração** é a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de *software*.
- **Polimorfismo** é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura (lista de parâmetros) mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse. A decisão sobre qual o método que deve ser selecionado, de acordo com o tipo da classe derivada, é tomada

em tempo de execução, através do mecanismo de ligação tardia. No caso de polimorfismo, é necessário que os métodos tenham exatamente a mesma identificação, sendo utilizado o mecanismo de redefinição de métodos. Esse mecanismo de redefinição não deve ser confundido com o mecanismo de sobrecarga de métodos.

4.3 MVC

MVC (*Model View Controller*) é um Padrão de Projeto que cria uma separação do código em três camadas, cuja finalidade é basicamente separar como as coisas são feitas, de como elas são apresentadas. (PESSOA, 2007)

Com o aumento da complexidade das aplicações desenvolvidas torna-se fundamental a separação entre os dados (***Model***) e o layout (***View***). O MVC resolve este problema através da separação das tarefas de acesso aos dados e a lógica de negócio, lógica de apresentação e de interação com o usuário, introduzindo um componente entre os dois (***Controller*)**. (WIKIPEDIA.ORG, 2008)

- **Model** - Responsável pelas regras de negócio, persistência de dados. Não deve ter nenhum tipo de formatação nesta camada, isto será feito na camada de visão (*view*).
- **View** - Responsável pela apresentação, pela formatação da saída. A *view* utiliza os recursos da *Model* e trata sua saída.
- **Controller** - Recebe as requisições vindas da camada de visão (*View*), manipula os objetos da *Model*, e redireciona para outros *Controllers*.

Desta forma, alterações feitas no *layout* não afetam a manipulação de dados, e estes poderão ser reorganizados sem alterar o *layout*.

5 Implementação do Totem Interativo

A implementação do Totem Interativo foi dividida em duas grandes fases, a parte dos componentes físicos “*hardware*” e o sistema “*software*” que será denominado T I 1.0, que serão detalhadas a seguir.

O escopo definido do projeto prevê um sistema que possibilite ao usuário localização dentro do shopping Center, ofereça opções entre as lojas cadastradas e vizualise mapas com a localização destas lojas, outro módulo do sistema deverá oferecer informações sobre os filmes em cartaz no cinema do shopping Center. Estas funcionalidades serão proporcionadas por meio de um totem implementado com tela *touchscreen* e será de uso público.

5.1 Componentes Físicos

Na fase de elaboração e viabilidade do projeto, levantou-se métodos, custos, componentes e identificou-se requisitos e necessidades. As grandes barreiras encontradas diziam respeito ao pequeno espaço do gabinete disponível para o acoplamento dos dispositivos de hardware, a impressora principalmente, pois no mercado é difícil encontrar uma impressora muito pequena por um custo razoável. Após pesquisa, encontrou-se impressora portátil, geralmente usada em palms, cuja principal função é imprimir recibos e relatórios simples, e que possui um custo acessível e características que se adequam ao projeto.

A tela com recurso de *touchscreen* é a parte principal para elaboração deste projeto. Não é fácil consegui-la no mercado brasileiro com as características e funcionalidades necessárias para a construção do totem; portanto, a tela utilizada foi obtida no mercado externo. É um modelo de fácil instalação e de boa qualidade, que usa a tecnologia capacitiva. Os demais itens que compõem a montagem do protótipo

já foram citados antes, o que é suficiente para o entendimento e elaboração do totem.

Conforme as figuras A1, A2, e A3, pode-se verificar o projeto implementado. Somente após a conclusão desta fase foi dada continuidade ao desenvolvimento do software, devido à necessidade de haver uma perfeita harmonia entre funcionamento do sistema e reação do *hardware*. Essa harmonia só seria possível após a verificação com testes de resposta do protótipo já em funcionamento.

5.2 Implementação e análise do software do sistema T I 1.0

A análise e a documentação do sistema foram desenvolvidas com base nos diagramas da UML. Apresenta-se na figura 5.2 o diagrama de classes, que irá representar a interação entre as entidades do sistema, e na figura 5.3 os diagramas de caso de uso, os quais relatam as funcionalidades e interações que o sistema possui com os seus atores em cada cenário. Primeiramente, nessa fase, foi definido o modelo do banco de dados.

5.2.1 Modelagem de Banco de Dados

Após fechado o escopo do sistema T I 1.0 e após a análise de todas possíveis ações do sistema, foi definido o modelo de banco de dados.

Através dos relacionamentos e das tabelas do banco de dados relacionadas na figura 5.1, todos os tipos e situações de questionamentos são atendidos de forma otimizada e rápida, pois o acesso às informações é de fácil

entendimento e através das *constraints* criadas (chaves primárias e chaves estrangeiras) o tempo de retorno dos dados é mínimo.

Na figura 5.1 visualiza-se a modelagem do sistema T I 1.0. Esta modelagem é separada em dois grupos, um de localização que determina as classes de clientes, a relação com os clientes e os dados cadastrais. No outro grupo é visualizado o grupo de cinemas que relaciona o cinema com as salas, filmes e horários.

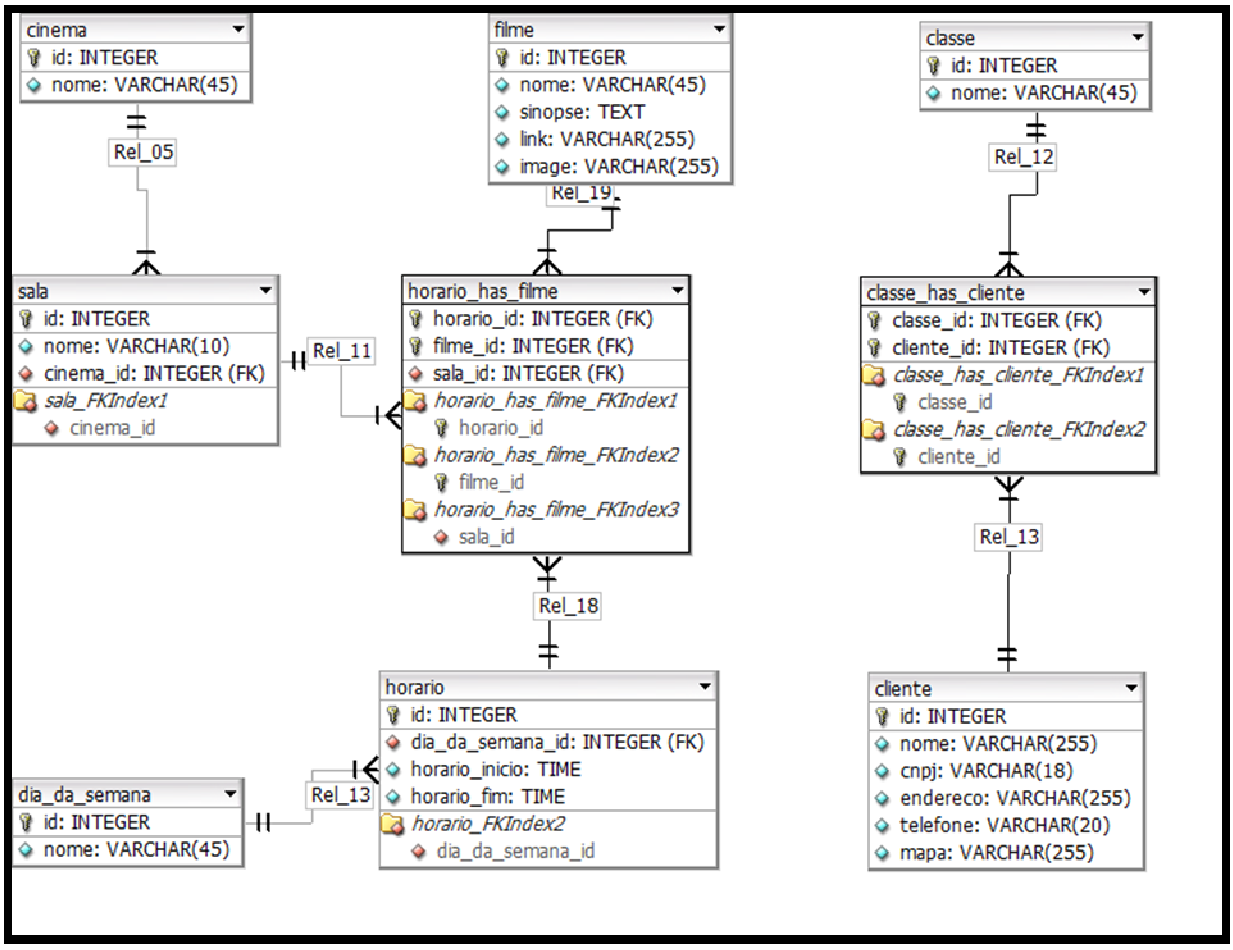

Figura 5.1 Modelagem do Banco de Dados Sistema T I 1.0

5.2.2 Diagrama de Classes

O diagrama de classes representa as interações entre as entidades e também os relacionamentos em que cada entidade exerce sobre a outra entidade. Através desse diagrama é possível compreender com facilidade as dependências entre as entidades do sistema.

Na figura 5.2 é visualizado o Diagrama de Classes do Sistema T I 1.0. De acordo com a modelagem do banco de dados, determinam-se quais classes serão contempladas, seus atributos e métodos.

A classe Cliente tem como atributos nome, CNPJ, endereço e telefone; como métodos localizar clientes e localizar mapas. A classe Filme tem como atributos nome, sinopse, *link* e trailer; como métodos ver horário, ver trailer e ver sinopse. As demais classes têm atributos e métodos conforme a figura.

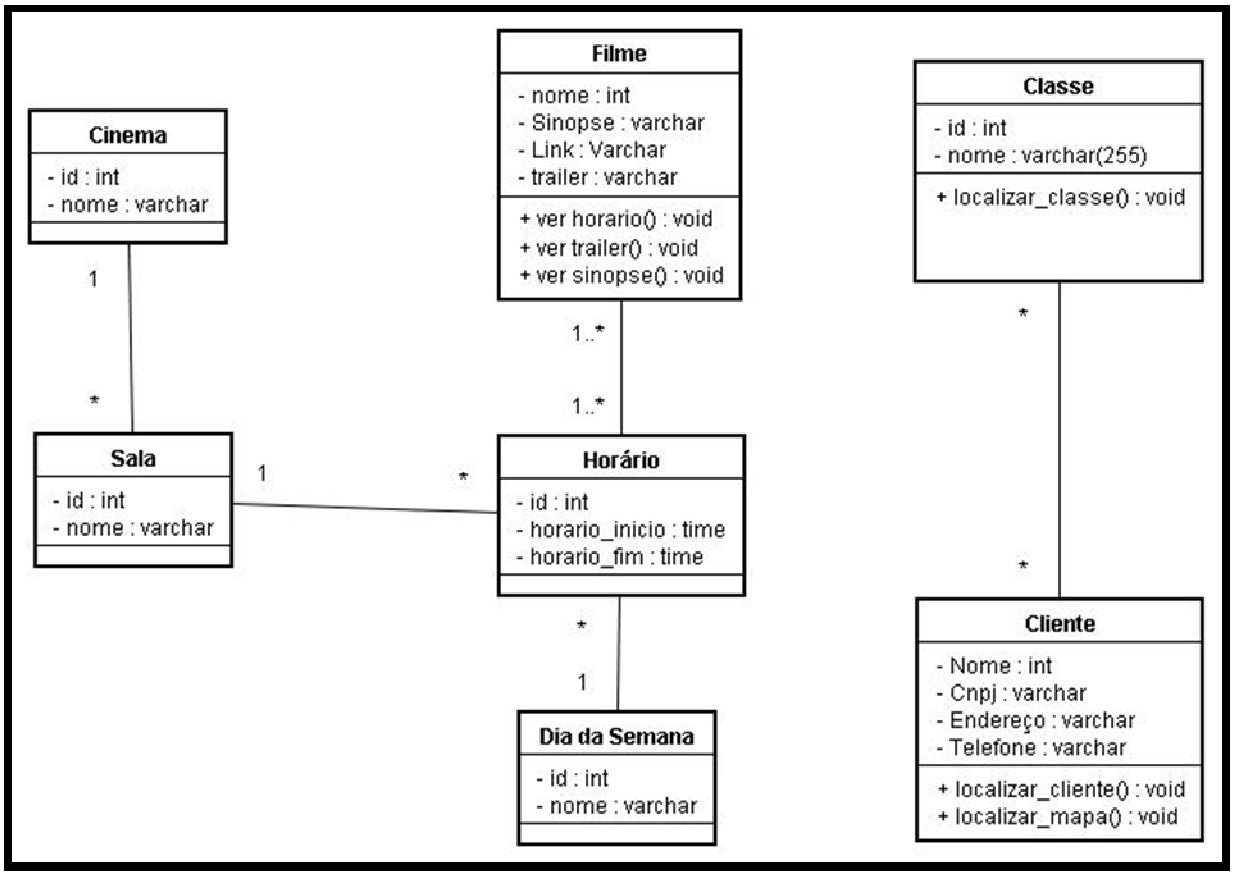

Figura 5.2 Diagrama de Classes Sistema T I 1.0

5.2.3 Casos de uso Sistema T I 1.0

Após a conclusão da modelagem e do diagrama de classes, foi iniciado o desenvolvimento dos diagramas de caso de uso. Cada caso de uso representa uma ação de um ator no sistema em um determinado cenário.

O sistema T I 1.0 é constituído de casos de uso que representam as funcionalidades que um ator pode realizar sobre o sistema. Os casos de uso foram documentados com seus fluxos principais, regras de negócios, pré-condições, pós-condições e fluxo de exceções.

Na figura 5.3 visualiza-se o Diagrama de Casos de uso Sistema T I 1.0. Esse diagrama demonstra de uma forma generalizada as funcionalidades pretendidas no sistema, e mostra com quais funcionalidades o usuário interage. São algumas funcionalidades como selecionar classe, ver filmes em cartaz, localizar clientes.

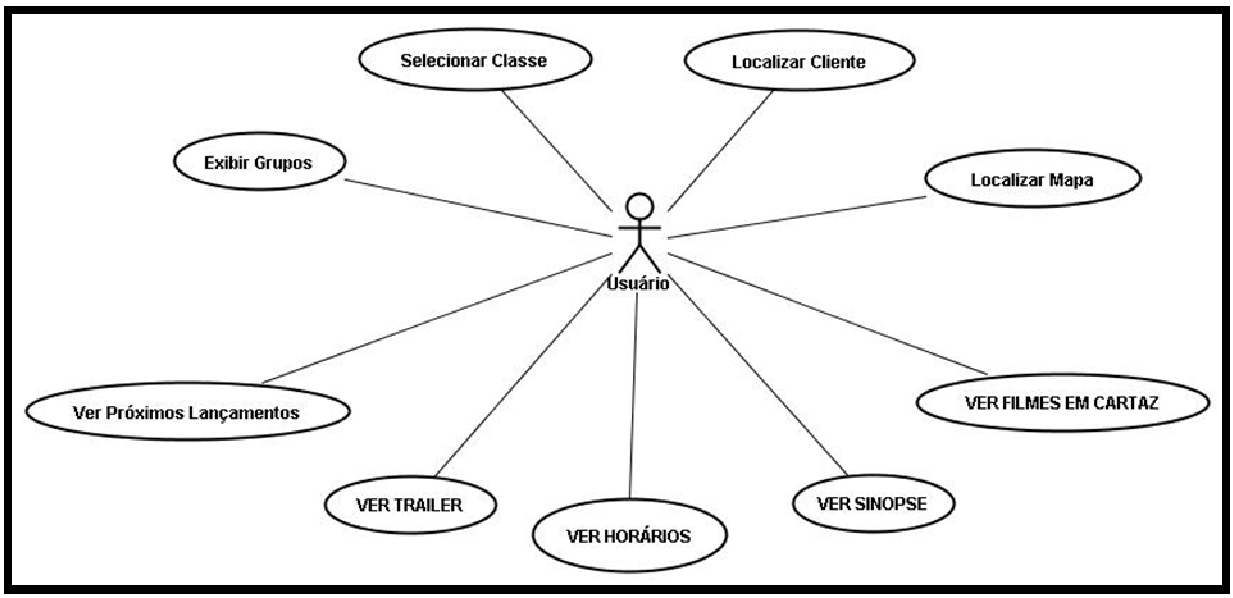

Figura 5.3 Diagrama de casos de uso do sistema T I 1.0

5.2.4 Arquitetura do Projeto

A hierarquia de pastas do sistema T I 1.0 foi desenvolvida de acordo com o padrão MVC. Nas figuras de 5.4 à 5.9 visualiza-se como este padrão é implementado.

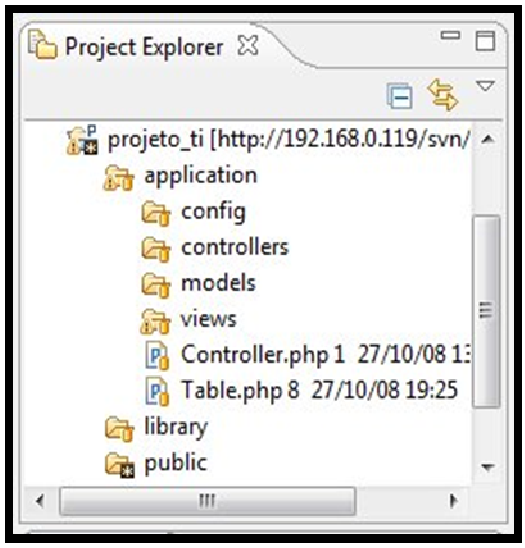

Figura 5.4 Estrutura de pastas do sistema T I 1.0

Na figura 5.4 visualiza-se como foram estruturadas as pastas do projeto, sendo que a application (aplicação) abriga a parte mais importante. A pasta “*Config*” tem as configurações de acesso ao Banco de Dados, de linguagens e demais configurações do projeto. E as pastas *models, controllers e views* seguem o padrão proposto pela Zend Framework.

Na figura 5.5 se pode verificar o interior da estrutura dos *controllers*.

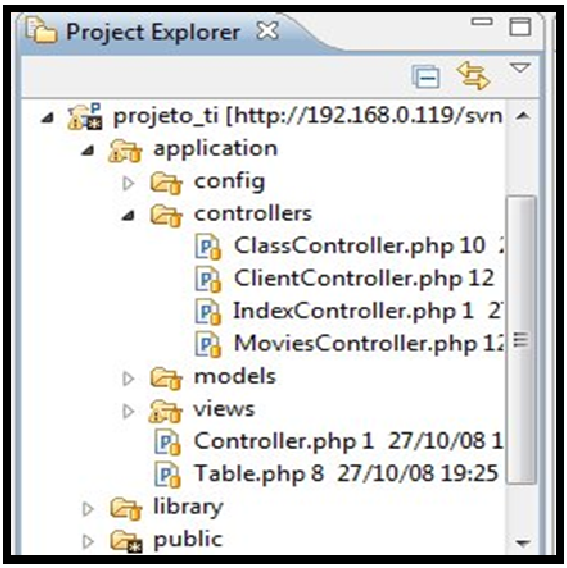

Figura 5.5 Estrutura dos *Controllers* do sistema T I 1.0

Os arquivos do interior do *controllers* possuem classes que gerenciam as requisições feitas pelos usuários do sistema, acionando de acordo com as necessidades, as *Models* dos respectivos métodos, Assim como os arquivos de visualização do sistema através das *Views*.

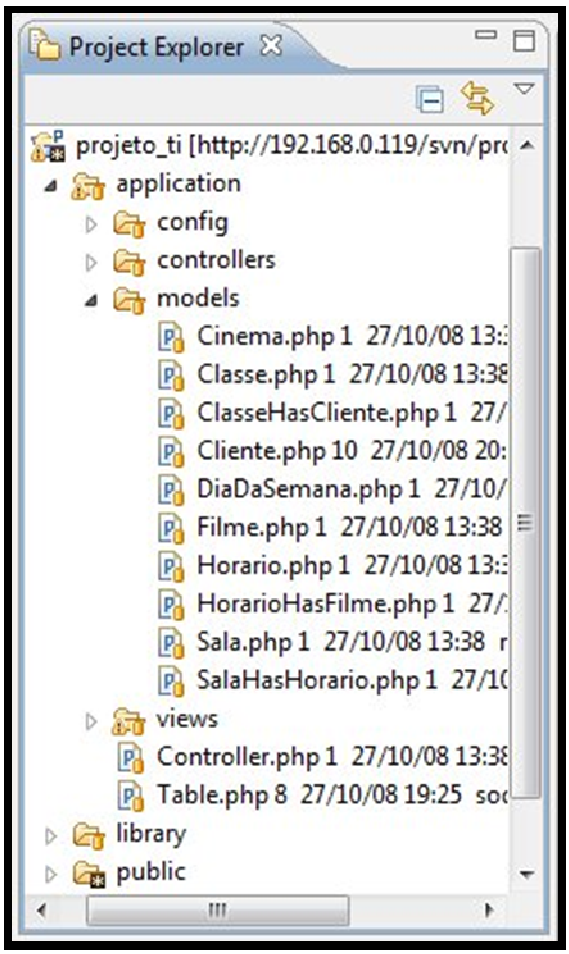

Figura 5.6 Estrutura dos *Models* do sistema T I 1.0

Na figura 5.6 pode-se verificar o interior da estrutura do *Models*, organizada conforme a modelagem do banco de dados, pois é no *Models* que se implementa as regras de negócios do sistema e que se faz a comunicação com o Banco de Dados.

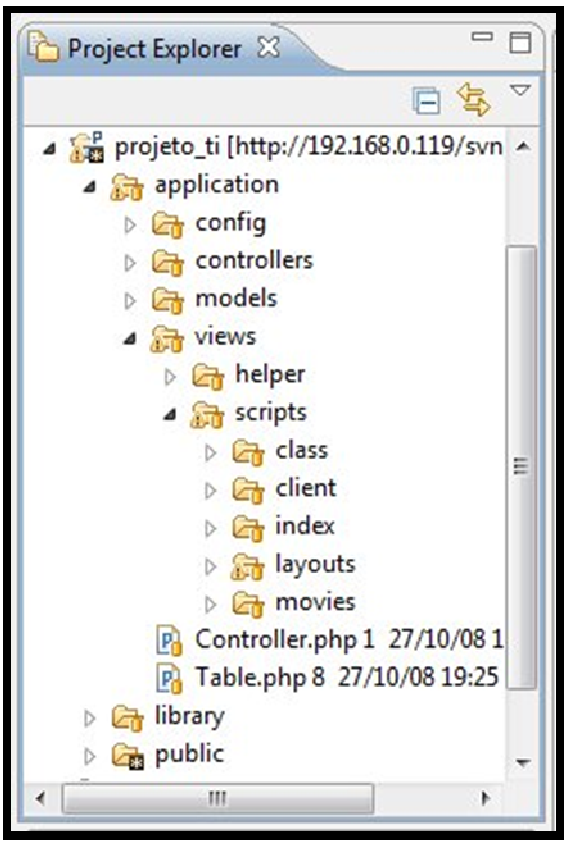

Figura 5.7 Estrutura das *Views* do sistema T I 1.0

A figura 5.7 demonstra como é composta a pasta *views*. A pasta *scripts* é onde se encontram as telas do sistema, e a pasta “*helper*” é onde arquiva-se *scripts* auxiliares comumente usados em diversas partes do sistema. Ambas são utilizadas pela *View*.

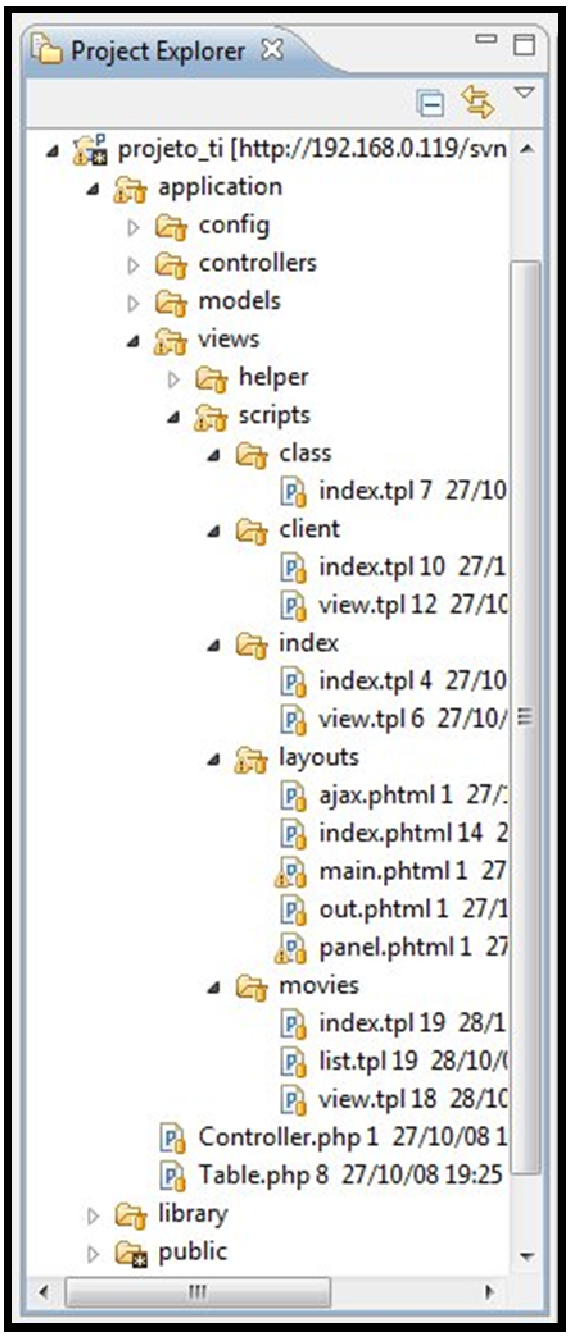

Figura 5.8 Estrutura Interna da pasta *View*

A figura 5.8 demonstra o interior da *views*, que contém os arquivos de saída de dados já processados pelos *controllers* e visualizados no *browser* (navegador).

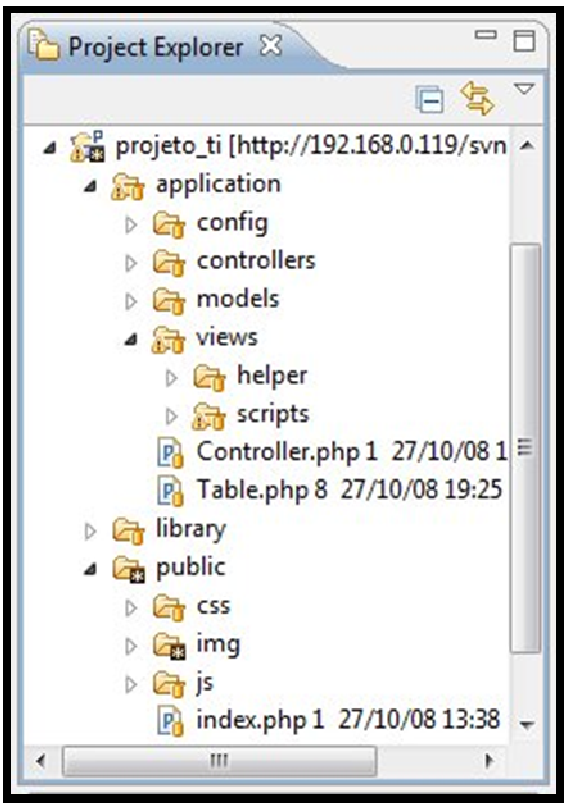

Figura 5.9/A Estrutura da pasta *public* do sistema T I 1.0

A figura 5.9/A mostra a única pasta acessada pelo usuário, a *Public*, que contém os arquivos de CSS, JavaScript e Imagens. E também onde é acessado o arquivo.php do sistema, o “índex.php”, que gerencia qual controller que será acessado.

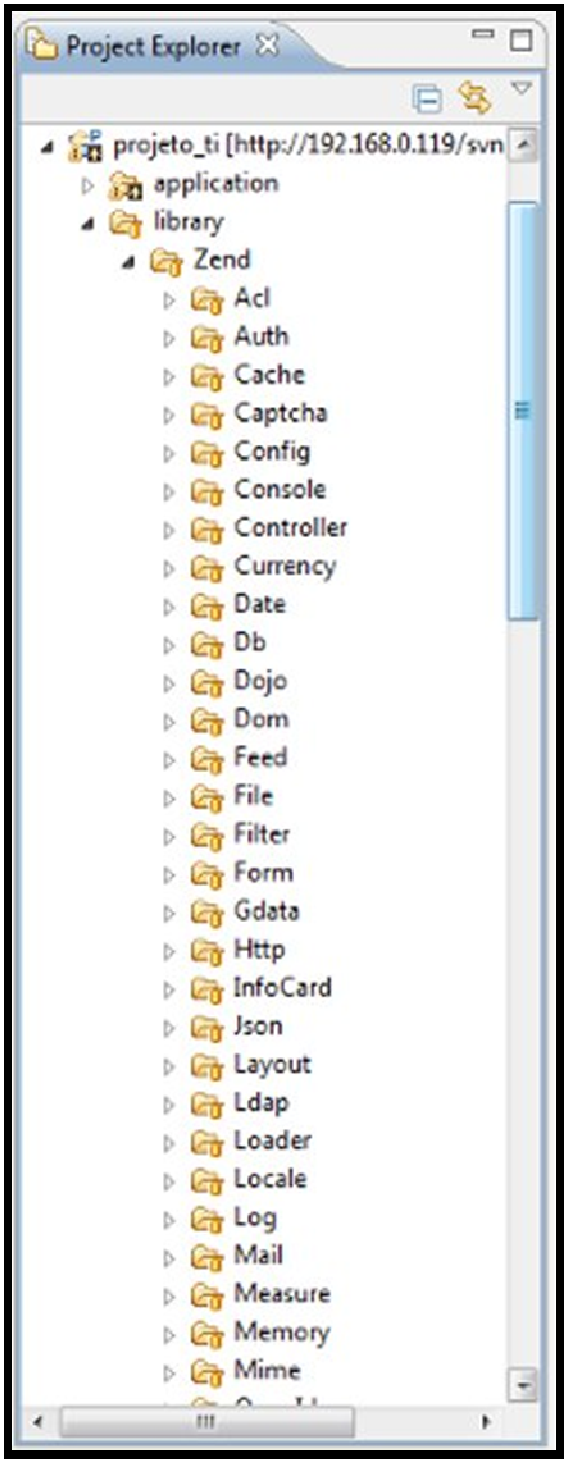

Figura 5.9/B Estrutura interna da pasta *Library* do sistema T I 1.0.

A pasta *library* contém a Zend Framework, cujas características e funcionalidades foram detalhadas neste projeto.

5.3 Fluxo Básico do Sistema T I 1.0

Neste capítulo é detalhado o fluxo básico do sistema através de telas. A figura 5.10 é a pagina inicial do totem. Esta pagina tem disponível um local centralizado para colocação de banners publicitários, o fluxo é descrito no caso de uso 5.3.1.

O uso de logotipos e imagens de instituições neste capítulo são meramente ilustrativas, não tendo vínculo algum com o projeto ou com quaisquer elementos que compõe o desenvolvimento deste.

Figura 5.10 Tela Index do sistema T I 1.0.

5.3.1 Caso de Uso “Exibir Grupos”.

FLUXOS DE EVENTOS

Fluxo Básico

- O usuário toca a tela do sistema T I 1.0.
- O sistema faz uma consulta no banco de dados e retorna com uma listagem de todos os grupos cadastrados.[E1]
- O sistema exibe uma lista com as opções na tela.
- O caso de uso é encerrado.

Fluxo de Exceção[E]

- **E1**-O usuário aciona a opção retornar.
  O sistema retorna para o fluxo básico do caso de uso.

A figura 5.11 é resultado da ação efetuada no caso de uso 5.3.1. A tela disponibiliza as opções que o usuário tem para interagir com o sistema. O usuário pode escolher entre informações e cinemas.

Figura 5.11 - Tela resultado do caso de uso exibir grupos do sistema T I 1.0.

5.3.2 Caso de uso “Localizar Classes”.

FLUXOS DE EVENTOS

Fluxo Básico

- O usuário escolhe a opção INFORMAÇÕES.[A1]
- O sistema faz uma consulta no banco de dados e retorna com uma listagem de todas as classes cadastradas para informações.[E1]
- O usuário escolhe a opção CINEMAS.[A2]
- O sistema faz uma consulta no banco de dados e retorna com todos os filmes em cartaz.[E1]
- O caso de uso é encerrado.

Fluxos Alternativos [A]

- **A1** - O usuário escolhe a opção Informações.
  O sistema exibe todas as classes cadastradas.
- **A2** – O usuário escolhe a opção Cinemas.
  O sistema exibe todos os filmes em cartaz.

Fluxo de Exceção

- **E1**-O usuário aciona a opção retornar.
  O sistema retorna para o fluxo básico do caso de uso.

5.4 Módulo Informações

O módulo informações é a opção de escolha quando o usuário seleciona o botão informações. A figura 5.12 mostra a ação de seleção do botão informações.

Figura 5.12 – Tela “Seleção de Informações” do sistema T I 1.0.

5.4.1 Caso de uso “Localizar Classes Informações”.

FLUXOS DE EVENTOS

Fluxo Básico

- O usuário escolhe a opção INFORMAÇÕES.[A1]

- O sistema faz uma consulta no banco de dados e retorna com uma listagem de todas as classes cadastradas para informações.[E1]
- O caso de uso é encerrado.

Fluxos Alternativos [A]

- **A1** – O usuário escolhe a opção Cinemas.
  O sistema exibe todos os filmes em cartaz.

Fluxo de Exceção

- **E1**-O usuário aciona a opção “Voltar”.
  O sistema retorna para a tela anterior.
  O caso de uso é encerrado.

A figura 5.13 é o resultado da ação do caso de uso 5.4.1. E mostra as opções disponíveis para localização e informações.

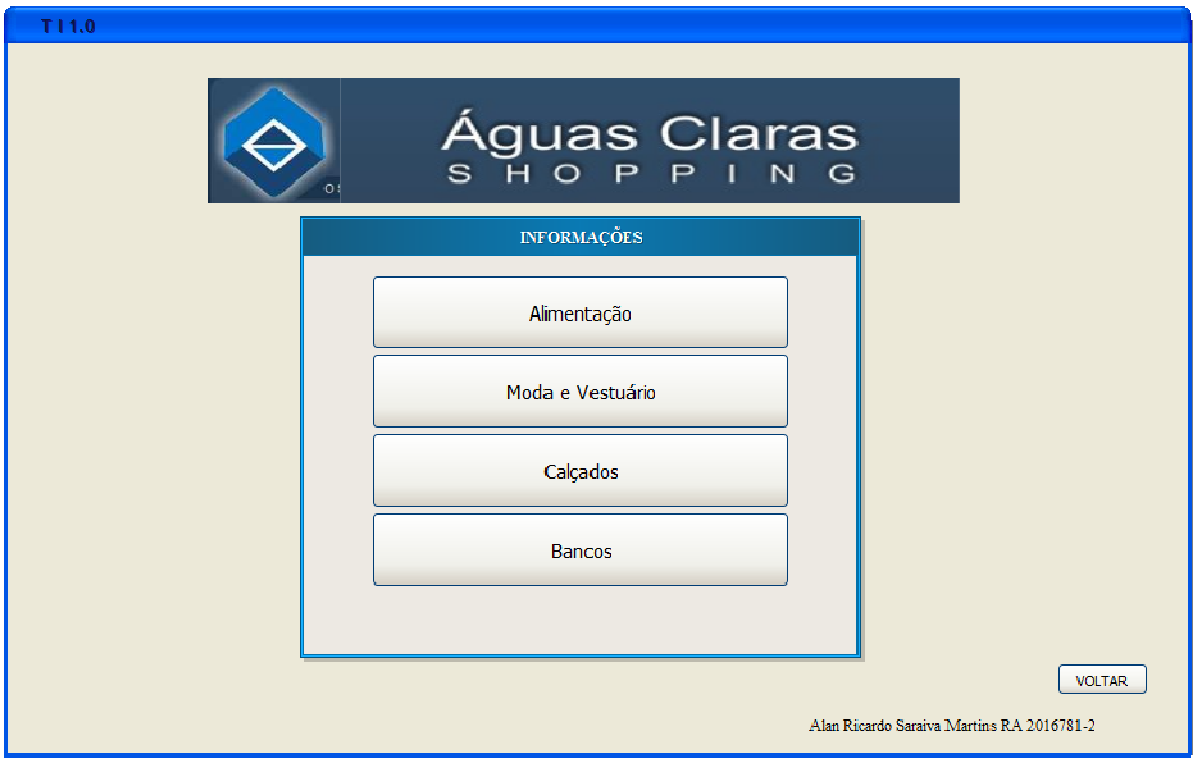

Figura 5.13 – Tela “resultado do U.C. Localizar Classe” do sistema T I 1.0.

A figura 5.14 mostra a seleção da opção alimentação, que é o exemplo usado na procura por localização. Esta seleção interage com sistema de acordo com o caso de uso 5.4.2.

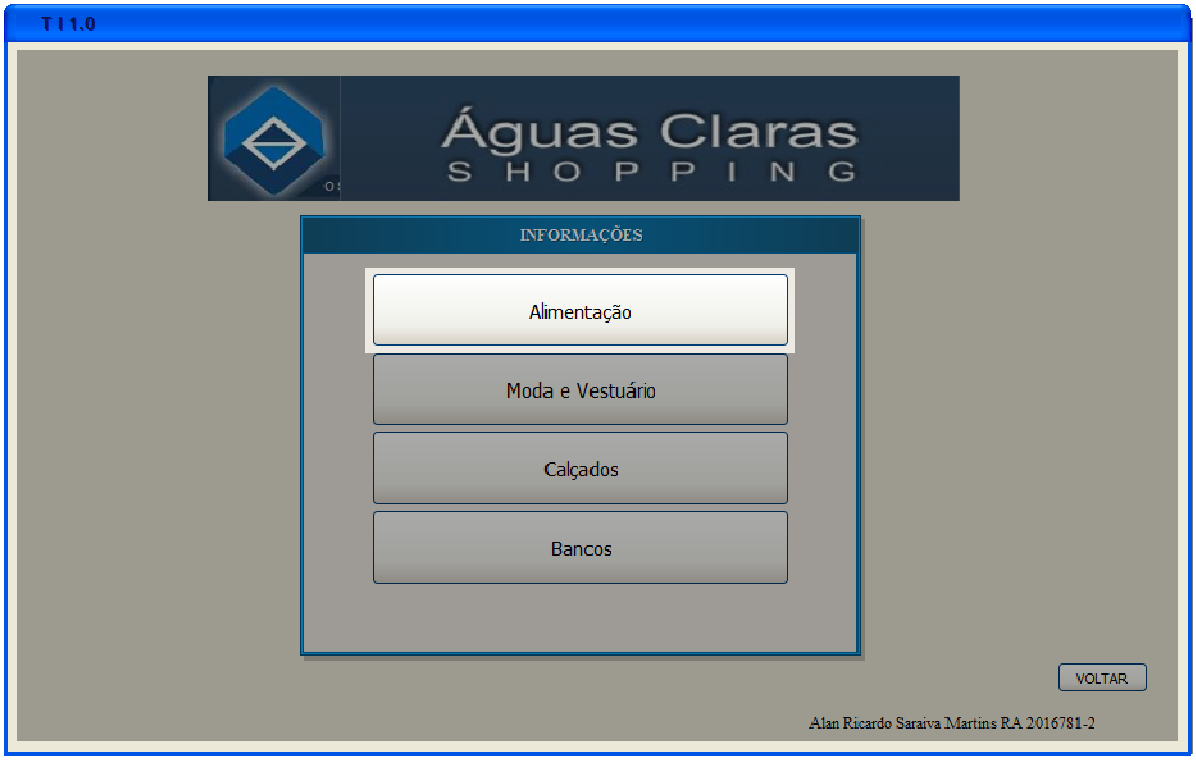

Figura 5.14 – Tela “Seleção de Classe” do sistema T I 1.0.

5.4.2 Caso de uso “Localizar Clientes”.

FLUXOS DE EVENTOS

Fluxo Básico

- O usuário aciona uma das classes listadas.[E1]
- O sistema faz uma consulta no banco de dados e retorna com uma listagem de todos os clientes cadastrados para determinada classe.
- O sistema exibe todos os clientes correspondentes.[E2][E1]
- O caso de uso é encerrado.

Fluxo de Exceção [E]

- **E1**-O usuário aciona a opção retornar.
  O sistema retorna para a tela anterior.
- **E2**-O usuário aciona a opção imprimir.
  O sistema envia uma solicitação de impressão.

A figura 5.15 mostra o resultado da ação do caso de uso 5.4.2. Visualiza-se uma listagem de todas as lojas cadastradas que participam da classe alimentação.

Figura 5.15 – Tela “resultado do U.C. Localizar Cliente” do sistema T I 1.0.

A figura 5.16 mostra a seleção da loja giraffas, que é uma das opções disponibilizadas na tela. E que aciona o caso de uso 5.4.3.

Figura 5.16 – Tela “Seleção de Cliente” do sistema T I 1.0.

5.4.3 Caso de uso “Localizar Mapa”.

FLUXOS DE EVENTOS

Fluxo Básico

- O usuário seleciona um dos clientes da lista.[E1]
- O sistema faz uma consulta no banco de dados e retorna com o mapa relacionado com determinado cliente.
- O sistema exibe o mapa correspondente.[E2][E1]
- O caso de uso é encerrado.

Fluxo de Exceção [E]

- **E1**-O usuário aciona a opção retornar.
  O sistema retorna para o fluxo básico.
- **E2**-O usuário aciona a opção imprimir.
  O sistema envia uma solicitação de impressão.

A figura 5.17 mostra o resultado do caso de uso 5.4.3, e visualiza no mapa do shopping a localização da loja Giraffas como exemplo. Ao final o usuário tem a opção de imprimir o endereço e a localização da loja chegando assim mais facilmente ao local.

Figura 5.17 – Tela resultado do U.C. “Localizar Mapa” do sistema T I 1.0.

5.5 Módulo Cinemas

O módulo cinemas é a opção de escolha quando o usuário seleciona o botão cinemas. A figura 5.18 mostra a seleção da opção cinemas visualizando assim as possíveis opções de interação do usuário com as informações relacionadas.

Figura 5.18 Tela “Seleção do Grupo Cinemas” do sistema T I 1.0.

A figura 5.19 é resultado da escolha pela opção cinemas, disponibilizam as diversas formas de interação que o usuário pode ter.

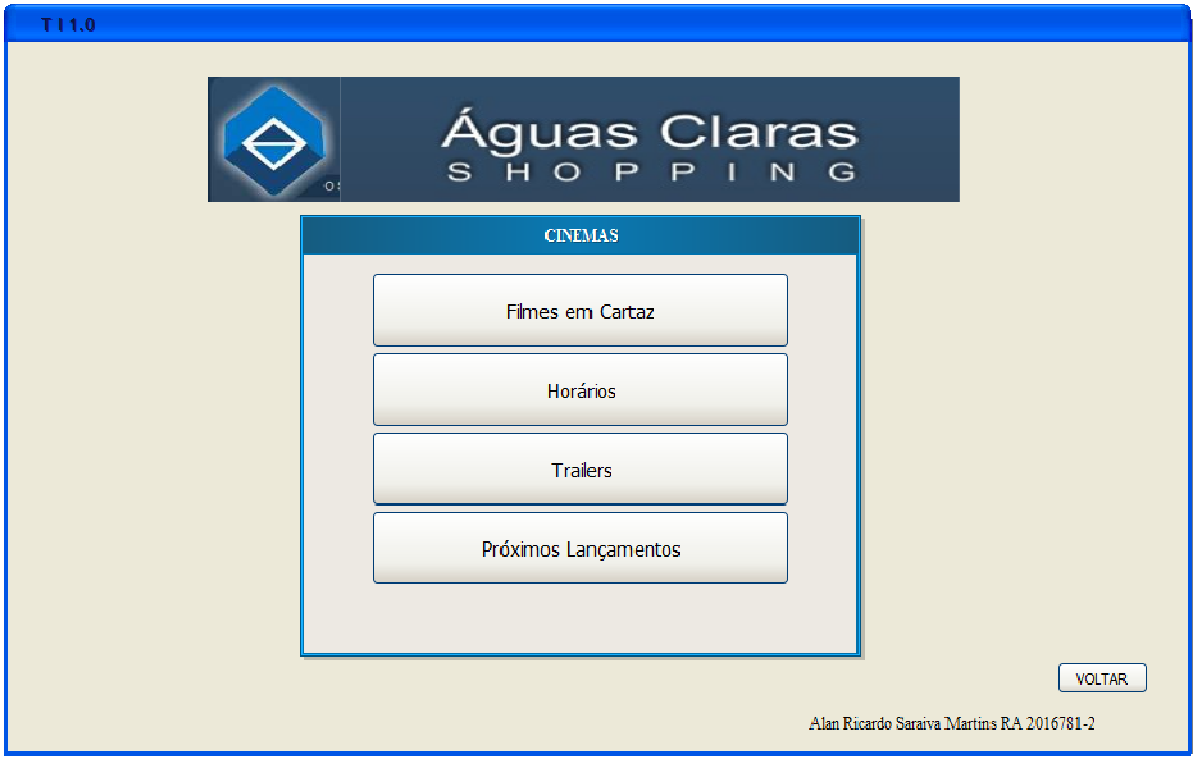

Figura 5.19 Tela “Localizar Classe Cinemas” do sistema T I 1.0.

Após a seleção do usuário, o sistema exibe a tela com as opções referentes a cinemas: **“Filmes em Cartaz”**, **“Horários”**, **“Trailers”**, **“Próximos Lançamentos”**.

A figura 5.20 mostra a escolha da opção filmes em cartaz como exemplo, acionando o caso de uso 5.5.1.

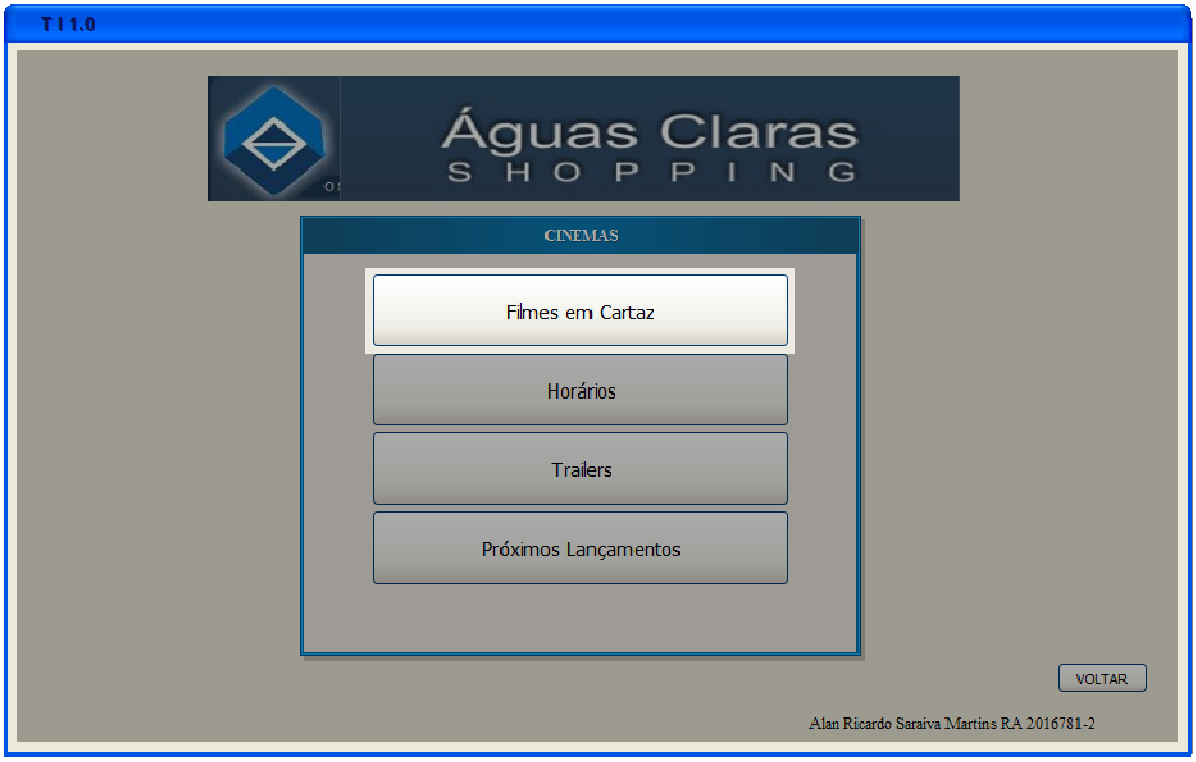

Figura 5.20 Tela “Seleção de Classe Cinema” do sistema T I 1.0.

5.5.1 Caso de uso “Ver Filmes em Cartaz”.

FLUXOS DE EVENTOS

Fluxo Básico

- O usuário aciona o opção “Filmes em Cartaz”.[E1]
- O sistema faz uma consulta no banco de dados e retorna com uma listagem de filmes em cartaz no determinado momento.
- O caso de uso é encerrado.

Fluxo de Exceção [E]

- **E1**-O usuário aciona a opção “Voltar”.
  O sistema retorna para a tela anterior.
  O caso de uso é encerrado.

A figura 5.21 mostra o resultado do caso de uso 5.5.1. e disponibiliza os filmes que estão em cartaz no cinema do shopping.

Figura 5.21 Tela “Resultado do U.C. “Ver Filmes em Cartaz” do sistema T I 1.0.

A figura 5.22 mostra a seleção de uma opção usada como exemplo, que é o filme Harry Portter. E aciona o caso de uso 5.5.2.

Figura 5.22 Tela “Selecionar Filmes” do sistema T I 1.0.

5.5.2 Caso de uso “Ver Sinopse do Filme”.

FLUXOS DE EVENTOS

Fluxo Básico

- O usuário seleciona um dos filmes listados na opção “Filmes em Cartaz”. [E1]
- O sistema faz uma consulta no banco de dados e retorna com um texto contendo a sinopse relacionada com o filme escolhido pelo usuário.
- O sistema exibe a visualização da sinopse na tela. [E2][E1][E3]
- O caso de uso é encerrado.

Fluxo de Exceção [E]

- **E1**-O usuário aciona a opção “Voltar”.
  O sistema retorna para a tela anterior.
- **E2**-O usuário aciona a opção Trailer.
  O sistema aciona o caso de uso “Visualizar Trailer”.

- **E3**-O usuário aciona a opção Horários.
  O sistema aciona o caso de uso “Visualizar Horários”.

A figura 5.23 exibe a sinopse do filme, resultado da escolha pelo usuário do filme.

Figura 5.23 Tela “Resultado do U.C. Ver Sinopse do Filme” do sistema T I 1.0.

5.5.3 Caso de uso “Ver Horários do Filme”.

FLUXOS DE EVENTOS

Fluxo Básico

- O usuário seleciona a opção “Horários”. [E1]
- O sistema faz uma consulta no banco de dados e retorna com a listagem de todos os filmes em cartaz e seus respectivos horários.
- O sistema exibe a visualização da listagem na tela. [E2][E1][E3]
- O caso de uso é encerrado.

Fluxo de Exceção [E]

- **E1**-O usuário aciona a opção “Voltar”.
  O sistema inicia novamente o caso de uso.
- **E2**-O usuário aciona a opção imprimir.

O sistema envia uma solicitação de impressão.

- **E3**-O usuário aciona um determinado filme.

O sistema aciona o caso de uso “Ver Sinopse dos Filmes em Cartaz”.

A figura 5.24 exibe o resultado do caso de uso 5.5.3 que é a visualização de horários do filme.

Figura 5.24 Tela “Resultado do U.C. Ver Horários do Filme” do sistema T I 1.0.

5.5.4 Caso de uso “Ver Trailer”.

FLUXOS DE EVENTOS

Fluxo Básico

- O usuário seleciona Trailer. [E1]
- O sistema faz uma consulta no banco de dados e retorna com um link do trailer relacionado com o filme.
- O usuário aciona a execução do trailer do filme relacionado.
- Após o encerramento do trailer.
- O caso de uso é encerrado.

Fluxo de Exceção [E]

- **E1**-O usuário aciona a opção “Voltar”.

O sistema retorna para a opção cinemas do fluxo básico.

A figura 5.25 é resultado da escolha pelo usuário da opção ver trailer do filme. O usuário poderá assistir ao trailer do filme ou retornar para opções anteriores.

Figura 5.25 Tela “Resultado do U.C. Ver Trailer do Filme” do sistema T I 1.0.

A figura 5.26 mostra a opção caso o usuário selecione próximos lançamentos, acionando o caso de uso 5.5.5; nesta tela o usuário terá uma visão geral dos próximos lançamentos e poderá com antecedência visualizar a sinopse e os trailers dos filmes.

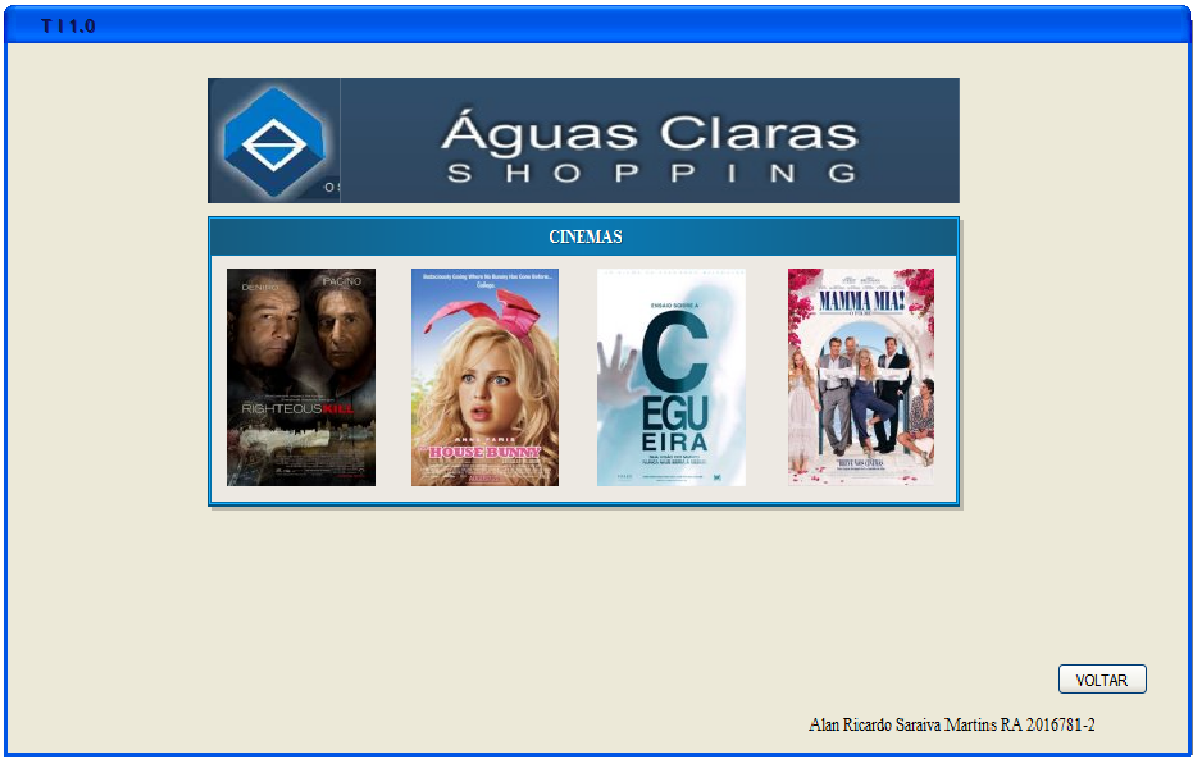

Figura 5.26 – Tela “Resultado do U.C. Ver Próximos Lançamentos” do sistema T I 1.0.

5.5.5 Caso de uso “Ver Próximos Lançamentos”.

FLUXOS DE EVENTOS

Fluxo Básico

- O usuário aciona próximos lançamentos da opção Cinemas. [E1]
- O sistema faz uma consulta no banco de dados e retorna com uma lista dos próximos filmes que serão lançados.
- O sistema exibe a visualização da lista na tela. [E1]
- O usuário seleciona um filme.
- O sistema exibe a sinopse do filme.
- O usuário aciona a execução do trailer do filme.
- O caso de uso é encerrado.

Fluxo de Exceção [E]

- **E1**-O usuário aciona a opção retornar.
  O sistema retorna para a opção cinemas do fluxo básico.

6 Conclusão

Apresentou-se proposta para desenvolvimento de protótipo que pudesse auxiliar a população a localizar-se no interior de *shopping centers* ou grandes centros comerciais. Durante o curso de Engenharia de Computação, um dos temas abordados é a inclusão digital na sociedade, e este projeto vai ao seu encontro com foco na usabilidade, identificação rápida dos comandos e interatividade.

Durante a etapa de referencial teórico, pesquisou-se ampla bibliografia, que serviu de sustentação para as argumentações apresentadas. O estudo se propôs a demonstrar uma forma viável de implementação, com fundamentação acadêmica, baseado em padrões atuais e boas práticas. Nas etapas de desenvolvimento do projeto foi possível se levantar as necessidades e como fazer para solucioná-las.

O uso da linguagem PHP foi fundamental na elaboração do sistema T I 1.0, pela facilidade de interação com o HTML, pela velocidade de execução, e boa flexibilidade e segurança na comunicação com o banco de dados. Além, claro, de ser uma linguagem *Open-Source*.

O desenvolvimento do protótipo decorreu conforme programado e os objetivos foram alcançados. O totem é uma realidade que pode ser empregada em vários centros comerciais e mesmo em complexos hospitalares e outros locais públicos ou privados onde for necessária a divulgação de informações.

O custo total do projeto sem a implementação do sistema ficou em torno de R$2200.00, um custo razoável levando em consideração a quantidade de produtos remanufaturados e de grande complexidade tecnológica, a relação específica dos custos encontra-se no apêndice C. Não foi determinado valores para o sistema T I 1.0, pois o mesmo foi implementado de forma prática e acadêmica.

6.1 Sugestão para projetos futuros

A implementação física do Totem interativo não necessita de futuras investidas. Por se tratar de equipamento desenvolvido exclusivamente para atendimento ao público, ele já traz o que de melhor se adapta às suas necessidades. Mas o sistema, sim, merece continuidade no desenvolvimento, pois com a mudança de requisitos, podem surgir carências de acoplamento de hardware que ofereçam suporte às próximas versões.

Em seguida serão propostas algumas soluções para complementação deste projeto:

- O acoplamento do Totem interativo com diversas formas de pagamento on-line, possibilitando assim a venda de ingressos para cinemas, o pagamento de cartão de estacionamento e situações assemelhadas.
- O desenvolvimento de módulo onde o usuário recebe algumas sugestões para a escolha de que presente comprar, após inserir algumas características da pessoa que vai receber o presente. Tudo isso de forma simples e interativa.
- O amadurecimento do software do sistema T I 1.0, extrapolando o espaço físico do shopping, e evoluindo para uma localização regional, com a colaboração de ferramentas como o GoogleMaps.
- A adaptação do sistema T I 1.0 para o uso de pessoas com deficiências físicas, auditivas e visuais.

Referências

DANTAS, Juliana; OLIVEIRA, Magno. ***SISTEMA DE VENDAS ON-LINE BASEADO EM PHP E MYSQL.*** 2006. 90f. Graduação (Bacharel) - FUNDAÇÃO UNIRG DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO, 2006. Disponível na URL: http://www.topguru.com.br/monografia_final_magno_e_juliana.pdf
Acesso em 25/10/2008.

DUARTE, Eber. ***Novidades da versão 5.0 do MySQL***. Disponível na URL: http://www.sqlmagazine.com.br/Colunistas/eber/13_mysql5.asp
Acesso em 13/10/2008.

FURLAN, José Davi**.** ***Modelagem de Objetos através da UML- the Unified Modeling Language****.* São Paulo: Makron Books, 1998.

"HowStuffWorks - Como os monitores touchscreen sabem que você os está tocando?". Publicado em 04 de setembro de 2001.
http://eletronicos.hsw.uol.com.br/questao716.htm. Acesso em 22/10/2008.

JACOBSON, Ivan. ***UML – Guia do Usuário.*** São Paulo: Editora Campus, 2000.

JUNJA, Manik. ***What is touch screen?*** Publicado em 18 de junho de 2001. http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci214510,00.html
Acesso em 22/10/2008.

MARCOS, Douglas. ***UML – Guia de Consulta Rápida****.* São Paulo: Editora Novatec, 2001.

NIEDERAUER, Juliano. ***Desenvolvendo Websites com PHP.***
São Paulo: Editora Novatec. 2004.

PESSOA, Marcio. ***Segurança em PHP:*****desenvolva programas PHP com alto nível de segurança e aprenda como manter os servidores web livres de ameaças.** São Paulo: Novatec Editora, 2007.

PRATES, Rubens; NIEDERAUER, Juliano. ***MySQL 5 GUIA DE CONSULTA RÁPIDA****.* São Paulo: Editora Novatec. 2006.

RUBIN, ERIK. ***Microsoft .Ner Compact Framework.*** Indianápolis: Editora Sams, 2003.

SANTOS, Antonio Raimundo dos, ***Metodologia Científica: a construção do conhecimento***. Rio de Janeiro: DP&A editora, 2000.

SOUZA M., Maxuel. ***Wirelless, Sistemas de Rede sem Fio***.
Rio de Janeiro: Editora Brasport. 2002.

SOUZA S., Marcelo; ***Análise do Servidor Web Apache em Clusters OpenMosix com Memória Compartilhada Distribuída.*** 2003. Centro Baiano de Computação de Alto Desempenho. Universidade Católica de Salvador.Salvador Acesso em 09/11/2008. Disponível na URL: http://inf.unisul.br/~ines/workcomp/cd/pdfs/2912.pdf

WELLING, Luke; THOMSON, Laura. ***Tutorial MySQL, uma introdução objetiva nos fundamentos do banco de dados MySQL.*** Rio de Janeiro: Editora Ciência Moderna Ltda. 2004.

WIKIPÉDIA ***– A enciclopédia livre*** - http://pt.wikipedia.org - Acesso em 03/11/2008.

YAHOO – ***YAHOO UI LIBRARY*** Acesso em 10/11/2008
Disponível na URL: http://developer.yahoo.com/yui/docs/module_event.html

ZENDFRAMEWORK, http://framework.zend.com – Acesso em 05/11/2008.

Apêndice – A: Fotos do totem

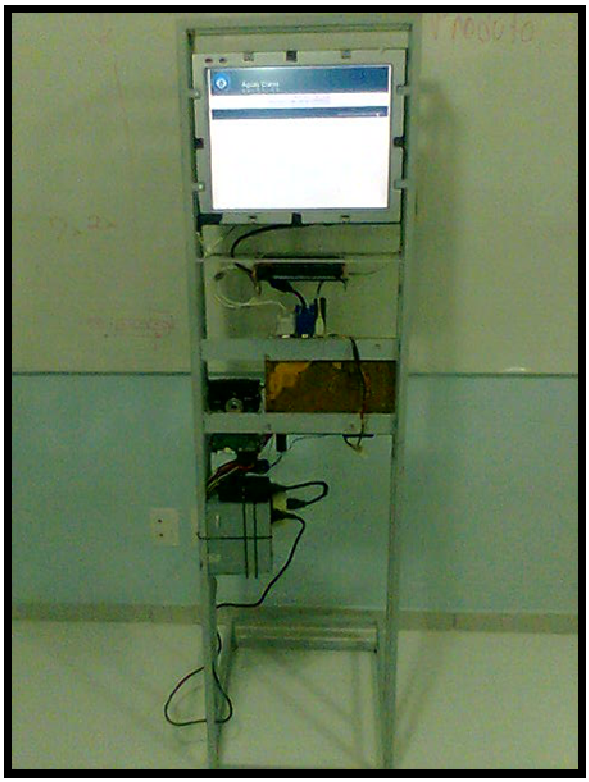

Figura A1: Vista frontal do protótipo

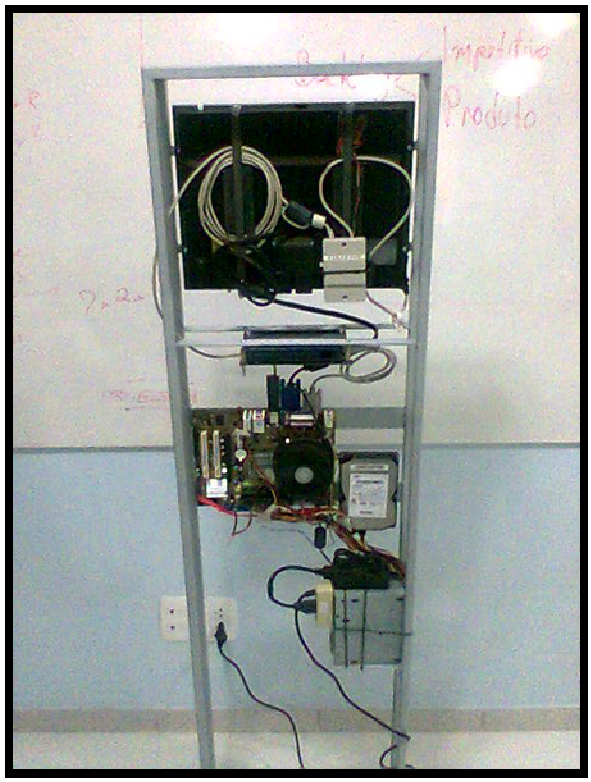

Figura A2: Vista por trás do protótipo

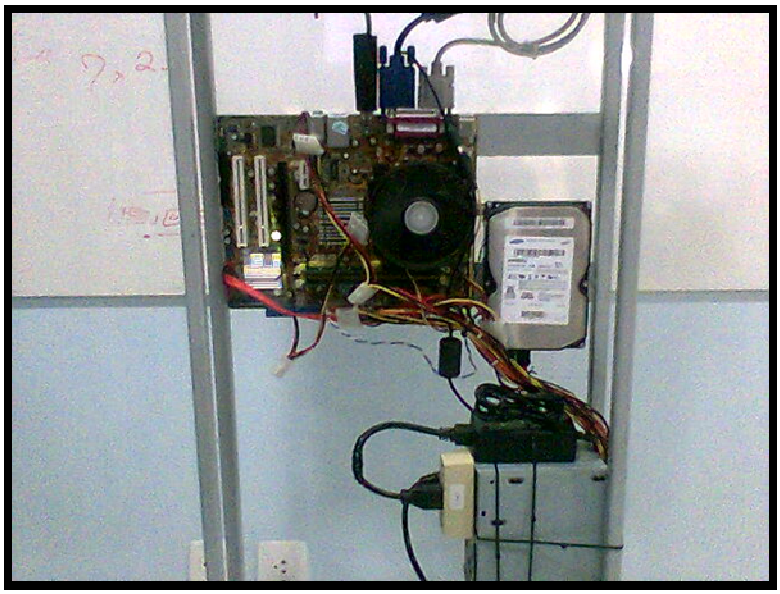

Figura A3: Vista do protótipo em detalhe

Apêndice – B: Custos do projeto

GABINETE/MÓVEL	
Estrutura/Armação	R$ 150,00
Parafusos	R$ 10,00
Fonte Energia	R$ 70,00
Transformador	R$ 30,00
Total	R$ 260,00
HARDWARE	
Placa Mãe	R$ 150,00
Memória	R$ 80,00
Processador	R$ 190,00
Hard Disk	R$ 130,00
Placa Wirelles	R$ 120,00
Teclado Wirelles	R$ 130,00
Impressora	R$ 290,00
Monitor	R$ 450,00
Tela Touchscreen	R$ 400,00
Caixas acústicas	R$ 35,00
Total	R$ 1.975,00
SOFTWARE	
Não estimado	
Total Geral	R$ 2.235,00

Apêndice – C: Código do sistema T I 1.0

CONFIG

FUNCTIONS.PHP

```
<?php
/**
 * @author Alan Ricardo <alan.saraiva@gmail.com>
 * @license http://www.fsf.org/licensing/licenses/gpl.txt
 *
 * @param string $sClass
 * @return void
 */
```

Função nativa do PHP que carrega dinamicamente os models, controllers.

Qualquer classe quando dá “new”.

```
function __autoload( $sClass )
{
	$defaultPath = array(
		DIR_LIBRARY ,
		DIR_APPLICATION,
		DIR_APPLICATION . 'models',
		DIR_APPLICATION . 'controllers'
	);
	Zend_Loader::loadClass( $sClass , $defaultPath );
}
```

Função que faz uma depuração do código, utilizada na implementação.

```
function debug( $mixPrint , $boolExit = 0 )
{
	echo "<fieldset><legend>Debug</legend><pre>";
	echo var_dump( $mixPrint );
	echo "<br /><br />";
	echo "</pre></fieldset><br />";
```

```
            if( $boolExit ) exit;
}
    Zend_Registry::get( 'logger' )->debug( $message );
}
Inclui arquivos recursivamente com exceção do svn
function includeRecurse($dirName)
{
  if(!is_dir($dirName))
      return false;
  $dirHandle = opendir($dirName);
  while(false !== ($incFile = readdir($dirHandle))) {
      if( $incFile != "." && $incFile != ".." && !strstr( $dirName , ".svn" ) )
      {
          if(is_file("$dirName/$incFile"))
              include_once("$dirName/$incFile");
          elseif(is_dir("$dirName/$incFile"))
              includeRecurse("$dirName/$incFile");
      }
  }
  closedir($dirHandle);
}
 ?>
```

CONFIG.PHP

```
<?php
/**
 * Configuracao geral do sistema.
 *
 * Este arquivo contem diretivas de configurações que afetam módulos distintos do
 * sistema. Comportamentos específicos dos componentes devem ser indicados em
 * seus respectivos arquivos de configuração.
```

```
 * @author Alan Ricardo <alan.saraiva@gmail.com>
 */
@ini_set( "error_reporting", E_ALL & ~E_NOTICE ); Trata os erros
@ini_set( "display_errors", 1 ); Exibe os erros
@ini_set( "display_startup_errors", 1 ); Capturar os erros
@ini_set( "html_errors", 0 ); Trata erros do html
@ini_set( 'session.cache_expire', 60 ); tempo da session
@ini_set( 'session.cookie_httponly' , true ); Configuração da session
@ini_set( 'session.use_only_cookie' , true ); persistencia da session
define( "THROW_EXEPTIONS" , true ); Diretivas definidas sobre excessesões se captura ou não
// Application Paths
define( "DIR_ROOT", realpath( dirname( __FILE__ ) . "/../.." ) . "/" ); definição de diretório raiz
define( "DIR_APPLICATION", DIR_ROOT . "application/" ); definição do diretorio da aplicação
define( "DIR_LIBRARY" , DIR_ROOT . "library/" ); definição do caminho da library
define( "DIR_CONFIG"  , DIR_APPLICATION . "config/" ); Definição dos diretorios da aplicação
define( "DIR_LANG"      , DIR_CONFIG . "languages/" );  arquivo de configuração das traduções
define( "DB_CONFIG"   , "db.ini" ); arquivo de configuração do banco
// Uniform Resources Locator
define( "URL", str_replace( "\\" , "/" , str_replace( realpath( $_SERVER['DOCUMENT_ROOT'] ) , "" ,
DIR_ROOT ) ) ); configuração de acesso da url
define( "URL_IMAGE", URL . "img/" ); carrega as imagens
define( "URL_JAVASCRIPT", URL . "js/" ); carrega javascript
define( "URL_STYLESHEET", URL . "css/" ); carrega css
// Others
define( "APP_CHARSET" , "UTF-8" ); codificação da saída do sistema
define( "APP_LOCALIZATION" , "pt_BR" ); configuração do php
// Include Path
ini_set( "include_path" , ini_get( "include_path" ) . PATH_SEPARATOR . DIR_LIBRARY ); onde os
diretorios podem dar include.
ini_set( "default_charset" , APP_CHARSET );
```

DB.INI

```
[main]

db.adapter = PDO_MySql

db.config.username = root

db.config.dbname = bd_ti

db.config.password =

db.config.host = localhost
```

HTACCESS

```
RewriteEngine on

RewriteRule    ^$ public/    [L]

RewriteRule    (.*) public/$1 [L]

RewriteEngine on

HTACCESS PUBLIC

AddOutputFilterByType DEFLATE text/html text/css text/javascript application/x-javascript

RewriteRule !\.(js|css|ico|txt|gif|jpg|png|swf|xml|mp3|html|htm|pdf|doc|JPG|avi|wmv|mov|wma)$
index.php
```

INDEX.PHP

```
<?php

        include( '../application/config/functions.php' );

        include( '../application/config/config.php' );

        include( 'Zend/Loader.php' );

        Zend_Session::start();

        $front = Zend_Controller_Front::getInstance();

        $front->setBaseUrl( "/projeto_ti/" );

        $front->throwExceptions( THROW_EXEPTIONS );

        $front->setControllerDirectory( array(

            'default'   => DIR_APPLICATION . 'controllers',

        ));

        try{
```

```
                $front->dispatch( $front->getRequest() , $front->getResponse() );
        }
        catch( Exception $e )
        {
                if( VIEW_EXEPTIONS )
                        debug( $e );
        }
?>
```

CONTROLLER.PHP

```
?php
/**
 * Classe abstrata que herda da Zend_Controller_Action
 * Responsável por construir regras especificas da aplicação
 *
 * @author Alan Ricardo <alan.saraiva@gmail.com>
 * @category Toten
 * @version 1.0
 */
abstract class Controller extends Zend_Controller_Action
{
        /**
         * Inicialização dos models utilizados nas classes finais
         * @var array
         */
        public $uses = array();
        /**
        * Função responsável pelo construtor da classe,
        * que sobrescreve a função de mesmo nome na zend
        */
```

```
	public function init()
	{
		$this->initDb();
		if( file_exists( DIR_LANG . 'lang.' . APP_LOCALIZATION ) )
		{
			$translate = new Zend_Translate( 'csv' , DIR_LANG . 'lang.' .
APP_LOCALIZATION );
			Zend_Registry::set( 'Zend_Translate' , $translate );
		}
		if( $this->uses )
			foreach( $this->uses as $use )
				$this->$use = new $use();
		if ( $this->components )
		{
			foreach ( $this->components as $component )
			{
				Zend_Loader::loadClass( $component , $component );
				$this->$component = new $component();
			}
		}
		$logger = new Zend_Log();
		$writer = new Zend_Log_Writer_Firebug();
		$logger->addWriter( $writer );
		Zend_Registry::set( 'logger' , $logger );
		$this->initViewAttributes();
	}
	/**
	 * Função responsável por renderizar os templates no layout default
	 * @param string $action
	 * @param string $template
	 */
	public function render( $action = null , $template = "index" )
```

```
        {
                $Front = Zend_Controller_Front::getInstance();
                $action = $action ? $action : $Front->getRequest()->getActionName();
                $this->view->content_layout = $this->view->render( $action . ".tpl" );
                parent::render( $template , null , true );
        }
        /**
         * Função responsável pela conexão com o banco de dados
         */
        protected function initDb()
        {
                $config = new Zend_Config_Ini( DIR_CONFIG . DB_CONFIG , "main" );
                $db = Zend_Db::factory( $config->db->adapter , $config->db->config->toArray() );
                $profiler = new Zend_Db_Profiler_Firebug('All DB Queries');
                $profiler->setEnabled(true);
                $db->setProfiler( $profiler );
                Zend_Registry::set( 'db', $db );
        }
        /**
         * Função responsável pela configuração dos templates e layouts do objeto view
         */
        protected function initViewAttributes()
        {
                $Front = Zend_Controller_Front::getInstance();
                $controller = $Front->getRequest()->getControllerName();
                $action = $Front->getRequest()->getActionName();
                $module                 = $Front->getRequest()->getModuleName();
                $this->view->addScriptPath( DIR_APPLICATION . "views/scripts/layouts" );
                $this->view->addScriptPath( DIR_APPLICATION . "views/scripts/{$controller}" );
                $this->view->addHelperPath( DIR_APPLICATION . "views/helper" ,
"Preceptor_View_Helper_" );
                $this->view->data               = new stdClass();
```

```
		$this->view->url 			= $Front->getRequest()->getBaseUrl();
		$this->view->controller = $controller;
		$this->view->action 	= $action;
		$this->view->module 	= $module;
		$this->view->messages 		= $this->_helper->_flashMmessenger->getMessages();
}
/**
 * Função responsável por receber uma requisição inicial
 */
public function indexAction()
{
		$model = $this->uses[0];
		$this->view->rs = $this->$model->fetchAll();
		$this->render();
}
/**
 * Função responsável pela visualização individual de objetos
 */
public function viewAction()
{
		$id = Zend_Filter::get( $this->_getParam( "id" ) , "int" );
		$model = $this->uses[0];
		if( $id )
			$this->view->rs = $this->$model->find( $id )->current();
		$this->render();
}
/**
 * Função responsável por agrupar no array os objetos duplicados
 *
 * @param array|object $data
 * @param string $value
```

```
	 * @param string $label
	 * @param string $first
	 * @return array
	 */
	public function toSelect( $data , $value = "id" , $label = "name" , $first = "Selecione ..." )
	{
		if( is_object( $data ) )
			$data = $data->toArray();
		if( $first )
			$select[""] = $first;
		foreach ( $data as $key => $val )
		{
			$select[$val[$value]] = $val[$label];
		}
		return $select;
	}
}
```

TABLE.PHP

```
<?php
/**
 * Classe abstrata para manipulacao de dados em uma tabela
 *
 * @package Models
 * @author Preceptor Educacao a distancia
 * @version 1.0
 */
abstract class Table extends Zend_Db_Table_Abstract
{
	private $datas = array();
	protected function _setup()
```

```
        {
                parent::setDefaultAdapter( Zend_Registry::get( 'db' ) );
                parent::_setup();
        }
        public function __call( $name , $values )
        {
                $name  = $this->underscore( preg_replace( '/findBy/i', '', $name ) );
                $select = $this->select();
                $select->from( $this->_name , '*' )
                        ->where( $name . " = ?" , $values[0] );
                return $this->fetchAll( $select );
        }
    }
```

CONTROLLERS

INDEXCONTROLLER.PHP

```
<?php
/**
 * IndexController responsável pelo controle de requisições
 *
 * @author Alan Ricardo <alan.saraiva@gmail.com>
 * @category Toten
 * @version 1.0
 */
class IndexController extends Controller
{
        /**
         * Comentada na Classe principal "Controller.php"
         *
         * @access public
         * @return void
```

```
	 */
	public function indexAction()
	{
		$this->render();
	}
	/**
	 * Comentada na classe principal
	 *
	 * @access public
	 * @return void
	 */
	public function viewAction()
	{
		$this->render( null  , "index");
	}
}
```

CLASSCONTROLLER.PHP

```
<?php
/**
 * ClassController responsável pelo controle de requisições
 *
 * @author Alan Ricardo <alan.saraiva@gmail.com>
 * @category Toten
 * @version 1.0
 */
class ClassController extends Controller
{
	public $uses = array( "Classe" );
	/**
	 * @access public
```

```
         * @return void
         */
        public function indexAction()
        {
                /**
                 * Comentado na classe principal "Controller.php"
                 */
                $this->view->class = $this->Classe->fetchAll();

                $this->render();
        }
}
```

CLIENTECONTROLLER.PHP

```
<?php
/**
 * ClientController responsável pelo controle de requisições
 *
 * @author Alan Ricardo <alan.saraiva@gmail.com>
 * @category Toten
 * @version 1.0
 */
class ClientController extends Controller
{
        /**
         * Comentado na classe principal "Controller.php"
         *
         * @var array
         */
        public $uses = array( "Cliente" , "ClasseHasCliente" );
        /**
```

```
        * Função comentada na classe principal "Controller.php"
        *
        * @access public
        * @return void
        */
        public function indexAction()
        {
                $id = Zend_Filter::get( $this->_getParam( "id" ) , "int" );
                $this->view->rs = $this->Cliente->fetchClientByClass( $id );
                $this->render();
        }
        /**
        * Comentado na classe principal "Controller.php"
        *
        */
        public function viewAction()
        {
                $id = Zend_Filter::get( $this->_getParam( "id" ) , "int" );
                $this->view->client = $this->Cliente->fetchRow( array( "id =?" => $id ) );
                $this->render();
        }
}
```

MOVIESCONTROLLER.PHP

```
<?php
/**
 * MoviesController responsável pelo controle de requisições
 *
 * @author Alan Ricardo <alan.saraiva@gmail.com>
 * @category Toten
 * @version 1.0
```

```
 */
class MoviesController extends Controller
{
	/**
	 * Função comentada na classe principal "Controller.php"
	 *
	 * @var array
	 */
	public $uses = array( "Filme" );
	/**
	 * Função comentada na classe principal
	 *
	 */
	public function indexAction()
	{
		$this->render();
	}
	/**
	 * Função responsável pela listagem de todos os filmes e horários
	 *
	 */
	public function listAction()
	{
		$status = Zend_Filter::get( $this->_getParam( "status" ) , 'HtmlEntities' );
		if( $status != "P" && $status != "A" )
			$this->_redirect( "/" );
		$this->view->rs = $this->Filme->showHours( $status );
		$this->render();
	}
	/**
	 * Função comentada na classe principal
```

```
        *
        */
        public function viewAction()
        {
                $id = Zend_Filter::get( $this->_getParam( "id" ) , "int" );
                $this->view->movie = $this->Filme->fetchRow( array( "id =?" => $id ) );
                $this->render();
        }
        /**
        * Função responsável por localizar o trailer do filme requisitado
        *
        */
        public function trailersAction()
        {
                $id = Zend_Filter::get( $this->_getParam( "id" ) , "int" );
                $this->view->movie = $this->Filme->fetchRow( array( "id =?" => $id ) );
                $this->render();
        }
}
?>
```

MODEL

CINEMA.PHP

```
<?php
class Cinema extends Table
{
        protected $_name = 'cinema';
        protected $_primary = 'id';
}
```

CLASSE.PHP

```
<?php
class Classe extends Table
{
	protected $_name = 'classe';
	protected $_primary = 'id';
}
```

CLASSHASCLIENTE.PHP

```
<?php
class ClasseHasCliente extends Table
{
	protected $_name = 'classe_has_cliente';
	protected $_primary = array( 'classe_id' , 'cliente_id' );
}
```

CLIENTE.PHP

```
<?php
class Cliente extends Table
{
	protected $_name = 'cliente';
	protected $_primary = 'id';

	public function fetchClientByClass( $id )
	{
		$select = $this->select();
		$select->from( array( "c" => $this->_name ) , "*" )
			->join( array( "cc" => "classe_has_cliente" ) , "c.id = cc.cliente_id" , array() )
				->where( "classe_id =?" , $id );
		return $this->fetchAll( $select );
	}
}
```

DIADASEMANA.PHP

```
<?php
class DiaDaSemana extends Table
{
	protected $_name = 'dia_da_semana';
	protected $_primary = 'id';
}
```

FILME.PHP

```
<?php
class Filme extends Table
{
	protected $_name = 'filme';
	protected $_primary = 'id';
	public function showHours( $status )
	{
		$weekDay = getdate( time() );
		$select = $this->select();
		$select->from( array( "f" => "filme" ) , new Zend_Db_Expr( "f.* , f.id AS Filme_id , f.nome AS Filme_nome , h.* , s.*" ) )
			->joinLeft( array( "hf" => "horario_has_filme" ) , "f.id = hf.filme_id" , array()  )
			->joinLeft( array( "h" => "horario" ) , "h.id = hf.horario_id" , array() )
			->joinLeft( array( "s" => "sala" ) ,  "s.id = hf.sala_id" , array() )
			->where( "f.status = ?" , $status )
			->order( array( "f.nome"  , "h.horario_inicio" , "h.horario_fim" )  );
		if( $status == "A" )
			$select->where( "dia_da_semana_id = ?" , intval( $weekDay['wday'] ) + 1 );
		$hourMovies = $this->fetchAll( $select );
		$data = array();
		foreach( $hourMovies as $key => $hourMovie )
```

```
                {
                        $data[$hourMovie->Filme_id]['name'] = $hourMovie->Filme_nome;
                        $data[$hourMovie->Filme_id]['image'] = $hourMovie->image;
                        $data[$hourMovie->Filme_id]['id']   = $hourMovie->Filme_id;
                        if( $hourMovie->horario_inicio )
                                $data[$hourMovie->Filme_id]['hour'] = true;
                        $data[$hourMovie->Filme_id]['hours'][$key] = array(
                        'start' => $hourMovie->horario_inicio,
                        'end'   => $hourMovie->horario_fim,
                        'room'      => $hourMovie->nome
                        );
                }
                return $data;
        }
}
```

HORARIO.PHP

```
<?php
class Horario extends Table
{
        protected $_name = 'horario';
        protected $_primary = 'id';
}
```

HORARIOHASFILME.PHP

```
<?php
class HorarioHasFilme extends Table
{
        protected $_name = 'horario_has_filme';
        protected $_primary = array( 'horario_id' , 'filme_id' );
}
```

SALA.PHP

```
<?php

class Sala extends Table

{

	protected $_name = 'sala';

	protected $_primary = 'id';

}
```

SALAHASHORARIO.PHP

```
<?php

class SalaHasHorario extends Table

{

	protected $_name = 'sala_has_horario';

	protected $_primary = array( 'sala_id' , 'horario_id' );

}
```

VIEW

SCRIPTS/CLASS

INDEX.TPL

```
<?php foreach ( $this->class as $class ) : ?>

	<?=$this->formButton( "bt-class" , utf8_encode( $class->nome ) , array( "class" => "large-button" , "onclick" => "window.location = '{$this->url}/client/index/id/{$class->id}';") )  ?><br />

<?php endforeach; ?>
```

SCRIPTS/CLIENT

INDEX.TPL

```
<div class="box">

	<div class="box-header">

		<h1><?=$this->translate( "client" )?></h1>

	</div>

	<div class="box-content">
```

```
                <?php $cont=0; foreach ( $this->rs as $client ) : ?>

                        <div class="left"><a href="<?=$this->url?>/client/view/id/<?=$client->id ?>"
><?=$this->image( "logos/" . $client->logo  , array("width" => "150")) ?></a> </div>

                <?php endforeach; ?>

                <div class="clear"></div>

        </div>

</div>
```

VIEW.TPL

```
<?=$this->image( "mapas/" . $this->client->mapa, array("width" => "1000" )) ?>

<br />

<div id="content-print">

        <p>Nome: <?=$this->client->nome?></p>

        <p>Endereço: <?=$this->client->endereco?></p>

        <p>Telefone: <?=$this->client->telefone?></p>

</div>

<?=$this->formButton( "print" , $this->translate( "print" ) , array( "class" => "large-button" , "onclick" =>
"window.print()" ) ) ?>
```

SCRIPTS/INDEX

INDEX.TPL

```
<?=$this->formButton( "bt-start" , $this->translate( "start" ) , array( "style" => "height: 50px;") ) ?>
```

VIEW.TPL

```
<?=$this->formButton( "info"  , $this->translate( "info" ) , array( "class" => "large-button" , "onclick" =>
"window.location = '{$this->url}/class/';" ) )?>

<?=$this->formButton( "movies"  , $this->translate( "movies" ) ,  array( "class" => "large-button" ,
"onclick" => "window.location = '{$this->url}/movies/';")
```

SCRIPTS/LAYOUT

INDEX.PHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>Sistema TI 1.0</title>
                  <meta http-equiv="Content-Type" content="text/html;
charset=<?=APP_CHARSET?>"/>
                  <?=$this->style( "../js/yahoo/reset-fonts-grids/reset-fonts-grids" )?>
                  <?=$this->style( "../js/yahoo/fonts/fonts-min" )?>
                  <?=$this->style( "../js/yahoo/container/assets/skins/sam/container" )?>
                  <?=$this->style( "main" )?>
                  <link href="<?=URL_STYLESHEET?>print.css" rel="stylesheet" media="print" />
   </head>
   <body class="yui-skin-sam">
         <div id="doc3">
         <div id="hd">
         <a href="<?=$this->url ?>"><?=$this->image( "aguasclaras.jpg" )?></a>
         <a href="<?=$this->url ?>"><?=$this->image( "logo.jpg" )?></a>
         </div>
         <div id="bd" align="center">
                              <?=$this->content_layout?>
         </div>
         <div id="ft" align="right">
                  <?php if( ( $this->controller != "index" ) || ( $this->action != "index" ) ):?>
                                    <?=$this->formButton( "back" , $this->translate(
"back" ) , array( "class" => "large-button" , "onclick" => "history.back();" ) )?>
                           <?php else: ?>
                                    <script>
                                             windowAll = true;
                                    </script>
                                    <br />
                           <?php endif ?>
                     <p><?=$this->translate( "author name" )?> <?=$this->translate( "register
academic" )?> <?=$this->translate( "register academic author" )?>    </p>
```

```
                </div>
        </div>
                <?=$this->script( "yahoo/yahoo-dom-event/yahoo-dom-event" )?>
                <?=$this->script( "yahoo/yahoo/yahoo-min" )?>
                <?=$this->script( "yahoo/connection/connection-min" )?>
                <?=$this->script( "yahoo/element/element-beta-min" )?>
                <?=$this->script( "yahoo/datasource/datasource-beta-min" )?>
                <?=$this->script( "preceptor/preceptor" )?>
                <?=$this->script( "preceptor/string/string" )?>
                <?=$this->script( "preceptor/dom/dom" )?>
                <?=$this->script( "preceptor/ajax/ajax" )?>
                <?=$this->script( "preceptor/toggle/toggle" )?>
                <?=$this->script( "preceptor/layout/layout" )?>
                <script>
                        YAHOO.util.Event.onDOMReady( function()
                        {
                                if( windowAll )
                                {
                                        YAHOO.util.Event.on( document , 'click' , function(ev){
                                                window.location = '<?=$this->url?>/index/view';
                                        } );
                                }
                        });
                </script>
    </body>
</html>
```

SCRIPTS/MOVIES

INDEX.TPL

```
<div class="box">
        <div class="box-header">
```

```
            <h1><?=$this->translate( "movies" )?></h1>

        </div>

        <div class="box-content">

            <?=$this->formButton( "bt-movies" , $this->translate( "movie poster" ) , array( "class" => "large-button" , "onclick" => "window.location = '{$this->url}/movies/list/status/A';") )  ?><br />

            <?=$this->formButton( "bt-next_releases" , $this->translate( "next releases" ) , array( "class" => "large-button" , "onclick" => "window.location = '{$this->url}/movies/list/status/P';") )  ?><br />

        </div>

</div>
```

LIST.TPL

```
<div class="box">

        <div class="box-header">

            <h1><?=$this->translate( "movies" )?></h1>

        </div>

        <div class="box-content height-movies">

        <?php foreach ( $this->rs as $posters ) : ?>

            <div class="left">

                <a href="<?=$this->url?>/movies/view/id/<?=$posters['id'] ?>" ><?=$this->image( "filmes/GR" . $posters['image']  , array( "style" => "width: 113px; height: 160px;" ) ) ?></a>  

                <?php if( $posters['hour'] ): ?>

                <h3><?=$this->translate( "hour" )?></h3>


                <?php foreach( $posters['hours'] as $key => $hour ): ?>

                    <?=$hour['room']?>

                    <br />

                    <?=$this->date( $hour["start"] , "H:i" )?> <?=$this->translate( "crase" )?>

                    <?=$this->date( $hour["end"] , "H:i" )?>

                    <br /><br /><br />

                <?php endforeach; ?>

                <?php endif; ?>
```

```
                </div>
        <?php endforeach; ?>
        <div class="clear"></div>
        </div>
</div>
```

TRAILERS.TPL

```
<div class="box">
        <div class="box-header">
                <h1><?=$this->translate( "trailers" )?></h1>
        </div>
        <div class="box-content">
        <?=$this->image( "filmes/GR" . $this->movie->image ) ?>
        <?=$this->movie->link ?>
        </div>
</div>
```

VIEW.TPL

```
<div class="box">
        <div class="box-header">
                <h1><?=$this->translate( "breadcumbs" )?></h1>
        </div>
        <div class="box-content">
                <div class="left">
                        <?=$this->image( "filmes/GR" . $this->movie->image ) ?>
                </div>
                <div class="box-content">
                        <p><?=nl2br( utf8_encode( $this->movie->sinopse ) ) ?></p>
                </div>
        </div>
        <div class="clear"></div>
```

```
	<br />
	<div class="box-button">
		<?=$this->formButton( "bt-trailers" , $this->translate( "trailers" ) , array( "class" => "large-button" , "onclick" => "window.location = '{$this->url}/movies/trailers/id/{$this->movie->id}';") ) ?><br />
	</div>
	<div class="clear"></div>
</div>
```

HELPER

DATE.PHP

```
<?php
class Preceptor_View_Helper_Date
{
	function date( $date = null, $format = "d/m/Y" , $returnNow = false )
	{
		if ( $date != null )
		{
			$date = $this->fromString( $date );
			return date( $format , $date );
		}
		else
		{
			if( $returnNow )
				return date( $format , time() );
		}
	}
	function fromString( $date )
	{
		if ( is_integer( $date ) || is_numeric( $date ) ) {
			return intval( $date );
```

```
                } else {
                        return strtotime( $date );
                }
        }
}
```

IMAGE.PHP

```
<?php
class Preceptor_View_Helper_Image
{
        /**
         * @var string
         */
        const URL = URL_IMAGE;
        /**
         * @param string $sPath Caminho para o arquivo.
         * @param string $aAttribute
         * @return Zend_View_Helper_Image
         */
        public function image( $sPath, $aAttribute = array() )
        {
                $sSource = preg_match( "/^(http|ftp|https|ftps)+:\/\//", $sPath ) === 1 ? $sPath :
self::URL . $sPath;
                return "<img src=\"". $sSource . "\" ". $this->getTagAttributes( $aAttribute ) ." />";
        }
        protected function getTagAttributes( $aAttribute )
        {
                $sContent = "";
                foreach( (array) $aAttribute as $sName => $sValue )
                {
                        $sContent .= $sName . "=\"". htmlspecialchars( implode( " ", (array) $sValue )
) . "\"";
```

```
                }
                return $sContent;
        }
}
```

SCRIPT.PHP

```
<?php
class Preceptor_View_Helper_Script
{
        /**
         * @var string
         */
        const URL = URL_JAVASCRIPT;
        /**
         * @param string $sFile Caminho para o arquivo.
         * @param string $sType Tipo do script a ser importado.
         * @return Zend_View_Helper_Script
         */
        public function script( $sFile, $sType = "text/javascript" )
        {
                if ( preg_match( "/^(http|ftp|https|ftps)+:\/\//", $sFile ) != 1 )
                {
                        $sFile = self::URL . $sFile;
                }
                return "<script type=\"" . $sType . "\" src=\"" . $sFile . ".js\"></script>";
        }
}
```

STYLE.PHP

```
<?php
class Preceptor_View_Helper_Style
```

```
{
        /**
         * @var string
         */
        const URL = URL_STYLESHEET;
        /**
         * @param string $file Caminho para o arquivo.
         * @return string
         */
        public function style( $file )
        {
                if ( preg_match( "/^(http|ftp|https|ftps)+:\/\//", $file ) != 1 )
                {
                        $file = self::URL . $file;
                }
                return '<link href="'.$file.'.css" rel="stylesheet" type="text/css" />';
        }
}
```

PUBLIC/CSS

MAIN.CSS

```
html{background: #F0F8FF;}
body{ }
#hd{ background: #2f4b69; }
#bd{ padding: 1%;}
#ft{
        bottom:0;
        color:#FFF;
        padding: 1%;
        left:0;
    background: #2f4b69;
```

```
        line-height: 20px;
        }
a{ color: #000; }
h1{ font-size: 110%; text-indent: 10px; }
p{ left: 10px; }
.medium-button{ width: 100px; height: 30px; }
.large-button{ width: 200px; height: 50px; }
.box{ position: relative; border: 2px solid #12648E }
.box-header{ height: 20px; background-color: #12648E; width: 100%; font-weight: bold; color:#FFF; }
.box-button{ position: absolute; bottom: 5px; right: 25%; }
.box-content { padding: 10px; padding-left: 5px; width: 87%; }
.box-content p{ margin-left: 115px; text-align: justify; }
.clear{ clear: both; }
.left{ float: left; }
.right{ float: right; }
.height-movies{ width: 485px;  }
#content-print{ display: none; }
```

PRINT.CSS

```
#hd{ display: none; }
#ft{ display: none; }
img{ display: none; }
#print{ display: none; }
#content-print{display: inherit;}
p{ font-size: 150%; }
```