UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

RECONSTRUÇÃO DE UMA APLICAÇÃO WEB UTILIZANDO PADRÕES

EMANOELLE CAROLINE ROPELATO

BLUMENAU
2007

2007/1-09

EMANOELLE CAROLINE ROPELATO

RECONSTRUÇÃO DE UMA APLICAÇÃO WEB UTILIZANDO PADRÕES

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Everaldo Artur Grahl, Mestre - Orientador

BLUMENAU
2007

2007/1-09

RECONSTRUÇÃO DE UMA APLICAÇÃO WEB

UTILIZANDO PADRÕES

Por

EMANOELLE CAROLINE ROPELATO

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Everaldo Artur Grahl, Mestre – Orientador, FURB
Membro:	Prof. Jomi Fred Hübner, Doutor – FURB
Membro:	Prof. Oscar Dalfovo, Doutor – FURB

Blumenau, 10 de julho de 2007

Dedico este trabalho a meu pai que sempre estive ao meu lado e mesmo não estando mais presente fisicamente, estará sempre comigo em meu pensamento. A minha mãe, meu noivo e todas as pessoas que direta ou indiretamente me auxiliaram na realização deste.

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

A meu pai, que mesmo estando ausente neste momento, sempre ofereceu apoio quando mais precisei.

À minha mãe, pela força que ofereceu, mesmo passando por um período ruim de nossas vidas.

À minha família, que mesmo longe, sempre esteve presente.

Ao meu noivo, que passou por este momento junto comigo.

Ao meu orientador, Everaldo Artur Grahl, por ter acreditado na conclusão deste trabalho.

Os bons livros fazem “sacar” para fora o que a pessoa tem de melhor dentro dela.

Lina Sotis Francesco Moratti

RESUMO

Este trabalho tem como objetivo demonstrar os benefícios com a utilização de padrões Web na reconstrução de uma aplicação Web de *HelpDesk*. Para isto, será abandonado o código fonte atual e refeito um novo, aplicando técnicas de desenvolvimento Web. O resultado final apresenta uma análise entre as duas aplicações.

Palavras-chave: *Design patterns*. *WebPatterns*.

ABSTRACT

This work has as objective to demonstrate the benefits with the use of Web standards in the reconstruction of an application Web *HelpDesk*. For this, the current code source and remade a new will be abandoned, applying techniques of Web development. The final result presents an analysis between the two applications.

Key-words: Design patterns. WebPatterns.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

PHP – *Hypertext Preprocessor*

SQL - *Structured Query Language*

ASP - *Active Server Pages*

HTML - *HyperText Markup Language*

Web - *World Wide Web*

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A *World Wide Web* (Web)[1] surgiu em 1990, após a criação da linguagem *HyperText Markup Language* (HTML), com o intuito de ligar computadores de universidades, para uso acadêmico, mas foi em 1993 que a internet se popularizou, com uso de recursos multimídia. Em 1995 tornou-se comercial no Brasil e desde então é considerada uma das ferramentas principais de trabalho. Com isso, ficou claro que a Web não seria apenas mais um lugar para se "fazer as mesmas velhas coisas de modo um pouquinho diferente", segundo Nóbrega (1999). À medida que eram solicitadas novas aplicações, o HTML passou a ser um coadjuvante e deu lugar às linguagens dinâmicas como Perl, Javascript, *Active Server Pages* (ASP), *Hypertext Preprocessor* (PHP), entre outras. Iniciaram-se as conexões com banco de dados e utilização da *Structured Query Language* (SQL). Assim, conforme Conallen (2003, p. 10), “as aplicações Web evoluíram de *sites* para sistemas Web”.

Atualmente, aplicações Web estão mais presentes e seu desenvolvimento representa boa parte da produção das desenvolvedoras de softwares, empresas de publicidade e agências de *design*, o que acaba proporcionando um ambiente amplo de trabalho e cada empresa adota uma metodologia conivente com seu ambiente de trabalho.

Existem duas abordagens básicas no desenvolvimento Web, o ideal artístico e o ideal de engenharia para resolver os problemas do cliente (NIELSEN, 2000, p. 95). A maioria dos projetos é construída a partir da primeira abordagem. O *leiaute* é aprovado pelo cliente e a aplicação evolui à medida que o HTML é gerado e à medida que a *Web Application* (*WebApp*)[2] é implementada. Muitos ainda utilizam esta abordagem por julgarem as *WebApp* um projeto limitado, imediato, volátil e sem necessidade de uma formalização. Enquanto um projeto é pequeno, pode ser conduzido desta maneira.

Hoje é comum deparar-se com grandes aplicações Web, que englobam centenas de objetos, funções, linguagens, classes, dados, textos, integração com outros sistemas e mudanças de versões. Um dos problemas é que as aplicações evoluíram, mas as metodologias aplicadas pelas empresas não. Ainda é utilizada uma abordagem *desktop* em aplicação Web e às vezes nenhuma metodologia, fazendo com que o desenvolvedor depare-se com um projeto demorado, com custos elevados, códigos inflexíveis, manutenções complicadas e um baixo

[1] É o ambiente multimídia internet, também conhecido como WWW.

[2] É o nome dado para as aplicações Web.

reaproveitamento das fontes.

Para o desenvolvimento deste trabalho foi selecionada uma ferramenta de HelpDesk, já existente e a partir disto foi proposta uma reformulação do sistema utilizando padrões e seguindo o ideal de engenharia.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é aplicar padrões na reconstrução de uma aplicação Web existente, comparando as vantagens e desvantagens que um sistema Web padronizado pode apresentar em relação a outro sem padronização.

Os objetivos específicos do trabalho são:

a) aplicar os *WebPatterns* na reconstrução, focando o padrão funcional;
b) apresentar os benefícios da utilização de *Patterns*;
c) documentar os *Patterns* utilizados para fins didáticos;
d) avaliar através de heurística de interface as aplicações antiga e nova.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido em quatro capítulos, conforme seguem as descrições.

No segundo capítulo é apresentada a fundamentação teórica do trabalho, através dos conceitos que envolvem o tema proposto, dando ênfase aos conceitos de padrões de desenvolvimento e *WebPatterns*.

No terceiro capítulo é apresentada à especificação e a implementação da ferramenta, através de seus requisitos, diagramas, tecnologias utilizadas e padrões de desenvolvimento.

Por fim, o quarto capítulo apresenta as conclusões obtidas com a realização do trabalho, análise e comparação entre as duas estruturas do software apresentado.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo deste capítulo é apresentar a fundamentação teórica sobre padrões de desenvolvimento de softwares Web e trabalhos correlatos.

2.1 APLICAÇÕES WEB

Com a popularização da internet e dos computadores domésticos na década de 90, impulsionou-se o desenvolvimento de aplicações que utilizam multimídia, utilizadas para proporcionar repasse de informações de maneira rápida e segura. Desta maneira, as aplicações Web logo evoluíram de simples *sites* para grandes sistemas. Os primeiros *sites*, desenvolvidos por Tim Berner-Lee formavam um sistema hipermídia que permitia apenas para pesquisadores o acesso a documentos e informações publicadas por outros pesquisadores, através de navegadores. Com o passar do tempo, começou-se a criar as *WebApps* que são desenvolvidas a partir de um sistema hipermídia, mas utilizando-se de recursos que tornem sua estrutura dinâmica e utilizando regras de negócio. Conallen (2003, p. 25) afirma que, "certamente, se não houver nenhuma regra de negócio no servidor, o sistema não deverá ser visto como uma aplicação Web".

WebApps são diferentes dos sistemas de software tradicionais, pois envolvem uma mistura de desenvolvimento de sistema e publicação, de *marketing* e computação, de arte e tecnologia. Está em constante manutenção evolutiva, o que torna importante a necessidade de uma arquitetura que proporcione o crescimento de maneira consistente e controlada.

> Projeto *WebApps* engloba atividades técnicas e não técnicas. A aparência do conteúdo é desenvolvida como parte do projeto gráfico, o leiaute de estética da interface com o usuário é criado como parte do projeto de interface, e a estrutura técnica da *WebApp* é modelada como parte do projeto arquitetural e navegacional. (PRESSMAN, 2006, p. 426).

2.2 ENGENHARIA DA WEB

Durante a década de 90 viu-se o grande crescimento da internet e com isso foram construídas muitas aplicações e *Web Sites* com a alegação de que não era mais preciso seguir

as antigas regras de desenvolvimento. Hoje se pode ver que isto não era verdadeiro e que o desenvolvimento de *WebApps* exige sim um planejamento para garantir seu sucesso.

2.2.1 Modelagem de análise para aplicação Web

A primeira impressão que se pode ter é uma contradição em utilizar modelagem de análise no contexto de engenharia de Web, uma vez que as *WebApps* são imediatas e voláteis. Com o passar dos anos foi-se verificando que mesmo com esta contradição é possível trabalhar com modelagens e engenharia de software, tornando o processo de desenvolvimento muito mais suave e garantir que o sistema seja mais manutenível no futuro.

De acordo com Pressman (2006, p. 409), a análise de uma aplicação Web focaliza três questões importantes:

a) que informação/conteúdo deve ser exibido;
b) que funções devem ser realizadas pelo usuário final;
c) que comportamentos devem ser exibidos pela *WebApp* quando a mesma apresenta conteúdos.

Esta área da engenharia Web é bastante preocupada em estabelecer um entendimento básico da *WebApp*, tipos de usuários e quais problemas serão resolvidos para o usuário, focando assim na análise de requisitos para Web que engloba três tarefas principais:

a) formulação: identifica metas e objetivos e tipos de usuários;
b) coleta de requisitos: identifica requisitos e casos de uso sob o ponto de vista do usuário;
c) modelagem de análise: desenvolve os diagramas listados a partir do segundo passo.

2.2.1.1 Análise para *WebApps*

A análise de uma *WebApp* é feita a partir das informações contidas nos casos de uso. A partir deste estudo é possível identificar as classes de análise existentes e seus atributos, o conteúdo a ser apresentado e as funções a ser realizada pela *WebApp*.

Para a identificação destas informações, Pressman (2006, p. 414) apresenta quatro tipos de análises:

a) de conteúdo: identifica os elementos estruturais que englobam objetos de conteúdo

(imagens, textos, animações, etc) que farão parte da *WebApp*. Inclui também todas as classes de análise, descrevendo seus atributos e operações;

b) de interação: descreve o modo de iteração entre o usuário e a aplicação. Para ter sucesso neste processo é muito importante desenvolver os diagramas de seqüência e estado, representando uma "conversa" entre usuário, conteúdo e comportamento da aplicação;
c) funcional: descreve as funcionalidades da *WebApp* em dois níveis de abstração: funcionalidade observável, que engloba qualquer função de processamento executada pelo usuário; operações contidas nas classes os quais manipulam atributos de classes e são executados pela comunicação entre as mesmas. Para demonstrar os detalhes de processamento o diagrama de atividades é o mais adequado;
d) de configuração: apresenta uma lista de atributos do servidor e do lado cliente que a *WebApp* precisa atender.

2.2.2 Análise de Relacionamento – Navegação

As análises anteriores tratavam de elementos de conteúdo e funcionais, conforme evolui o projeto estes elementos tornam-se parte da arquitetura da *WebApp*. Conforme Pressman (2006, p. 421), "cada um destes elementos tem o potencial de ser ligados a todos os outros elementos, podendo aumentar a complexidade navegacional na aplicação".

A proposta deste tema é estabelecer ligações adequadas entre os objetos e as funções do sistema fornecendo as habilidades requeridas pelo usuário (PRESSMAN, 2006). Com isto apresentar todos os relacionamento considerados úteis apresentando uma lista de melhoras para a implementação incluindo *links*, informações e navegação opcional e retirando o que venha apenas a poluir a *WebApp*.

Pressman (2006, p.421) apresenta os quatro passos para esta análise:

a) análise de interessados: identifica as categorias de usuários e sua hierarquia;
b) análise de elementos: apresenta os objetos de conteúdo e funcionais que serão de interesse para o usuário final;
c) análise de relacionamento: descreve os relacionamentos existentes entre os elementos;
d) análise de navegação: examina o tipo de acesso que cada usuário pode ter para

cada elemento disponível.

2.2.3 Heurísticas

A Avaliação Heurística é um método baseado na verificação de uma pequena lista de regras (heurísticas) ou na própria experiência dos avaliadores que visam, de forma econômica, fácil e rápida, descobrir grandes problemas potenciais da interface (NIELSEN, 1994). Segundo, Maciel et al. (2004), este método de avaliação foi concebido a partir de pesquisas dentro do contexto Windows, quando a Microsoft liberou o Windows 3.0, ou seja, no ambiente desktop.

O crescimento das aplicações Web e de suas estruturas apresentaram uma necessidade de criação de diretrizes de avaliação de qualidade especificas, com o intuito de melhorar a usabilidade dos sistemas. Especialistas em usabilidade, como Jakob Nielsen (1994), mostram que usabilidade assumiu um maior destaque que no passado, e descrevem novas regras de usabilidade para a Web.

Heurísticas adaptadas à Web

Existem vários meios de heurística para Web, mas as utilizadas no desenvolvimento deste trabalho foram às heurísticas propostas por Maciel et al (2004), já adaptadas ao ambiente Web e apresentadas a seguir:

a) status do sistema: refere-se aos meios disponíveis para informar, orientar e conduzir o usuário durante a interação com o sistema;
b) compatibilidade do sistema com o mundo real: existência de metáforas que facilitem a compreensão do conteúdo do sistema proporcionando uma melhora da interface entre o homem e o sistema;
c) controle do usuário e liberdade: relaciona-se ao controle que o usuário sempre deve ter sobre o processamento de suas ações pelo sistema;
d) consistência e padrões: consistência refere-se à homogeneidade e coerência na escolha de opções durante o projeto da interface do sistema. Contextos ou situações similares devem ter tratamento e/ou apresentação similares;
e) prevenção de erros: Todos os mecanismos que permitem evitar ou reduzir a

ocorrência de erros, assim como corrigir os erros que porventura ocorram;

f) reconhecimento ao invés de lembrança: colocar os objetos, ações e opções visíveis. Facilitar ao utilizador instruções e ajudas evitando a memorização por parte do usuário;
g) flexibilidade e eficiência no uso: capacidade da aplicação em se adaptar ao contexto e às necessidades e preferências do usuário, tornando seu uso mais eficiente;
h) estética e design minimalista: características que possam dificultar ou facilitar a leitura e a compreensão do conteúdo disponível na ferramenta. Dentre essas características, destacam-se a legibilidade, a estética e a densidade informacional;
i) ajudar os usuários a reconhecer, diagnosticar e corrigir erros: os erros deveriam ser expressados numa linguagem simples, sem códigos estranhos, indicando qual o problema o proporcionando sugestões para o ultrapassar;
j) ajuda e documentação: qualquer informação deverá ser fácil de encontrar, centrado na tarefa do utilizador, proporcionando uma listagem de passos a seguir sem ser muito comprida.

2.2.4 Diagramas para aplicação Web

Este tópico irá mostrar alguns diagramas específicos para desenvolvimento Web. A partir destes é possível montar uma *WebApp* desde o layout até as regras existentes.

2.2.4.1 *Wireframe*

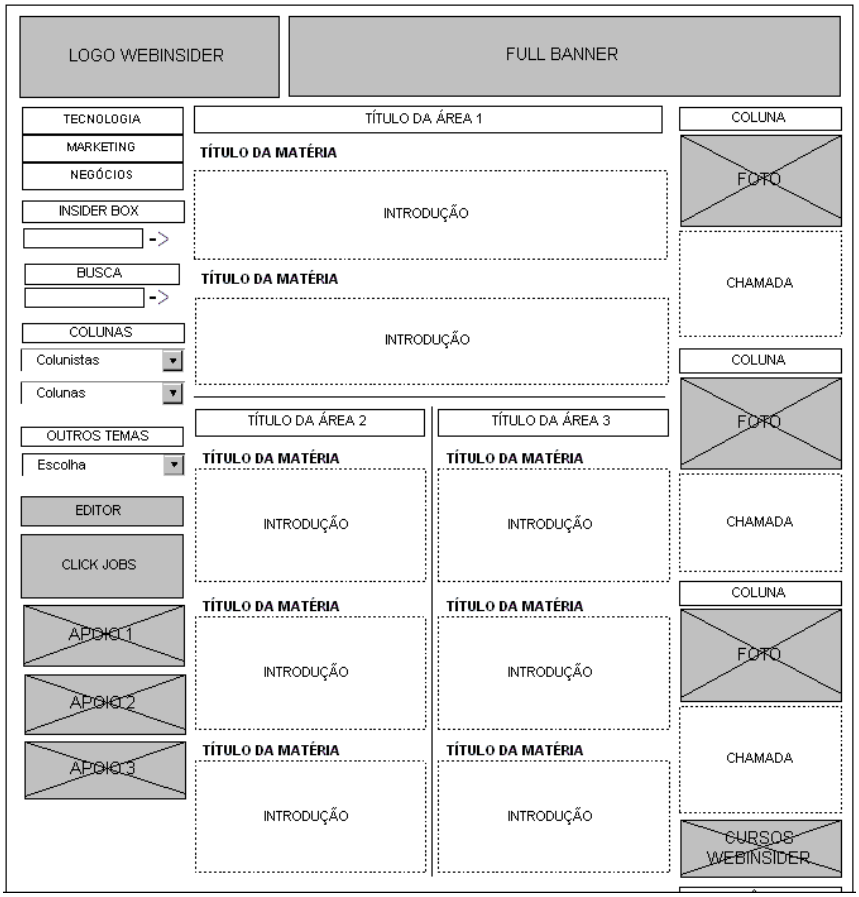
Segundo Oliveira (2003), "o *wireframe* é um documento que se torna cada vez mais fundamental para o trabalho do arquiteto de informação". Sua função, durante o projeto de interface, é estruturar o conteúdo de cada página, indicando o peso e relevância de cada elemento do leiaute e sua relação com os demais elementos formadores do todo, concentrando somente nas questões da interface - como acessibilidade e usabilidade - deixando questões estéticas para o desenvolvimento do leiaute.

Na construção do *wireframe*, o arquiteto busca representar esquematicamente todos os elementos que compõem a página. Imagens, textos, formulários, *flash* e mecanismos de busca

são representados por variações gráficas de elementos similares - como quadrados e círculos, traços contínuos e pontilhados, palavras em negrito e sublinhadas, largura e altura das páginas - padronizadas para todos os *wireframes* da documentação (OLIVEIRA, 2003).

Durante sua construção ainda pode contribuir com a usabilidade de um site, evitando conteúdos redundantes e sobreposição de conteúdo, além de leiautes complexos ou *links* escondidos. Além de poder prever em sua documentação alguns detalhes como maior rapidez de obtenção de resultados, supressão de etapas intermediárias, controle sobre o modo de exibição do conteúdo, menus e *links* sempre visíveis e padronizados, tornando a aplicação Web mais eficaz.

A figura 1 apresenta um modelo de *wireframe*, onde se pode ver as áreas destinadas as imagens no formato quadrado e de cor cinza, os menus representados por retângulos com fundo branco, áreas de textos mostradas a partir das linhas pontilhadas. Com esta apresentação o usuário já pode identificar a disposição de conteúdos de seu projeto.



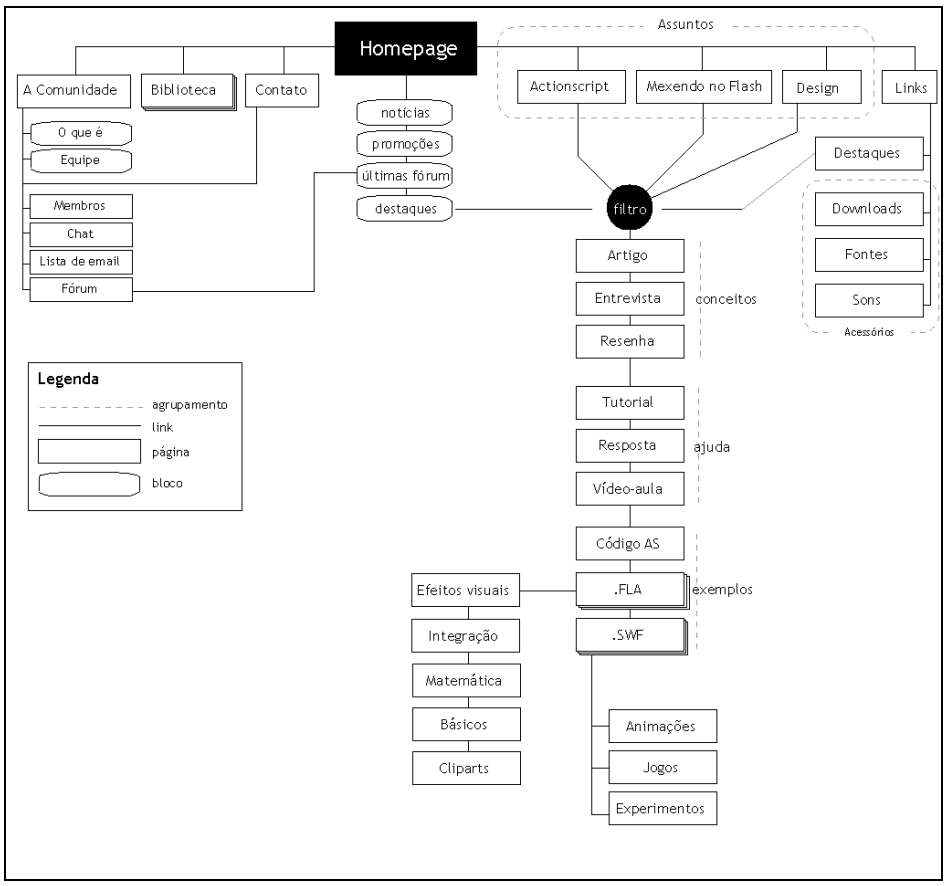
Fonte: Oliveira (2003).

Figura 1 - Modelo de *wireframe*

2.2.4.2 Diagrama de Navegação

Este diagrama semelhante a um diagrama de transições estado, mostra quais são as possíveis navegações entre as telas de um sistema será utilizado no projeto de navegação.

Neste modelo não se faz distinção entre a funcionalidade, apenas na navegação. Não apresenta a necessidade de selecionar elementos para fazer uma determinada função, mas sim que se pode navegar de uma tela a outra, considerado o comportamento normal. O importante aqui, conforme Xexéo (2007, p. 262) é ter uma idéia de quantas "telas abstratas" serão necessárias e ter uma noção do comportamento do sistema, conforme figura 2.



Fonte: Amstel (2007).

Figura 2 - Modelo de diagrama de navegação

> Entre as vantagens de construir um diagrama de navegação estão a sua simplicidade e informalidade. Apesar de abstratos, podem ser usados em discussões com o usuário com certa facilidade. Além disso, servem também para dar aos desenvolvedores uma visão geral do comportamento do sistema. (XEXÉO, 2007, p. 263).

2.2.4.3 Diagrama de Arquitetura de Conteúdo

Utilizado durante a fase de desenvolvimento da arquitetura da *WebApp* este diagrama focaliza a definição da estrutura global de hipermídia da aplicação.

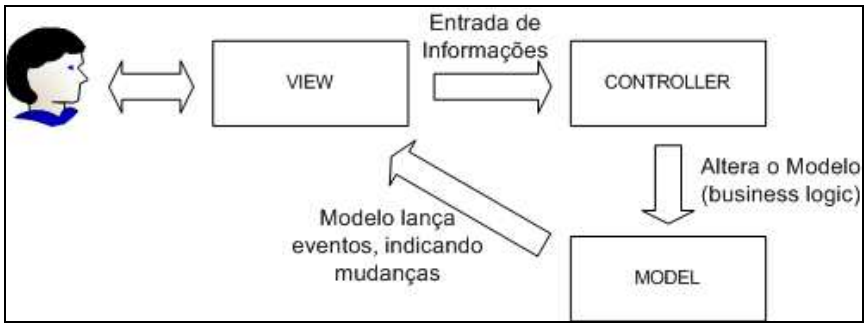
Existem quatro diferentes estruturas, conforme Pressman (2006, p. 441):

a) lineares: todas as páginas acompanham em seqüência com vínculos que levam de uma página a outra, normalmente com opções de avançar e retroceder, oferece também um caminho de desvio do caminho principal;
b) malha: aplicada quando o conteúdo pode ser organizado em categorias de duas ou mais dimensões, verticalmente e horizontalmente;
c) hierárquicas: a *home page* fornece uma visão geral do conteúdo que está subordinado a ela e ainda define os principais vínculos ás páginas dos níveis inferiores da hierarquia;
d) rede: são estabelecidos vários vínculos entre diferentes pontos de níveis equivalentes ou não e caminhos que farão com que o visitante retorne e passe várias vezes no mesmo ponto.

2.2.4.4 Diagrama de Arquitetura de *WebApp*

Aplicado durante a fase de desenvolvimento da arquitetura da *WebApp* este diagrama sugere dividir a aplicação em três camadas, mantendo a interface, aplicação e navegação separadas, permitindo que uma mesma lógica de negócios possa ser acessada e visualizada através de várias interfaces.

O diagrama sugerido por Pressman (2006, p.443) é a arquitetura *Model-View-Controler* (MVC), que desacoplam a interface com o usuário da funcionalidade e conteúdo informacional da *WebApp*, apresentada na figura 3.



Fonte: Sauvé (2006).

Figura 3 - Modelo de MVC

2.3 PADRÕES DE PROJETO

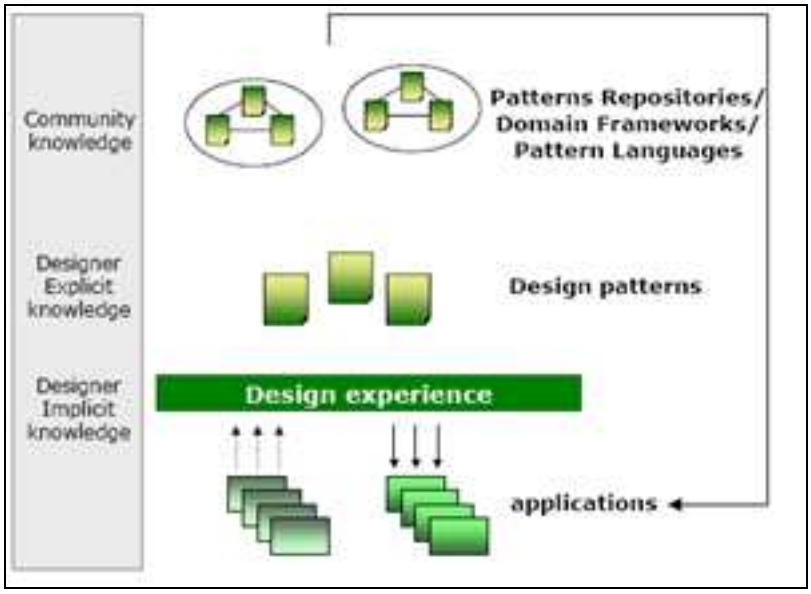
No início dos anos 70, um grupo de acadêmicos de Arquitetura, liderado por Christopher Alexander criou o *design patterns* (padrão de projetos), focado em engenharia civil e arquitetura, ao observar que a maioria dos projetos na área poderia ser descrita com uma reunião de um conjunto de soluções (ALLSOPP, 2005). Em 1987, um grupo de projetistas e analistas descobriu que para problemas comuns no dia-dia em seus projetos, eram aplicadas soluções parecidas. Assim, decidiram explorar algumas das idéias de Alexander e definiram um pequeno catálogo de *design patterns* para desenvolvimento de sistemas (BOLCHINI, 2000, p. 12).

Na definição de Christopher Alexander, “um *design pattern* apresenta um problema, que ocorre repetidamente em um ambiente, e descreve então o núcleo da solução a esse problema de tal maneira que se pode usar esta solução milhares de vezes” (BOLCHINI, 2000, p. 12). Em uma visão geral, pode ser definido como um formulário de documentação problema-solução em uma linguagem de programação. Cada *design pattern* recebe um nome que remete sua essência e aplicação, trazendo como conteúdo, uma solução comprovada de um problema em um determinado contexto. O problema e a solução devem estar descritos e documentados em cada *pattern*.

Um *design pattern* apresenta uma solução a um problema encontrado durante o projeto. Usando um *design pattern* o projeto tende a ter um formato muito simples e compreensível e também ser utilizado diversas vezes para diferentes aplicações (BOLCHINI, 2000, p. 9). Isto porque esses padrões visam capturar e catalogar os aspectos essenciais, universais dos problemas encontrados nos projetos e suas respectivas soluções.

A figura 4 mostra o papel dos padrões no processo de reuso de experiência. Os

desenvolvedores adquirem o conhecimento pessoal ao desenvolver aplicações; reutilizam e aplicam estas habilidades em novas aplicações. Os novos *patterns* são enviados a uma comunidade e são repassados para milhares de desenvolvedores, iniciando novamente o ciclo.



Fonte: Bolchini (2000).

Figura 4 – Paradigma de reuso

WebPatterns

Recentemente, uma comunidade de analistas verificou que é possível reaproveitar o conhecimento se forem criados *patterns* específicos para a Web, surgindo assim os *WebPatterns*. *WebPatterns* trazem estratégias eficazes a fim atender a abertura entre exigências e o projeto, fornecendo à essência de uma solução provada a um problema retornando do projeto Web dentro de um determinado contexto. Um teste padrão nunca fica dentro de uma única área ou em um único projeto, ele se envolvem com as áreas aonde foram aplicados, criando interconexões com outros interesses do projeto (BOLCHINI, 2000, p. 32).

Portanto a idéia geral dos *WebPatterns* é que determinado tipo de estrutura, em uma aplicação Web, que possua elementos específicos, com classe particular e valores da identificação possa ter seu núcleo (elementos, classes e valores) reaproveitado em outras aplicações. Os *WebPatterns* podem ajudar o trabalho de desenvolvedores experientes, explorar soluções novas do projeto e dar suporte a novos desenvolvedores. No geral os testes padrões realçam a qualidade do projeto, o custo e a execução do projeto (BOLCHINI, 2000, p. 32).

Os *WebPatterns* podem ser classificados em quatro tipos: estrutura, navegação, interface e funcional. A estrutura foca a estrutura da informação; a navegação visa a arquitetura de navegação; a interface foca os padrões de leiaute e iteração e o funcional verifica o comportamento iterativo pelo lado cliente e as funções do sistema que são comuns a todos os usuários (GONÇALVES et al., 2005, p. 7). Cada *WebPattern* possui características e padrões de projeto apropriados e deve ser usado em domínios de aplicação conforme o Quadro 1.

TIPO DE PADRÃO	PADRÃO DE PROJETO	CARACTERÍSTICA	EXEMPLOS
Estrutura	Centro de Coleção	Coleção de elementos de informação independentes.	Catálogos de produtos (Amazon, Submarino, Ponto Frio)
	Entidade Complexa	Entidades formadas por componentes e sub-componentes, em um mesmo contexto semântico.	*Site* de configuração de produto (Dell computadores)
Navegação	*Tour* Guiado	Fluxo seqüencial preestabelecido de navegação. Aquisição do conteúdo por partes.	*e-Learning*, Pedido de compra (www.voegol.com.br)
	Índice de Navegação	Apresenta os elementos de conteúdo em fluxo não seqüencial, a partir de um nó central (Centro de Coleção)	Portais de conteúdo (Terra, UOL, Globo)
Interface	*Layout*	Foca elementos de apresentação de conteúdo e estética	*Web Sites* institucionais
	Interação	Trata da interação humano–computador e ergonomia cognitiva. Foca elementos de interface e interatividade com o usuário.	*e-Learning*, Aplicações lúdicas
Funcional	Funcional – foco no usuário	Visa comportamentos interativos e individualizados	Interfaces de carrinho de compras, salas de bate-papo, *e-Learning*, Aplicações lúdicas
	Funcional – foco no sistema	Visa comportamentos gerais da aplicação (efetivos para todos os usuários)	*e-Business* em geral, *intranets* corporativas, *Web Sites* de trocas e leilões

Fonte: Gonçalves et al. (2005, p. 9).

Quadro 1 – Padrões de projeto para Web e elementos relacionados no domínio de modelagem

Os *WebPatterns* escolhidos para esta aplicação são voltados para o tipo funcional, visando captar todos os comportamentos dos usuários na iteração com as telas para acrescentar ao sistema uma ação mais precisa entre cliente x sistema. O lado do sistema também foi trabalhado, sendo analisadas cada iteração entre as classes e os resultados apresentados neste comportamento.

A seleção dos padrões foi baseada, portanto, nos repositório de *WebPatterns* executados especificamente para PHP e com foco funcional. Os padrões foram extraídos a partir do grupo WebPatterns.org (WEBPATTERNS, 2007).

Pattern	Definição	Exemplo
Singleton	Aplica-se em situações em que é preciso haver uma só instância de uma classe. Implementar esse padrão permite ao programador fazer essa instância única ser facilmente acessível por muitos outros objetos.	O exemplo mais comum é uma conexão com um banco de dados.
Factory	Permite a instanciação de objetos em tempo de execução. É chamado de *Factory* uma vez que é responsável por "produzir" um objeto.	Mudar subclasses mantendo a classe original sem alterações não prejudicando os outros elementos que a utilizam.
FastTemplate	Separa os códigos de linguagem (PHP, ASP, SQL) do HTML organizando o script e oferece possibilidade de reaproveitar modelos HTML previamente criados.	A divisão do PHP duas partes: o código de programação e o código de formatação. Em outras palavras, comandos PHP em um arquivo, *tags* HTML em outro.
Builder	Duas classes trabalham junto para construir um objeto. Uma controla e especifica que peças e variações entrarão em um objeto. Outra monta a especificação dada ao objeto.	A forma de apresentar os dados de um bloco de repetição. A classe construtora recebe todos os dados, monta as linhas (leiaute) que irão aparecer e quando concluído, a classe diretora manda as informações do objeto para que a construtora retorne para o código HTML.
Decorator	Representa o processo de criar uma hierarquia nova da classe que adicione o comportamento novo ou modificado a uma classe existente sem modificar a classe existente de seu estado original.	Criar janelas com rolagem vertical, outras com rolagem horizontal, e com os dois tipos de rolagem utilizando apenas uma classe base, sem decoração e responsável por todos os itens em comum das janelas e uma classe de decoração para cada tipo de janela, nas quais são definidas apenas suas características.
PEAR *Cache Lite*	Salvam o resultado da execução de um *script* em *cache* evitando a re-execução da página PHP original. Fornece classes para salvar em qualquer informação de um script PHP (trechos de código até scripts inteiros);	Pode-se gravar o resultado de uma consulta ao banco de dados. Em um sistema que sempre traz a informação do nome do usuário, para que essa informação não seja sempre carregada a mesma é gravada no *cache*.

Quadro 2 – *Web Patterns* utilizados na aplicação

Web Design Patterns

A fim de desenvolver padrões para *Web Design* para melhorar a qualidade final da aplicação da Web, Bolchini (2000, p. 8) fez testes nos resultados obtidos por outros desenvolvedores Web. A partir disto surgiu os Web *Design Patterns*. Atualmente o interesse, segundo Bolchini (2000, p. 8), está focalizado nos projetos, onde o hipermídia e a pesquisa da Web podem fornecer resultados favoráveis. O reuso de padrões no projeto de *design* é cada vez mais estratégico para reduzir o custo e para melhorar a qualidade do projeto.

Os Web *Design Patterns* são definidos por um molde da descrição similar àquele de *patterns* tradicionais do software, podendo ser classificados por áreas do projeto, por dimensão da aplicação ou por domínio da aplicação.

A grande estratégia em usar estes padrões é liberar o tempo dos *designers* experientes para pesquisar novos projetos e para os inexperientes, começar a trabalhar na forma correta do conhecimento (BOLCHINI, 2000, p. 8). Além de realçar a qualidade e o custo do projeto e da execução. Com respeito à estrutura particular de projetos Web, este padrão deve fornecer estratégias eficazes a fim suprir as necessidades entre exigências e a projetar.

Um Web *Design Pattern* muito conhecido é o *Breadcrumbs*, usado para apresentar a página em que o usuário está e toda a hierarquia de navegação que o mesmo veio a ser conduzido. Sempre aplicado antes do título da seção, ajuda o usuário a voltar a conteúdos anteriores que sejam ligados com o atual a partir de *links*, apresentado na figura 5.

Fonte: Web Patterns (2006).

Figura 5 – Exemplo de utilização do *Breadcrumbs*

Os padrões escolhidos para o desenvolvimento do trabalho são os que se adequaram mais no projeto e que a sua utilização é realmente importante para proporcionar um ambiente mais agradável de navegação. A seguir tem-se a lista de padrões, sintetizados nos quadros 3 e 4, retirados do grupo de pesquisa *Patterns for php* (PATTERNS FOR PHP, 2007).

Pattern	Problema	Solução
FAQ	Quando os usuários têm perguntas a respeito de um site ou dos tópicos relacionado ao mesmo.	O FAQ é uma página que começa com as perguntas, numeradas e categorizadas seguidas pelas respostas. Devem-se utilizar âncoras para facilitar a navegação, quando a página for longa. Se o FAQ tratar do site em geral fazer o FAQ acessível através do menu. Se o FAQ tratar de uma subseção particular de um local, colocar a ligação perto de onde pertence.
Menu *Fly-out*	Necessidade dos usuários em encontrar os itens do site diretamente no menu.	Mostrar o submenu quando passar o mouse sobre o menu visível. O menu move-se para cima, para baixo ou para a direita (nunca esquerdo) de tal maneira que não cubra os outros elementos de menu. Pode existir horizontalmente ou verticalmente.
Impressão de Tela	Necessidade de o usuário imprimir a página que estão vendo. Imprimir *Web pages* é frequentemente problemático porque as páginas não são projetadas ser impressas.	Desenvolver uma versão impressão-amigável e disponibilizar um *link* (geralmente uma imagem de impressora) ao lado do índice da página. Uma versão impressão-amigável é basicamente uma página apenas com textos sem outros elementos.
Caixa do atalho (*shortcut box*)	Os usuários querem alcançar uma funcionalidade específica de maneira direta.	Colocar *combobox* preenchidos com itens importantes que o usuário necessite pesquisar/encontrar rapidamente. Adicioná-lo em uma posição fixa da página. Quando um item for selecionado, os usuários são remetidos para a funcionalidade escolhida.
Área de Pesquisa (*searcharea*)	A necessidade dos usuários encontrar uma determinada informação no *site*.	Usar uma área dedicada com tipos diferentes da funcionalidade da busca. Agrupar os tipos diferentes da funcionalidade da busca e colocá-los em uma área retangular pequena. A área é colocada em uma posição visível na página mas não prejudicando a navegação principal. Ao lado colocar um *link* para uma busca avançada.

Fonte: Web Patterns (2006).

Quadro 3 – *Web Design Patterns* utilizados na aplicação

Pattern	Problema	Solução
Breadcrumbs	Os usuários precisam saber onde estão em uma estrutura hierárquica da aplicação.	Mostrar o trajeto do nível superior à página atual. O trajeto mostra a posição da página atual na estrutura total da informação. Cada nível da hierarquia é etiquetado e funciona como uma ligação a esse nível. A página atual fica no final dando a noção de onde o usuário esta navegando. Deve-se usar os separadores como > ou \ que sugerem um movimento descendente. Se o trajeto se tornar longo, algumas das etapas podem ser substituídas por um por "…". É colocado perto da área de conteúdo, acima do título da página.
Home Page	Os usuários necessitam compreender se estão no local correto e como podem se mover para realizar tarefas no *site*.	A *homepage* deve balançar a navegação, o índice, e os elementos que constituem o *site*, introduzindo sua finalidade e identidade. Os elementos exatos de uma *homepage* dependem altamente do *site*. Será diferente para cada único local para fora lá. A *homepage* é uma página especial. É conseqüentemente normal que tem uma disposição ligeiramente diferente do que as outras páginas do local.
Mensagem de erro em formulários	Os usuários estão tentando preencher as informações incorretamente.	Informar aos usuários que há um problema e como resolver o problema. Dizer também aos usuários onde o problema ocorreu.
Entrada de dados corretos	O usuário precisa fornecer dados formatados para a aplicação, mas não sabe qual o formato esperado.	Etiquetar cada campo com o nome da unidade de dados que pode ser informado. O campo não pode permitir que os dados incorretos sejam cadastrados. Evitar os campos onde os usuários podem datilografar o texto livre. Adicionalmente, explicar a sintaxe do campo com um exemplo ou uma descrição do formato.
Formulário	Os usuários necessitam fornecer a informação e emiti-la ao sistema.	Oferecer aos usuários um formulário com os elementos necessários. Certificar-se de que os usuários compreendem o que está sendo perguntando. Agrupar elementos e campos que descrevem uma mesma situação. Utilizar os inputs corretos para cada tipo de questionamento.

Fonte: Web Patterns (2006).

Quadro 4 – *Web Design Patterns* utilizados na aplicação - Continuação

2.4 TRABALHOS CORRELATOS

Vários autores abordaram temas de padronização utilizando componentes de métodos concretos e *templates*, reaproveitado os códigos, tanto para Web como para software.

Almeida (2004) apresentou em seu relatório de estágio um estudo com foco na utilização de metodologias baseadas em Padrões de Projeto para o desenvolvimento de uma ferramenta Web. Como estudo de caso, foi desenvolvida uma ferramenta que organiza os documentos referentes à pauta de uma sessão parlamentar, utilizando PHP 5. O foco principal de seu trabalho era a utilização dos padrões de projeto juntamente com o paradigma da Orientação a Objetos para o desenvolvimento da ferramenta e não a ferramenta propriamente dita. O resultado final apresentou os problemas da aplicação, bem como suas respectivas soluções baseadas em padrões de projeto, concluindo o quão importante se é trabalhar com um paradigma consistente e bem definido como a Orientação a Objetos, reforçado pela aplicação das metodologias de Padrões de Projeto, possibilitando, assim, um desenvolvimento mais consistente e menos sujeito à falhas.

Sorroche e Lopes (2003) apresentam um estudo de caso sobre o desenvolvimento e a implementação de um software (sistema de auxílio à matrícula), utilizando *design patterns* e a tecnologia J2EE, destacando a modelagem de um sistema que agrega estas duas tecnologias. A junção das tecnologias de padrões de projeto e J2EE em um mesmo sistema apresentou-se totalmente apropriada, segundo os autores. O uso de padrões de projeto ajudou a reduzir a complexidade e promoveu uma grande reutilização no desenvolvimento sistema em questão. No caso de um software de porte médio, conforme verificado na arquitetura do estudo de caso, estas características ficam ainda mais realçadas, uma vez que na própria especificação das classes é realizada a sua relação com os padrões de projeto.

Grott (2003) apresentou em seu Trabalho de Conclusão de Curso (TCC), um estudo sobre padrões de projeto (*design patterns*) e *frameworks* para o desenvolvimento de um *framework* de cálculo de impostos incidentes em vendas de mercadorias. Os padrões foram aplicados devido à complexidade dos sistemas, prazo menor de entrega e consequentemente, teve uma redução de custo. Os *design patterns* utilizados no decorrer deste TCC descrevem soluções para problemas recorrentes na formação de *frameworks*. Como resultado, Grott (2003) chegou a um conjunto de informações que podem ser utilizados por desenvolvedores interessados em estudar reutilização e aplicação de padrões de projeto e *frameworks* para a melhoria de seu desenvolvimento de software.

3 DESENVOLVIMENTO DO TRABALHO

Com base nos conceitos e materiais estudados durante a realização deste trabalho foi iniciada a fase de elicitação dos requisitos do software. Nas próximas seções serão apresentados os requisitos do software, especificação, detalhes da implementação e por fim os resultados e discussões.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O *HelpDesk* é utilizado para a gestão de ambiente e serviços em alguns departamentos de Tecnologia de Informação (TI), em uma empresa do ramo têxtil. Esta aplicação possui uma versão *WebApp* e uma aplicação Windows integradas. O aplicativo é desenvolvido por uma empresa especializada e não sofrerá nenhuma alteração. A versão Web foi reconstruída a partir dos conceitos do aplicativo e adicionada novas funções relevantes para um determinado departamento.

Com base nisto, definiu-se a reconstrução da ferramenta e a mesma deveria apresentar todas as funcionalidades já existentes na aplicação antiga, atendendo aos requisitos funcionais existentes, que são:

a) permitir ao usuário abrir novos atendimentos;
b) permitir ao atendente abrir novos atendimentos em nome de usuários e de maneira mais detalhada;
c) fazer pesquisa dos atendimentos por identificador, solicitante ou palavra-chave;
d) permitir ao usuário acrescentar comentários aos atendimentos;
e) permitir ao usuário acrescentar grau de solução a um chamado terminado;
f) restringir a abertura de chamados do usuário, caso exista um item terminado e sem solução;
g) permitir ao usuário mudar seus dados pessoais;
h) permitir ao usuário visualizar todos os atendimentos abertos por ele e pelo seu centro de custo;
i) permitir ao atendente manipular os atendimentos direcionados para ele ou para sua equipe;

j) gravar qualquer alteração realizada pelo atendente em um histórico.

Os requisitos não funcionais serão novos, por isto deverá:

a) adotar *WebPatterns* em PHP;

b) ser desenvolvida na linguagem PHP 5 orientada a objetos com banco de dados MySQL.

3.2 ESPECIFICAÇÃO

A especificação do software será apresentada através dos diagramas de casos de uso e diagrama de atividades. Os diagramas foram elaborados na ferramenta EA versão 6.0 versão *trial*. Serão apresentados também alguns dos diagramas Web – *wireframes*, diagrama de navegação, arquitetura de conteúdo e hierarquia de usuários - desenvolvidos nas ferramentas *ConceptDraw Web Wave* e Axure RP Pro, ambas na versão trial.

Não foram apresentados os casos de uso de configuração, onde estariam os cadastros de usuários, identificadores, atendentes e outros. O responsável por estes cadastros é o sistema de *Helpdesk desktop* e não a plataforma Web, que é voltada apenas para o atendimento de chamados.

Na figura 6, são apresentados os pacotes com os cenários correspondentes aos casos de uso especificados.

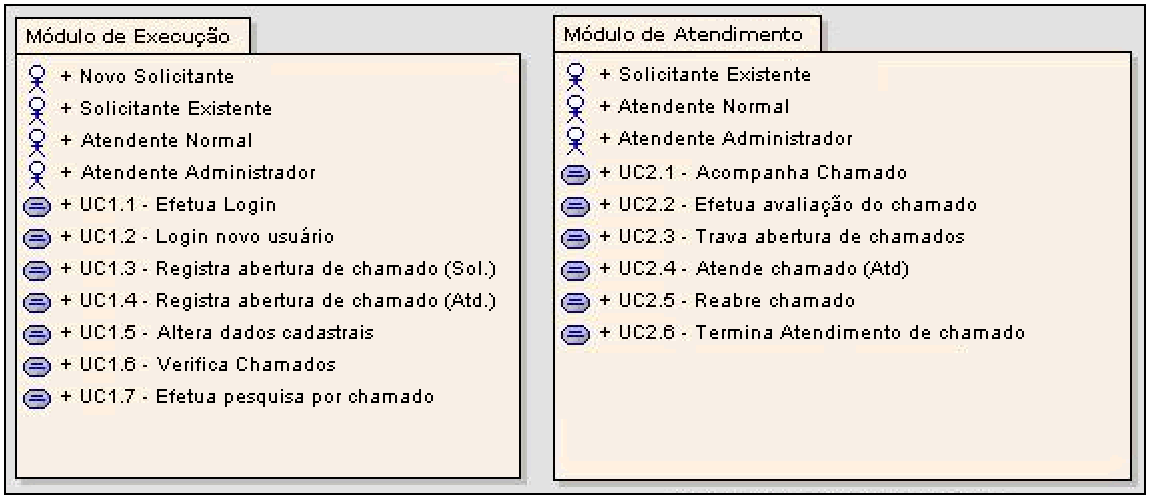


Figura 6 – Pacotes de casos de uso

3.2.1 Hierarquia de Usuários

A partir da análise de requisitos, viu-se que haveria dois tipos de usuário: solicitantes e atendentes. Só que dentro destes dois níveis ocorreriam divisão e uma diferença nos acessos.

a) usuário: corresponde a categoria mais geral do usuário, sendo refinada em níveis abaixo;
b) solicitante: são usuários que abrem os chamados para o departamento de sistema, acompanham os atendimentos e dão o parecer sobre os mesmos. Todos os solicitantes são cadastrados automaticamente por um processo gerado pela base de dados que traz informações do sistema de RH;
c) novo solicitante: estão cadastrados na base de dados, mas nunca utilizaram o sistema. Seu primeiro acesso precisa cadastrar uma senha para continuar a navegação;
d) solicitante existente: já entrou no sistema e possui senha cadastrada;
e) atendente: são usuários que fazem o atendimento de todos os chamados abertos para um determinado grupo. Podem abrir chamados em nome de qualquer solicitante ou ainda trabalhar em modo Solicitante. São cadastrados através do software *HelpDesk* (sistema desktop);
f) atendente normal: faz todo atendimento dos chamados que estão destinados a seu grupo. Possui restrições em alguns campos do Chamado;
g) atendente administrador: faz a mesma coisa que o Normal, mas tem acesso para mudar todas as informações que o sistema permite.

Para que não ocorressem problemas foi criado o diagrama de usuários, conforme apresenta a figura 7.

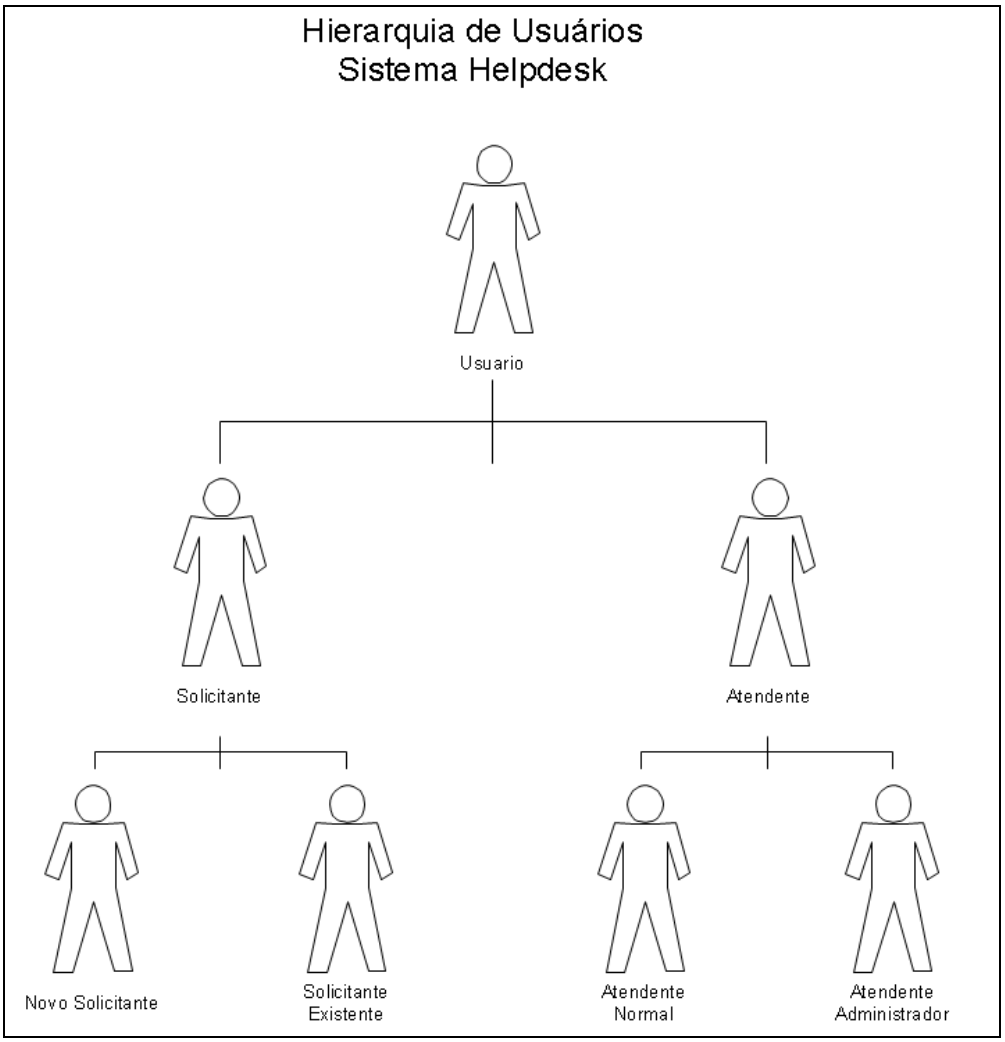


Figura 7 – Hierarquia de usuários do sistema *HelpDesk*

3.2.2 Módulo de execução

Neste módulo estão concentrados os casos de uso relacionados à execução dos chamados. É responsável tanto pela abertura de um chamado, listagem de chamados, pesquisas por informações, alteração de dados cadastrais. Os casos de uso deste módulo estão na figura 8.

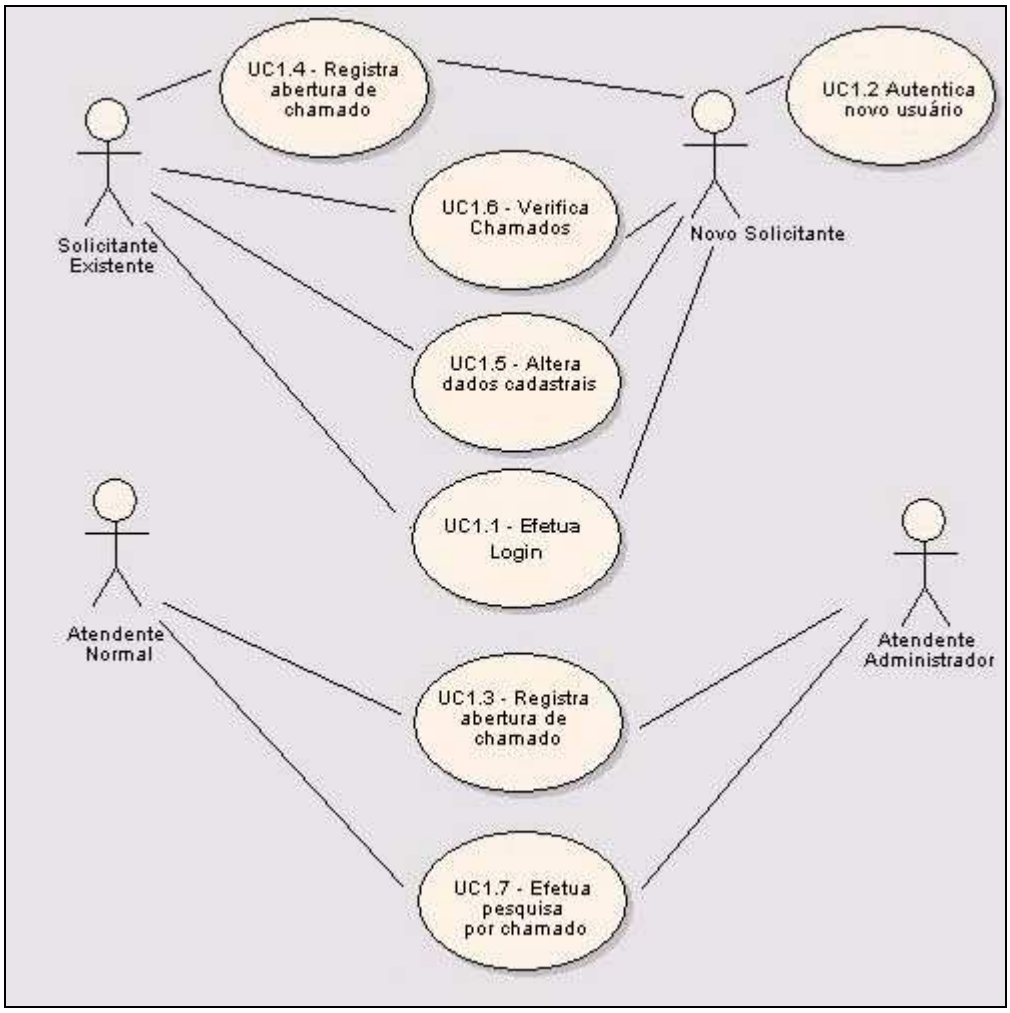


Figura 8 – Casos de uso do módulo de execução

3.2.2.1 UC1.1 – Efetua login

Este caso de uso permite a autenticação do usuário ao sistema. O cenário principal é descrito a seguir:

a) solicitante ou atendente informa *login* e senha;
b) sistema verifica se é atendente e marca opção “área técnica”;
c) sistema verifica a existência do mesmo na base e autentica;
d) o caso de uso é encerrado.

3.2.2.2 UC1.2 – Autentica novo usuário

Este caso de uso permite a autenticação do usuário e a criação de sua senha de acesso. O cenário principal é descrito a seguir:

a) solicitante informa login e clica em entrar;
b) sistema verifica a existência do mesmo na base e autentica;
c) sistema manda usuário para tela de cadastro de senha;
d) usuário informa uma senha;
e) sistema grava senha para usuário;
f) o caso de uso é encerrado.

3.2.2.3 UC1.3 – Registra abertura de chamado

Este caso de uso permite a abertura de chamados por parte do solicitante, é a etapa inicial do ciclo de vida de um chamado. Para isso o solicitante deve estar previamente cadastrado no sistema e já ter cadastrado sua senha de acesso.

O cenário principal para este caso de uso é descrito a seguir:

a) solicitante seleciona a opção “Tipo” que é o tipo de atendimento;
b) sistema exibe a lista de áreas responsáveis para o tipo escolhido;
c) solicitante seleciona a opção “Área Responsável”;
d) solicitante preenche as informações do formulário e clica em confirmar;
e) sistema faz validação dos dados, cadastra o Chamado e encaminha e-mail ao solicitante e aos responsáveis;
f) o caso de uso é encerrado.

No cenário alternativo “Limpar”, caso no passo “d” do cenário principal o usuário clique na opção limpar, o sistema remove todas informações inseridas na tela até o momento e não salva a solicitação.

No cenário alternativo “Validação”, caso no passo “e” do cenário principal o sistema identifique uma informação incorreta, retorna para a tela de abertura de chamados apresentando o erro ocorrido.

3.2.2.4 UC1.4 – Registra abertura de chamado

Este caso de uso permite a abertura de chamados por parte do atendente, é a etapa inicial do ciclo de vida de um chamado. Para isso o atendente deve estar previamente cadastrado no sistema.

O cenário principal para este caso de uso é descrito a seguir:

a) atendente informa um solicitante;
b) atendente seleciona a opção “Tipo” que é o tipo de atendimento;
c) sistema exibe a lista de áreas responsáveis para o tipo escolhido e lista nome dos atendentes pertencentes a área escolhida;
d) atendente seleciona a opção “Área Responsável”;
e) atendente preenche as informações do formulário e clica em confirmar ou atender;
f) sistema faz validação dos dados, cadastra o Chamado e encaminha e-mail ao solicitante e aos responsáveis;
g) o caso de uso é encerrado.

No cenário alternativo “Limpar”, caso no passo “e” do cenário principal o usuário clique na opção limpar, o sistema remove todas informações inseridas na tela até o momento e não salva a solicitação.

No cenário alternativo “Confirmar” caso no passo “e” do cenário principal o chamado é encaminhado para o grupo de atendimento ou para o atendente selecionado;

No cenário alternativo “Atender” caso no passo “e” do cenário principal o chamado é encaminhado para o atendente que está fazendo a Abertura do chamado;

No cenário alternativo “Validação”, caso no passo “e” do cenário principal o sistema identifique uma informação incorreta, retorna para a tela de abertura de chamados apresentando o erro ocorrido.

3.2.2.5 UC1.5 – Altera dados cadastrais

Este caso de uso contempla a atualização de dados para contato e senha de acesso ao sistema. O cenário principal para este caso de uso é descrito a seguir:

a) atendente/solicitante informa o ramal;
b) atendente/solicitante informa senha antiga e nova senha, repetindo a nova senha;

c) sistema permite apenas a troca de ramal e senha;
d) atendente/solicitante clica em alterar;
e) o caso de uso é encerrado.

No cenário alternativo “Limpar”, caso no passo “d” do cenário principal o usuário clique na opção limpar, o sistema remove todas informações inseridas na tela até o momento e não salva a solicitação.

No cenário alternativo “Senha Incorreta”, caso no passo “b” o usuário informar a senha atual incorreta ou informar a nova senha diferente da redigitação, o sistema retorna para a tela de “Meus dados” informando o problema que ocorreu.

3.2.2.6 UC1.6 - Verifica Chamados

No cenário alternativo Este caso de uso permite a consulta dos chamados de acordo com o status do mesmo.

O cenário principal para este caso de uso é descrito a seguir:

a) após conectar ao sistema o atendente/solicitante possui uma tela onde pode verificar cada uma das solicitações existentes, apresentadas conforme a configuração de status;
b) atendente/solicitante pode escolher qual ou quais status deverão aparecer no tela como padrão;
c) atendente/solicitante pode configurar o número de chamados que deseja ver na tela;
d) sistema grava a última escolha de Status e demais configurações em um cookie, para que traga sempre o padrão selecionado pelo usuário;
e) sistema apresenta a lista o chamados conforme os status escolhidos pelo usuário;
f) atendente/solicitante pode selecionar um chamado e verificar em detalhes todas as informações do mesmo, podendo alterar informações conforme sua hierarquia;
g) o caso de uso é encerrado.

3.2.2.7 UC1.7 - Efetua pesquisa por chamado

Este caso de uso possibilita ao atendente pesquisar chamados que estejam direcionados

para o seu grupo de trabalho através de informações passadas ao sistema.

O cenário principal para este caso de uso é descrito a seguir:

a) atendente informa os dados que considera relevante para a pesquisa;
b) atendente clica em “Pesquisar”;
c) o caso de uso é encerrado.

No cenário alternativo “Nova Pesquisa”, caso no passo “c” do cenário principal o selecione a opção Nova Pesquisa, o sistema irá ignorar as informações inseridas na tela até o momento e limpará os campos.

3.2.3 Módulo de atendimento

Neste módulo estão concentrados os casos de uso relacionados ao gerenciamento dos chamados, tanto para os usuários quanto para os atendentes. Este módulo é responsável pelo atendimento de um chamado, reabertura de chamados, inserção de novas tarefas, manutenção de chamados, cadastro de comentários e acompanhamento e avaliação do usuário ao término do chamado. Os casos de uso correspondentes a este módulo são ilustrados pela figura 9.

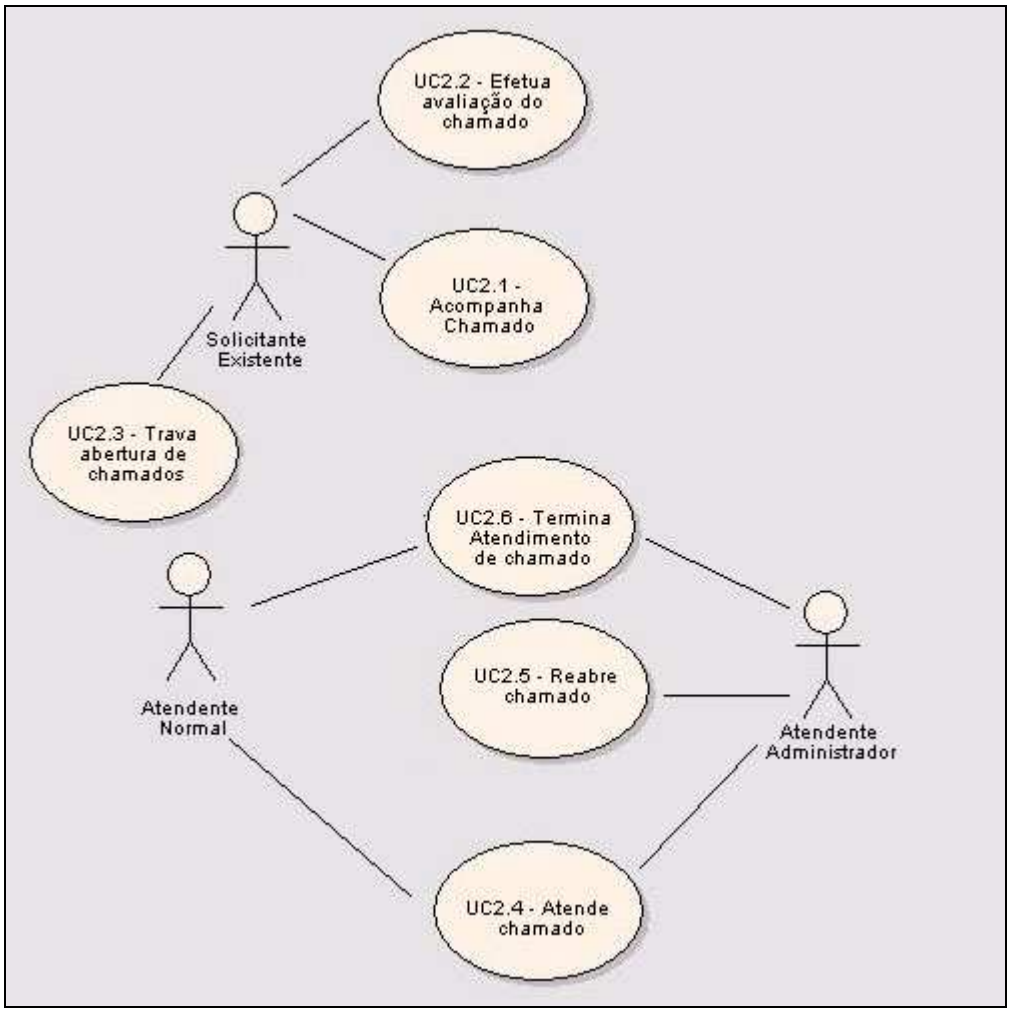


Figura 9 – Casos de uso do módulo de atendimento

3.2.3.1 UC2.1 – Acompanhamento de Chamado

Este caso de uso permite que o usuário acompanhe o andamento de seus chamados, podendo inclusive adicionar comentários ao mesmo.

O cenário principal para este caso de uso é descrito a seguir:

a) solicitante seleciona um chamado;
b) solicitante visualiza dados do chamado na aba “Chamados”;
c) solicitante insere comentários na aba “Comentários” para os atendentes;
d) solicitante visualiza todos os comentários disponíveis;
e) solicitante visualiza todas as tarefas que fazem parte do chamado, na aba “Tarefas”;
f) o caso de uso é encerrado.

No cenário alternativo “Inserir”, caso no passo “c” do cenário principal o usuário só poderá inserir informações enquanto o chamado não for finalizado. Quando finalizado, o sistema mostra apenas a lista de comentários, retirando o combo para preenchimento.

No cenário alternativo “Cancelar”, caso no passo “b” do cenário principal o usuário poderá cancelar o chamado, enquanto o mesmo estiver com o status “Em aberto” e sem nenhuma tarefa em andamento.

3.2.3.2 UC2.2 - Efetua avaliação do chamado

Este caso de uso possibilita ao solicitante cadastrar a avaliação referentes ao chamado. Permite que o solicitante registre o grau de solução e informações que considera relevantes para os atendentes.

O cenário principal para este caso de uso é descrito a seguir:

a) após conectar ao sistema o sistema verifica se existem chamados com status terminado e sem avaliação;
b) existindo, sistema lista todos os chamados terminados;
c) solicitante seleciona o chamado com que traz a imagem de sem avaliação;
d) solicitante entra na tela de acompanhamento de chamados e verifica um formulário de avaliação na aba “Chamados”;
e) solicitante escolhe entre Solucionado e não solucionado;
f) solicitante insere informação caso seja necessário e clica em inserir;
g) o caso de uso é encerrado.

No cenário alternativo “Informação”, caso no passo “e” do cenário principal o solicitante escolher 'Não solucionado' o sistema obrigará o mesmo a inserir uma informação para depois cadastrar a pesquisa.

3.2.3.3 UC2.3 – Trava abertura de chamados

Este caso de uso permite ao sistema travar a abertura de chamados feitas pelo solicitante. O sistema irá verificar se existem chamados com status terminados e sem avaliação, caso isto seja verdadeiro o solicitante não abrirá chamados até que preencha a

avaliação dos chamados terminados e sem avaliação.

O cenário principal para este caso de uso é descrito a seguir:

a) após conectar ao sistema o sistema verifica se existem chamados com status terminado e sem avaliação;
b) existindo, sistema lista todos os chamados terminados;
c) solicitante clica no botão ou no link “Abrir Chamados”;
d) sistema informa que não é possível abrir chamados e informa como o solicitante deve proceder;
e) solicitante segue os passos e executa o UC 1.6;
f) sistema destrava os links e permite a abertura de chamados;
g) o caso de uso é encerrado.

3.2.3.4 UC2.4 – Atende chamado

Este caso de uso permite aos atendentes dar inicio aos chamados que estão no sistema. Todas as informações sobre o processo utilizado para sua solução devem ficar armazenadas no atendimento. Após terminado, o atendente insere a solução e aguarda a avaliação do solicitante. Para isso o atendente deve estar previamente cadastrado.

O cenário principal para este caso de uso é descrito a seguir:

a) atendente escolhe um chamado em aberto que esteja direcionado para seu grupo, equipe ou sua responsabilidade;
b) atendente vai na aba tarefas e clica em “assumir”;
c) sistema verifica se o chamado não possui data de Inicio e coloca a data de inicio da tarefa para o chamado;
d) sistema altera Status do chamado para em andamento;
e) atendente pode alterar Tipo, Área, Status, Prioridade, Resumo, Aberto Em, Data Limite (informando motivo), Solicitante, Objeto;
f) administrador pode alterar tudo relacionado acima e mais: Início, Término, Limite (sem justificativa);
g) o sistema não permite que retorne para Status Em aberto, após ter a primeira tarefa atendida;
h) atendente pode inserir comentários para o solicitante ou para uso interno, este será

visto apenas pelos atendentes;

i) atendente pode criar novas tarefas para outros usuários;
j) sistema não permite a criação de duas tarefas para o mesmo atendente, enquanto o mesmo tiver uma em andamento;
k) atendente não pode dar continuidade a tarefa de outro atendente, quando a mesma estiver com o status diferente de “Em aberto”;
l) atendente pode excluir tarefas que não estiverem terminadas;
m) administrador pode excluir qualquer tarefa de qualquer atendente;
n) o caso de uso é encerrado.

No cenário alternativo “Sair”, caso no passo “b” do cenário principal caso o usuário clique na opção Sair, o sistema fecha a janela de tarefa e desconsidera qualquer alteração.

No cenário alternativo “Gravar”, caso no passo “b” do cenário principal caso o usuário clique na opção Gravar, o sistema grava todas as alterações, mas não inicia o chamado.

3.2.3.5 UC2.5 – Reabre chamado

Este caso de uso possibilita ao Administrador reabrir um chamado que foi terminado pelo atendente, mas o solicitante avaliou como não solucionado.

O cenário principal para este caso de uso é descrito a seguir:

a) administrador procura por chamados terminados com avaliação: “Não Solucionado”;
b) administrador entra na tela de atendimento de chamados;
c) sistema apresenta o campo de Reabertura de chamado na aba Chamados;
d) sistema altera *Status* do chamado para em andamento;
e) sistema altera o campo *Status* do chamado para “Em andamento” e cria uma nova tarefa para o grupo responsável;
f) o caso de uso é encerrado.

3.2.3.6 UC2.6 – Termina Atendimento de chamado

Este caso de uso possibilita ao atendente / administrador terminar o chamado ao qual estavam fazendo atendimento. A partir deste cenário o chamado chega ao seu ciclo final.

O cenário principal para este caso de uso é descrito a seguir:

a) atendente / administrador abrem a tarefa;
b) atendente / administrador clica no botão Terminar;
c) sistema encerra a tarefa e caso não exista mais nenhuma tarefa não terminada ou cancelada, direciona para a aba de “Chamados”;
d) atendente / administrador cadastram a solução utilizada no chamado e clicam em terminar;
e) sistema encerra o chamado, com a data final igual a atual;
f) o caso de uso é encerrado.

3.2.4 Diagrama de atividades

O diagrama de atividades a seguir contempla a descrição do caso de um travamento de abertura de chamado. O processo inicia quando o sistema encontra algum chamado com o status terminado e sem avaliação de grau de solução e termina no momento em que o solicitante informa o grau de solução do chamado. Mais detalhes pode ser visto na figura 10.

O usuário somente será encaminhado para a tela de Abertura de chamados quando não existir mais pendências em relação à avaliação do grau de solução.

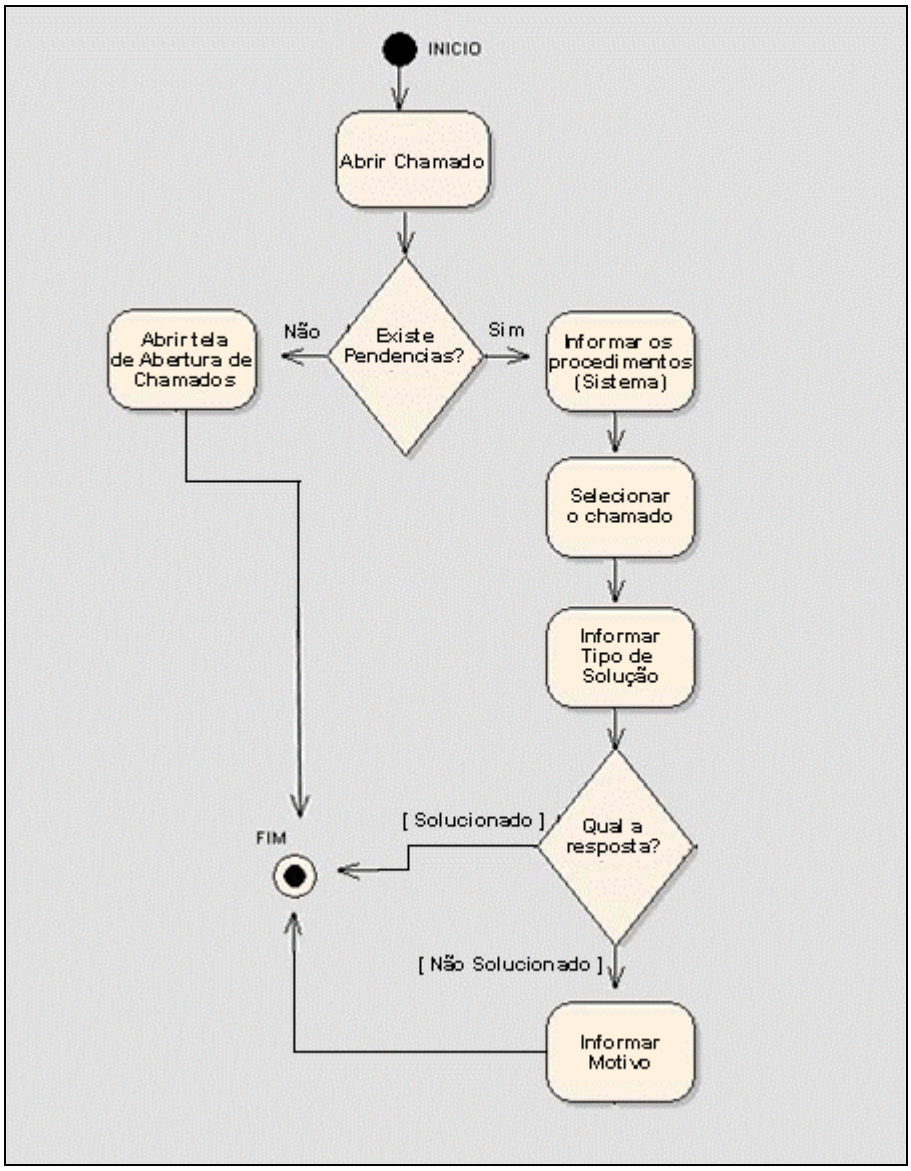


Figura 10 – Diagrama de atividades

3.2.5 *Wireframe* do sistema *HelpDesk*

O *wireframe* a seguir contempla o esboço da tela de consulta de chamados na área técnica. Foram criadas área distintas dentro do leiaute, facilitando a aprendizagem do usuário. O topo sempre mostrará a logomarca do sistema e os menus principais, seguido pela barra de informações (com nome, etc.). A seguir começa o “miolo”, apresentando o título da página, ferramentas de pesquisa e a listagem dos mesmos. Outros detalhes podem ser vistos na figura 11.

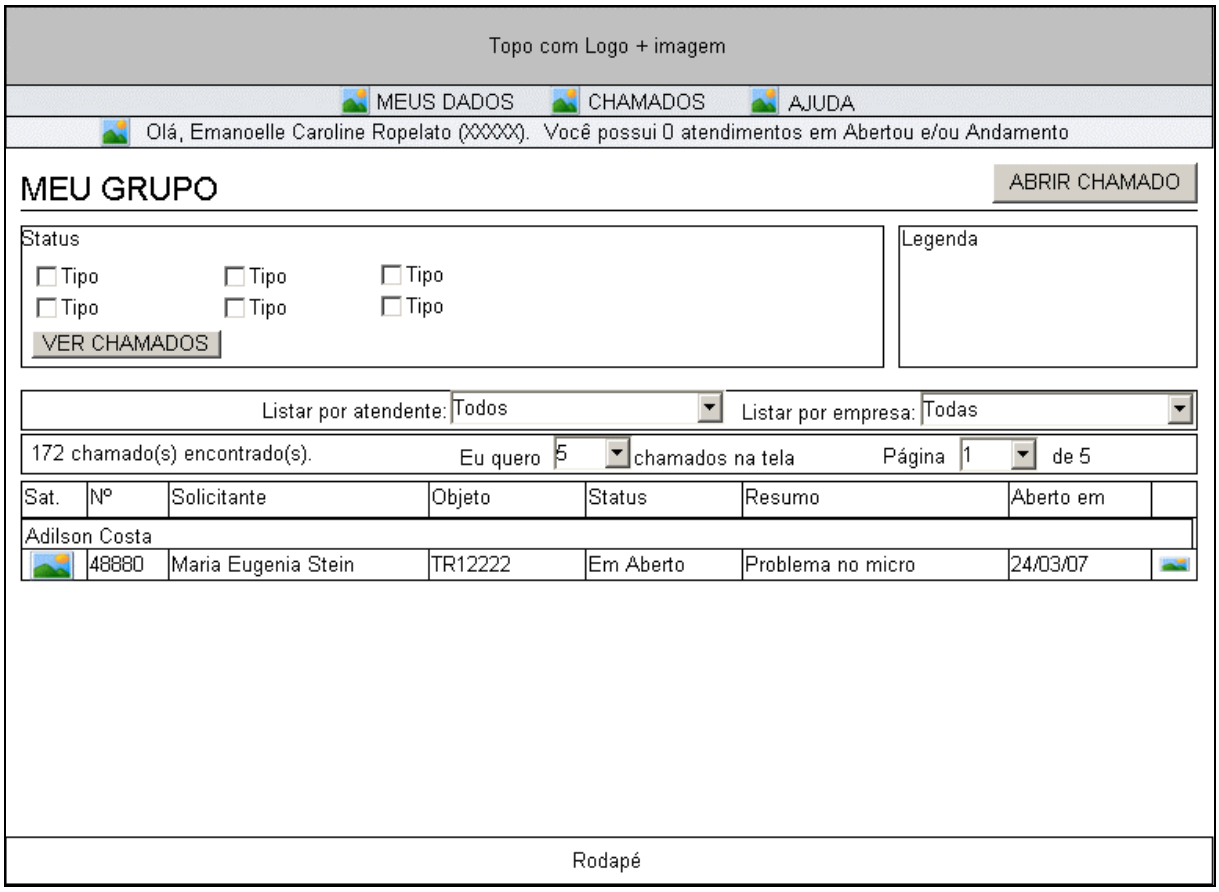


Figura 11 – *Wireframe* de uma página do sistema *HelpDesk*

3.2.6 Diagrama de Navegação

Este diagrama, desenvolvido utilizando a ferramenta ConceptDraw, mostra os caminhos entre as telas do sistema. Pode-se notar que não foi feita distinção entre a funcionalidade, a preocupação foi em mostrar como será a navegação entre uma página e outra, listando os caminhos que os dois tipos de usuários podem percorrer e apresentando uma noção do comportamento do sistema.

A figura 12 apresenta o esquema geral de navegação. Cada página da *WebApp* é representada pelos retângulos, os traços de ligação entre cada página são os *links* e os blocos representam a divisão de conteúdo de cada página. As linhas pontilhadas apresentam o agrupamento das páginas, sendo possível visualizar como será a navegação e o conteúdo disponível para cada nível de usuário do sistema.

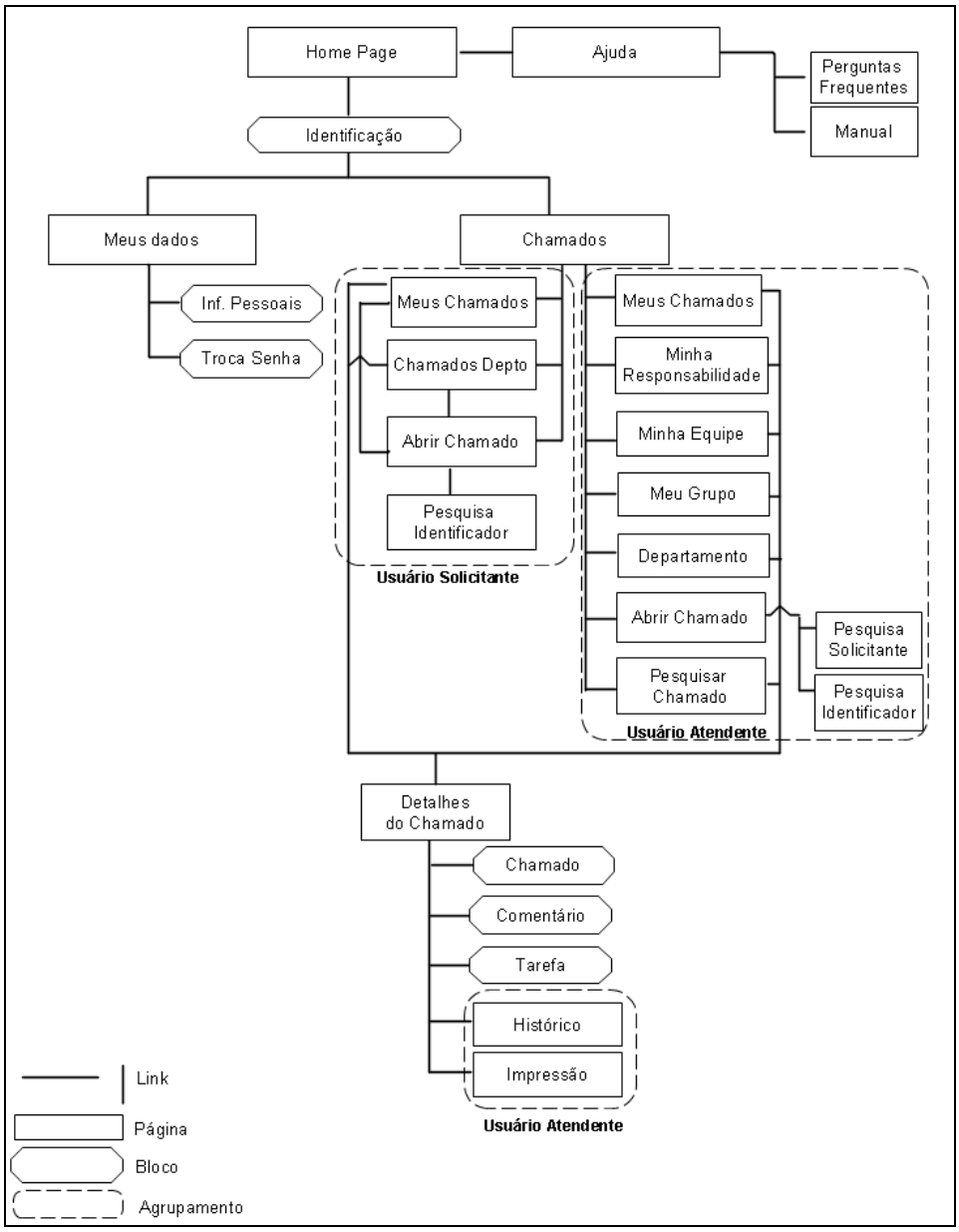


Figura 12 – Diagrama de navegação do sistema *HelpDesk*

3.2.7 Diagrama de Arquitetura de Conteúdo

O diagrama apresenta o tipo de arquitetura do conteúdo que o sistema apresenta. Através deste diagrama é possível visualizar como está estruturado o *HelpdDesk* e se realmente será uma maneira proveitosa de trabalhar a ferramenta. Após a montagem, viu-se que se tratava de uma estrutura em rede, aonde pode-se navegar de um local para outro sem se preocupar com nível hierárquico, conforme mostra a figura 13.

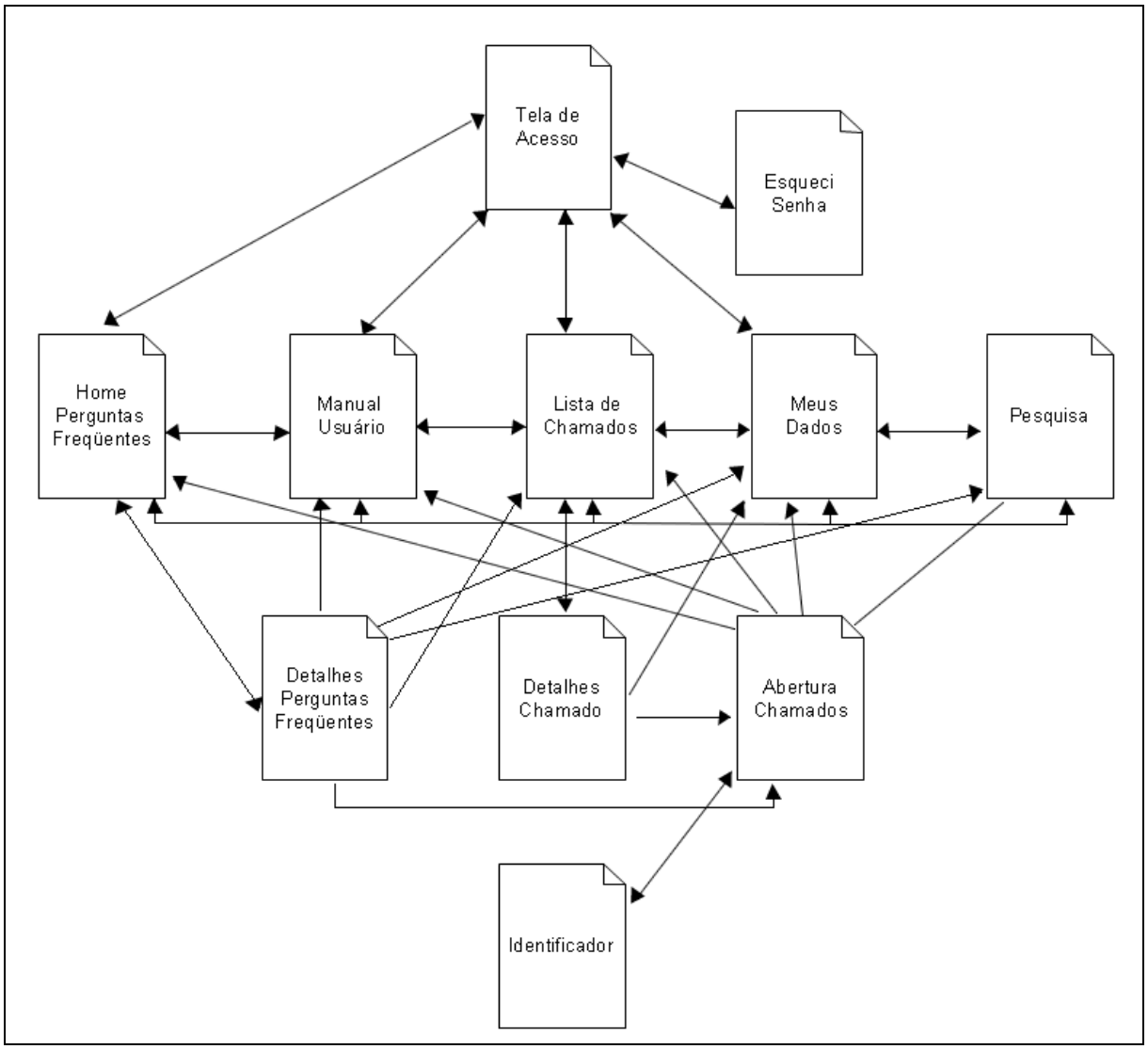


Figura 13 – Diagrama de arquitetura de conteúdo do sistema *HelpDesk*

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as principais técnicas e ferramentas utilizadas nas fases de desenvolvimento, bem como as tecnologias utilizadas. Também são abordados a operacionalidade do software e os resultados obtidos.

3.3.1 Técnicas e ferramentas utilizadas

Nos tópicos a seguir são apresentadas as principais técnicas utilizadas na produção deste trabalho, utilizou-se a linguagem de programação PHP na versão 5.0.

3.3.1.1 DreamWeaver e FireWorks 8

O DreamWeaver foi utilizado neste trabalho, principalmente nas fases de implementação, validação e testes. O DreamWeaver é uma ferramenta comercial para construção de aplicações Web, proporcionando ao usuário criar páginas sem escrever uma linha de código HTML, criar conteúdo dinâmico usando tecnologias de *script* tais como PHP, ASP e ASP.NET e conectar banco de dados como MySQL e Microsoft Access, como mostrado na figura 14.

O FireWorks 8 é uma ferramenta comercial e foi utilizada na fase de desenvolvimento do interface da aplicação e de implementação. É uma aplicação para criação, otimização e exportação de ilustrações de Web, que podem facilmente ser exportados do FireWorks para projetos DreamWeaver, facilitando a produtividade com importantes melhorias em desempenho e usabilidade.

Estas ferramentas fazem parte do Studio 8, um pacote de ferramentas para desenvolvimento Web, produzida pela Macromedia. Desde o final dos anos 90, o DreamWeaver vem tendo um sucesso crescente e hoje domina cerca de 80% do mercado de editores HTML (Adobe, 2007). Todo este sucesso fez com que a Adobe adquirisse a Macromedia em 2005.

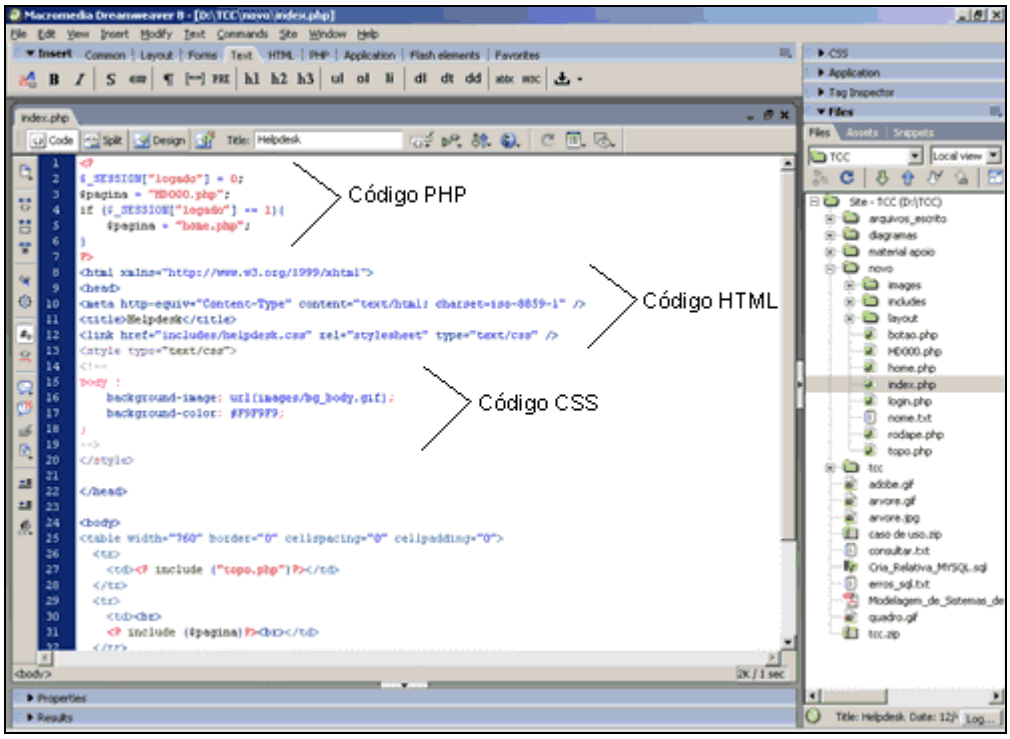


Figura 14 – Ambiente de desenvolvimento *DreamWeaver* 8

3.3.1.2 ConceptDraw Web Wave

Utilizada na fase de análise do projeto a ferramenta auxiliou no desenvolvimento de diagramas específicos para *WebApps*, tais como diagrama de navegação, fluxo de navegação entre outros. O software comercial, é projetado para modelagem de Web *Sites* e Aplicações sendo uma de suas características a quantidade de *wizards* que conduzem à seqüência apropriada das ações e ajudam a lidar com as tarefas mais complexas nas aplicações para internet, conforme a figura 15.

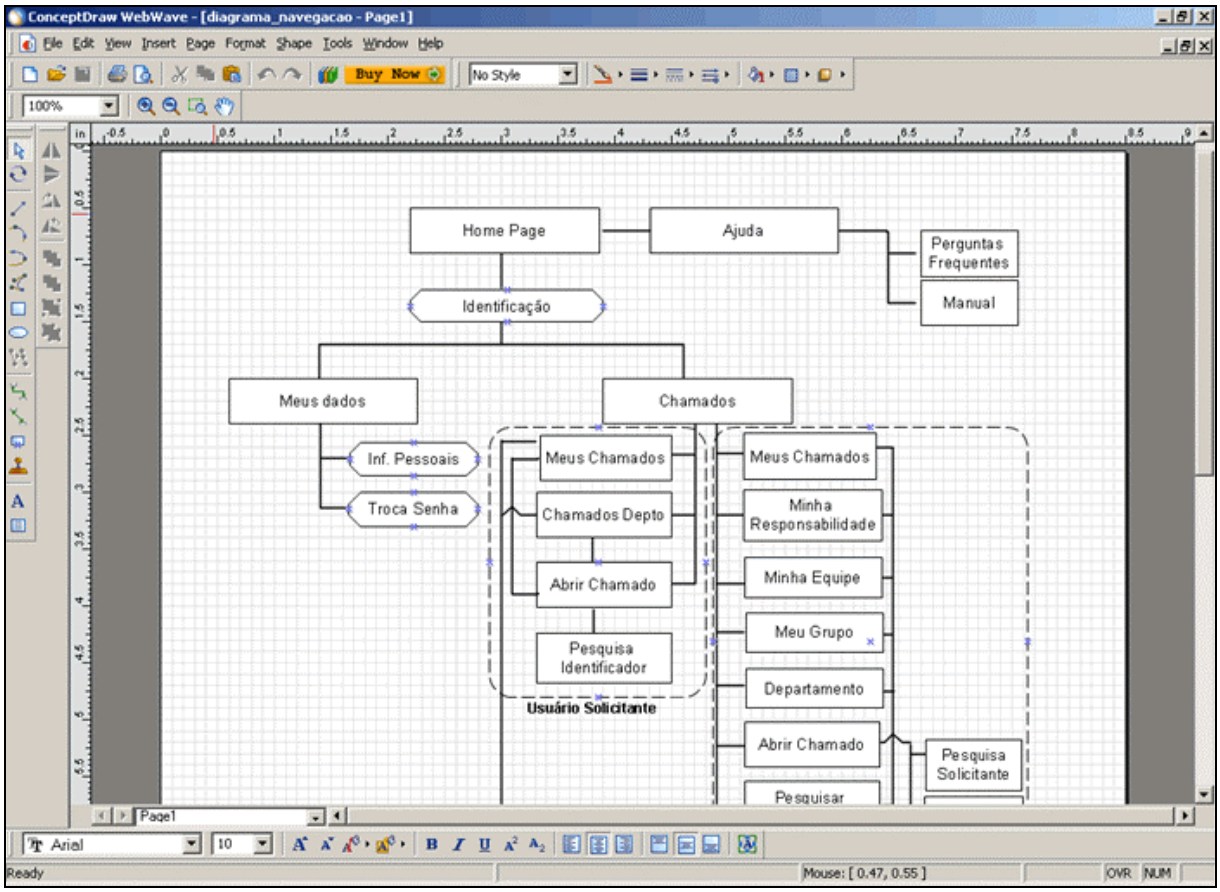


Figura 15 – Diagrama de navegação no ambiente ConceptDraw

3.3.1.3 *Axure* RP Pro

A ferramenta foi de grande auxílio durante a fase de análise do projeto para a construção dos *wireframes*, conforme a figura 16, para a *WebApp*. Permite criar *wireframes* e especificações para aplicações e Web *Site*s mais rápidos e mais fáceis do que criando imagens estáticas com as ferramentas atuais.

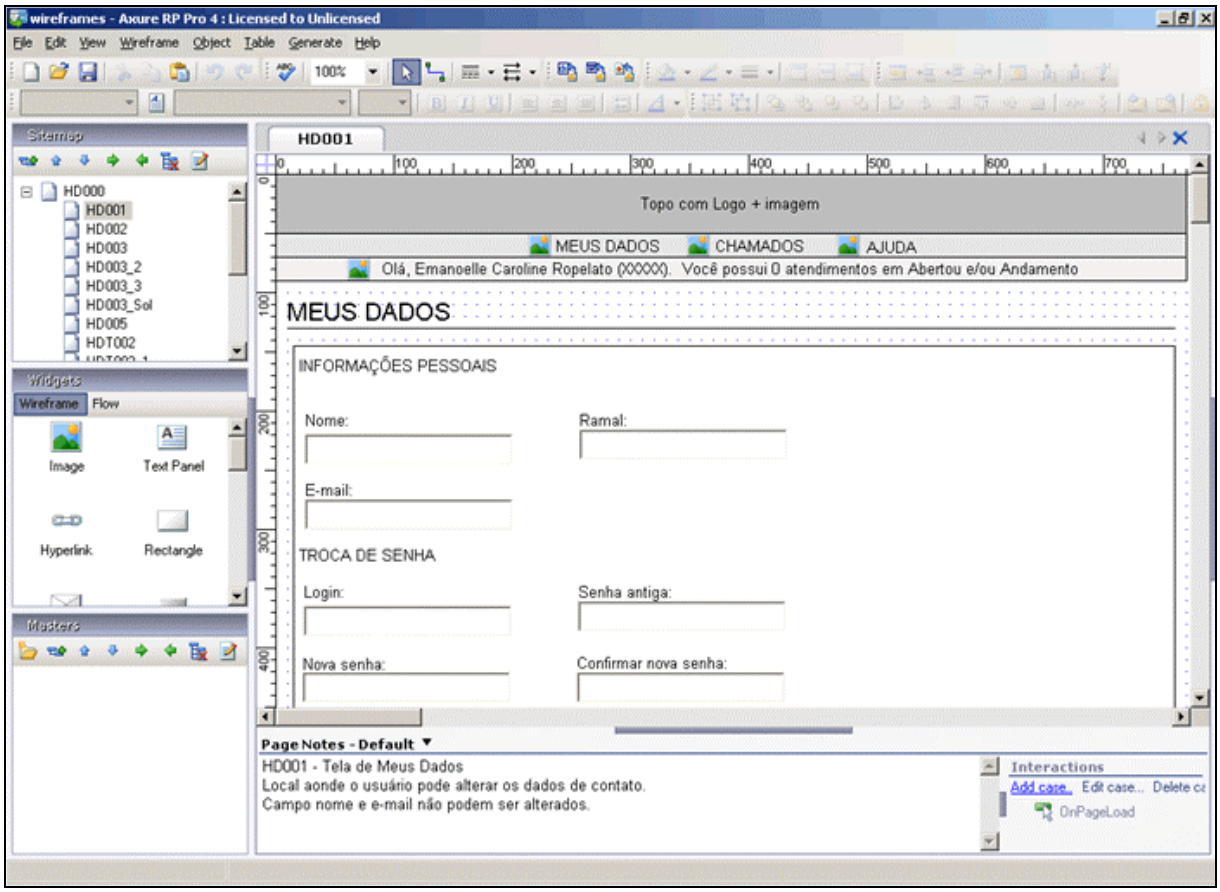


Figura 16 – *Wireframe* no ambiente Axure

3.3.1.4 Enterprise Architect

O Enterprise Architect é uma ferramenta CASE comercial e foi utilizada nas fases de elicitação de requisitos e análise para definição dos casos de uso, especificação de requisitos e regras de negócio e criação de diagramas de atividades.

O EA possui muitos recursos tais como:

a) geração de código e engenharia reversão para Java;

b) modelagem em UML 2.0;

c) exportação e importação para outras ferramentas CASE via XML.

3.3.1.5 MySQL

Como a estrutura original estava em Oracle, um banco de dados comercial, foi de

grande importância transferi-lo para um banco de dados livre. A escolha pelo MySQL, como base de dados foi feita a partir da linguagem utilizada na implementação, PHP 5.0, justamente porque ambos trabalham muito bem em conjunto (ALECRIM, 2007).

O MySQL é um sistema de gerenciamento de banco de dados que utiliza a linguagem SQL como interface, otimizado para aplicações Web bastante utilizado na internet.

Entre as características técnicas do MySQL, destacam-se:

a) alta compatibilidade com linguagens como PHP, Java, Python, C#, Ruby e C/C++;
b) baixa exigência de processamento;
c) recursos de transações, conectividade segura, indexação de campos de texto, replicação, etc;
d) instruções em SQL, como indica o nome.

3.3.1.6 PHP 5.0

PHP é uma linguagem dinâmica, gratuita, que permite fazer várias construções e que fornece uma excelente biblioteca para programação Web e ainda componentes nativos, não dependendo de componentes externos para algumas funcionalidades básicas. Conecta-se a vários bancos de dados disponíveis no mercado, o que torna sua utilização uma das melhores opções para desenvolvimento de aplicações Web. Por este motivo, é uma das ferramentas mais populares para o desenvolvimento de *WebApps*. Hoje é utilizada, conforme Lozano (2003, p. 27), “em aplicações complexas e *sites* de grande volume, portanto se beneficia da aplicação das soluções comprovadas e da maior produtividade do desenvolvimento em equipe”.

Com o lançamento da versão 5.0; tornou-se ainda mais poderosa entre as demais linguagens para Web, apresentando um modelo Orientado a Objetos (OO) completo, equivalente ao Java e C++. A OO permitiu a utilização da maioria dos padrões *Gang of Four* (GoF) e *Java 2 Enterprise Edition* (J2EE) quando adaptados as particularidades da linguagem PHP.

PHP já possui alguns *patterns* de construção de *scripts*, sendo alguns deles: PHP *Cache* - salvam em *cache* qualquer informação de um *script* PHP; Testes Unitários - garantem que *bugs* possam ser isolados em componentes funcionais da aplicação de modo objetivo; Autenticação pelo Servidor - salva *bookmarks* em qualquer página da aplicação, autenticando sem necessidade de código na aplicação (LOZANO, 2003, p. 52, 56, 58).

3.3.1.7 *Patterns* Utilizados

Foram utilizados na fase de codificação do projeto e selecionados a partir da necessidade da aplicação focando na interface, na funcionalidade e na comunicação usuário x sistema e aplicação x aplicação. Estes *patterns* fazem parte de bibliotecas conceituadas como a *Patterns for* PHP e *WebPatterns*.org, entidades sérias que buscam o aprimoramento das aplicações Web, e considerados métodos corretos de desenvolvimento Web pelas comunidades especializadas.

Utilizando os *WebPatterns* o código fonte do novo *HelpDesk* sofreu grandes alterações comparado com a estrutura antiga. Algumas dessas mudanças estão apresentadas abaixo.

O método de conexão ao banco de dados foi implementado utilizando o *pattern Singleton*, apresentada na figura 17.

```
class Conexao
{
    static private $instancia;

    // Construtor privado para conectar a base de dados
    private function __construct()
    {
        //Realiza a conexão
        $conectar = mysql_connect('localhost', 'usuarioHelpdesk', 'helpdesk');
        if (!$conectar) {
            //Ocorreu problemas na conexão, mostra erro na tela e e para a execução da função
            die('Não foi possível conectar: ' . mysql_error()."<br>");
        }
        //Seleciona o banco de dados
        $db_selected = mysql_select_db('db', $conectar);
        if (!$db_selected) {
            // não encontrou o banco informado, mostra erro na tela e para a execução da função
            die ('Não foi possível selecionar : ' . mysql_error());
        }
    }
    // ******************* Método singleton ******************
    //Responsável pelaverificação de existência de uma instância dessa classe previamente utilizada.
    // Se existir retorna a mesma , senão ccria uma nova instância.
    static public function singleton()
    {
        if (!isset(self::$instancia)) {
            $c = __CLASS__;
            self::$instancia = new $c;
        }

    // ******************* Método consultar ******************
    //Responsável pela execução das cláusulas SQL passadas para o método. Ao final retorna os
resultados encontrados.
    public function consultar($sql)
    {
        $result = mysql_query($sql);
        if (!$result) { // Não encontrou nenhum resultado
            die('<br><br>Consulta não encontrou resultados: ' . mysql_error());
        }
    }
```

Figura 17 – Código para conexão ao banco de dados usando *Singleton*

A ferramenta original era implementada conforme a figura 18.

```
/** Configuração para ODBC Oracle */
/** criando sessoes de conexao para o helpdesk**/
$sys['db']['driver'] =    'oci8';              /** Driver do Banco de Dados */
$sys['db']['host'] =      'localhost';         /** Endereço do Servidor de Banco de Dados */
$sys['db']['name'] =      'db';                /** Nome do DataBase ml*/
$sys['db']['user'] =      'usuarioHelpdesk';   /** Usuário de Conexão do Banco de Dados */
$sys['db']['password'] = 'helpdesk';           /** Senha do usuário para a conexão do Banco de Dados */


// Instancia o objeto de conexao
$db = NewADOConnection( $sys['db']['driver'] );
// Faz a conexao
$db->Connect(   $sys['db']['host'], $sys['db']['user'], $sys['db']['password'], $sys['db']['name']);
//$db->debug=true;
```

Figura 18 – Código para conexão ao banco de dados

Apesar de parecer menos complexa, a estrutura antiga apresentava uma chamada para a conexão muito ultrapassada e foi desenvolvida através de uma extensão PHP que gera códigos automaticamente e de difícil manutenção por parte do programador. Para todas as páginas que conectavam o banco de dados era necessário passar todas as informações do banco, usuário e senha. Na nova aplicação esta estrutura já esta dentro da programação e com isto o programador não precisa estar sempre passando estes parâmetros, a comparação apresentada na figura 19 mostra esta diferença.

Aplicação Atual

```
include_once ('conexao.php'); //arquivo aonde estão as configurações de conexão

// ************** Utilizando conexão ao banco de dados. ****************************

// Instancio o Método singleton dentro da class Conexao
$conexaoBD = Conexao::singleton();
//Conectando a tabela de noticias
$conexaoBD->consultar('Select Nome,Conteudo,Id,Data,Autor From Noticias Where Destacar=1');
```

Aplicação Antiga

```
include_once ('global.php');

// Instancia o objeto de conexao
$db = NewADOConnection( $sys['db']['driver'] );
// Faz a conexao
$db->Connect($sys['db']['host'], $sys['db']['user'], $sys['db']['password'], $sys['db']['name']);
$sql = "Select Nome,Conteudo,Id,Data,Autor From Noticias Where Destacar=1";
$resultado = $db->Execute($sql);
```

Figura 19 – Comparação entre aplicações na classe de conexão ao banco de dados

O *FastTemplate* foi utilizado para separar os scripts de programação da interface gráfica da aplicação. A aplicação foi modelada usando três camadas: de programação, onde estão todos os scripts em php; modelo dinâmico, responsável pela geração das linhas de retorna da camada de programação; modelo estático, que apenas apresenta o resultado final para o usuário. Depois de modelada foi possível verificar o grande ganho na estrutura da aplicação e na manutenção do mesmo. A figura 20 mostra como o código da página de

Perguntas freqüentes ficou menos complexo e prático.

Camada de programação

```php
<?php
/* templateFAQ.php
 Script responsável pela programação da área de FAQ*/
include_once('class.FastTemplate.php3'); //Carregando classe FastTemplate
$model = new FastTemplate('./templates'); // Seta a pasta aonde estão disponíveis os templa

$model -> define (
   array ( 'pagina' => 'Lista.html',    // template para area estática da aplicação
           'linhas' => 'itemLista.html' // template para cada linha de registro
         )
);
// Instancio o Método singleton dentro da class Conexao
$conexaoBD = Conexao::singleton();
//Conectando a tabela de FAQ
$RS_FAQ = $conexaoBD->consultar('Select Id,Nome From FAQS order by Nome');

while ($FAQ = mysql_fetch_object($RS_FAQ)) {
   // As variaveis {ID,NOME} estarão configuradas dentro do template
   // utilizado em cada linha de registro do BD.
   $model->assign('{ID}',$FAQ->Id);
   $model->assign('{NOME}',$FAQ->Id);

   // a variável {LINHAS} aparecerá no modelo estático do HTML e será substituído
   // pela concatenação de cada processamento sobre o handle ".linhas".
   $model->parse('{LINHAS}','.linhas');

}

$model->parse('OUTPUT','pagina');
$model->FastPrint('OUTPUT');
?>
```

Camada dinâmica

```html
<!--
ARQUIVO itemLista.html
MONTA A ESTRUTURA DAS LINHAS BASEADO NO RESULTADO DA  CONSULTA PHP-->
<tr>
  <td bgcolor="#EFF3FC">
    <img border="0" src="images/seta-azul.gif"><a href="FAQ.php?Id={ID}" class="txt_conteudo">{NOME}</a>
  </td>
</tr>
```

Camada estática

```html
<!--
ARQUIVO Lista.html
ÁREA ESTATICA DA APLICAÇÃO MOSTRA OS ELEMENTOS HTML VISUALIZADOS PELO USUÁRIO FINAL-->
<html>
<body>
<table width="98%" border="0" align="center" class="txt_conteudo">
  <tr>
    <td height="26"><span class="tit_sessao"><b>Perguntas freq&uuml;entes</b></span><br />
        <img src="images/linha.gif" width="571" height="5" border="0" /></td>
  </tr>
  <tr>
    <td>
      <table width="100%" border="0" align="center" cellpadding="0" cellspacing="1" bgcolor="#BED3F0" class="txt_conteudo">
        <!-- MOSTRA OS RESULTADOS OBTIDOS NO ARQUIVO itemLista.html-->
        {LINHAS}
      </table>
    </td>
  </tr>
  <tr>
    <td></td>
  </tr>
</table>
</body>
</html>
```

Figura 20 – Aplicação do *FastTemplate*

A programação com a utilização do *FastTemplate* fez com que a aplicação fosse facilmente modificada por programadores ou por *designers*, uma vez que um não necessita obrigatoriamente do outro para dar continuidade ao seu trabalho já que o fonte HTML esta separado do fonte PHP. A figura 21 mostra como era complicada a manutenção da página anteriormente, uma vez que os códigos estavam misturados e qualquer alteração errada

poderia prejudicar o funcionamento do mesmo.

```
<?
//Conexão com banco de dados
$db = NewADOConnection( $sys['db']['driver'] );
$db->Connect($sys['db']['host'],$sys['db']['user'],$sys['db']['password'],$sys['db']['name']);

//Select e retorno dos resultados
$sql = "Select Id,Nome From FAQS order by Nome";
$resultado = $db->PageExecute($sql,$nTotalRegsPagina,$nPagina);
$nTotalRegs = $resultado->MaxRecordCount();

?>
<table width="98%" border="0" align="center" class="txt_conteudo">
  <tr>
    <td height="26"><span class="tit_sessao"><b>Perguntas freq&Uuml;entes</b></span><br>
      <img src="images/linha.gif" width="571" height="5" border="0"></td>
  </tr>

  <tr>
    <td>
    <!-- PROGRAMAÇÃO PARA MOSTRAR A LISTA DE REGISTROS NA TELA-->
      <? if($nTotalRegs >= 1 ){ ?>
<table width="100%" border="0" align="center" cellpadding="0" cellspacing="1" bgcolor="#BED3F0" class="txt_conteudo">
          <tr>
            <td><table width="100%" border="0" cellpadding="3" cellspacing="0" bordercolor="#FFFFFF" bgcolor="#FFFFFF">
     <? } ?>
        <?
      for ($i=0;$i<$nTotalRegs;$i++){
        if ($resultado->EOF){
          //$i==$nTotalRegs;
        }else{
        ?>
              <tr>
                <td bgcolor="#EFF3FC">
                    <img border="0" src="Arquivos/seta-azul.gif"> <a href="FAQ.php?Id=<? echo $resultado->fields[0] ?>" class=
"txt_conteudo"><? echo  $resultado->fields[1] ?></a>                </td>
              </tr>

          <?
          $resultado->MoveNext();
        }
      }
      $resultado->Close();
    ?>
    <? if($nTotalRegs >= 1 ){ ?>
        </table></td>
       </tr>
     </table>
    <? }?>
   </td>
  </tr>
  <tr>
    <td></td>
  </tr>
</table>
</td></tr><tr><td colspan="2" height="255">
```

Figura 21 – Fonte da aplicação antiga

Como a aplicação pode ser configurada para outros bancos de dados, foi utilizado o método *factory* para carregar em tempo de execução os *drivers* de banco de dados. O código utilizando este padrão ficou claro e fácil de manter, conforme a figura 22.

Chamado do Método Factory

```php
<?

// Carregar um driver MySQL
$mysql = driversUsados::drivers('MySQL');

// Carregar um driver Oracle
$Oracle = driversUsados::drivers('oci8');

class Conexao
```

Classe Factory

```php
<?
class driversUsados
{
    // Usando metodo Factory
    function &drivers($nomeDrive)
    {
        if (include_once 'dbconn/' . $nomeDrive . '.php') {
            $classname = 'ADODB_' . $nomeDrive;
            return new $classname;
        } else {
            throw new Exception ('Driver não encontrado');
        }
    }
}
?>
```

Figura 22 – Fonte da aplicação utilizando *Factory*

O código anterior apresenta um código difícil para fazer uma alteração, isto porque a ferramenta anterior foi gerada usando uma extensão do DreamWeaver que gera bibliotecas de classes automáticas. São vários arquivos de conexão e no momento da manutenção o programador perde muito tempo em localizar aonde consertar o problema, como mostra a figura 23.

```php
function ADOLoadCode($dbType)
{
global $ADODB_LASTDB;

    if (!$dbType) return false;
    $db = strtolower($dbType);
    switch ($db) {
        case 'maxsql': $db = 'mysqlt'; break;
        case 'postgres':
        case 'pgsql': $db = 'postgres7'; break;
    }
    @include_once(ADODB_DIR."/drivers/adodb-".$db.".inc.php");
    $ADODB_LASTDB = $db;

    $ok = class_exists("ADODB_" . $db);
    if ($ok) return $db;

    $file = ADODB_DIR."/drivers/adodb-".$db.".inc.php";
    if (!file_exists($file)) ADOConnection::outp("Missing file: $file");
    else ADOConnection::outp("Syntax error in file: $file");
    return false;
}
```

Figura 23 – Fonte da classe de *drivers* antiga

A estrutura de interface da aplicação foi divida em três partes: topo; conteúdo e rodapé. Sendo que as três são chamadas em um mesmo arquivo pai. Nele são configuradas as chamadas para as funções de scripts que auxiliam a execução da ferramenta. Aplicou-se nesta página o *pattern* de *builder*, conforme a figura 24.

```php
function ADOLoadCode($dbType)
{
global $ADODB_LASTDB;

    if (!$dbType) return false;
    $db = strtolower($dbType);
    switch ($db) {
        case 'maxsql': $db = 'mysqlt'; break;
        case 'postgres':
        case 'pgsql': $db = 'postgres7'; break;
    }
    @include_once(ADODB_DIR."/drivers/adodb-".$db.".inc.php");
    $ADODB_LASTDB = $db;

    $ok = class_exists("ADODB_" . $db);
    if ($ok) return $db;

    $file = ADODB_DIR."/drivers/adodb-".$db.".inc.php";
    if (!file_exists($file)) ADOConnection::outp("Missing file: $file");
    else ADOConnection::outp("Syntax error in file: $file");
    return false;
}
```

Figura 24 – Fonte da aplicação usando *builder*

A estrutura anterior para esta página era muito confusa, misturava códigos *javascript*, PHP e HTML. Essa mistura prejudicava o trabalho do *designer* que poderia, sem querer,

apagar algum componente, colocando em risco o funcionamento do HelpDesk, conforme figura 25.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title><? echo $sys['site']['TituloSite'] ?></title>
<link href="Arquivos/helpdesk.css" rel="stylesheet" type="text/css" />
<script language="javascript" src="Arquivos/funcao.js"></script>
</head>

<body  <? if (($MontraUnLoad == 1) && ($reabre==1)){?>onUnload="reabre_tarefa('<? echo $nId?>')" <? }
?>>
<? if (($verfone==1) &&($sys['usuario']['Ramal'] == "")){ ?>
        <script language=javascript>
        function pagina(txt){
            return typeof(suporteVBscript)=="undefined"?confirm("Seu ramal não foi informado.\nDeseja
informar agora?"):VBconfirm("Seu ramal não foi informado.\nDeseja informar agora?")==6;
        }
        </script>
        <script language=vbscript>
        suporteVBscript=1
        function VBconfirm(mensagem)
            VBconfirm=msgbox(mensagem,vbyesno)
        end function
        </script>
        <script language=javascript>
            if(pagina("Seu ramal não foi informado.\nDeseja informar agora?")) window.location.href=
'meusDados.php';
        </script>
<? }?>
<map name="Map"><area shape="rect" coords="9,21,107,65" href="index.php" alt="Voltar ao Login">
</map>
<table width="760" border="0" cellspacing="0" cellpadding="0">

         <? if ($lista_chamado==1){?>
             <tr>
               <td><img src="image/img_satisfacao.gif" hspace="5" vspace="5"></td>
             </tr>
         <? }?>
     <? //if (($sys['tecnico']==1) && ($sys['sessionId']!=0)){ ?>
     <? //}?>
   </table>
  </td>
   <? } ?>
  <!-- ************************************ Fim Canto esquerdo ************************************
 -->

  <!-- ************************************ Início Centro ************************************ -->
 <td width="615" valign="top" bgcolor="#FFFFFF" height="0"> 
 <!--
 </td>
 </tr> -->
<!-- ************************************ Fim Canto direito ************************************
-->
<!-- ************************************ Começo rodapé ************************************ -->
<!-- <tr>
   <td bgcolor="#FFFFFF"> </td>
   <td bgcolor="#FFFFFF"> </td>
 </tr>
 <tr>
```

Figura 25 – Fonte da aplicação sem utilização de padrão

Para tornar o carregamento das informações mais rápido foi adicionada em alguns retornos de consultas o *Webpattern* Cache *Lite*. Este armazena informações que não precisam ser sempre atualizadas. Na figura 26 esta o código da área que apresenta o nome do usuário,

matrícula e número de chamados do mesmo, que aparece sempre abaixo do menu.

Gravando Cache de Usuário

```
<?
require_once('Cache/Lite.php'); // arquivo da biblioteca PEAR

$options = array(
                "cacheDir" => "cache/",
                "lifeTime" => 5
);
$objCache = new Cache_Lite_Output($options);
// Será processado apenas se não estiver gravado no cache.
// O nome cacheUsuario foi criada para esta aplicação
if (!$objCache->start("cacheUsuario"))
{
    // funcao que grava o nome do usuário, numero do chamado e total de chamados em andamento
    // após a execução da consulta no banco de dados
    grava_Usuario();
    $objCache->end(); // armazena no cache
}
?>
```

Lendo as informações do Cache de usuário

```
  <tr>
    <td colspan="3"><div align="center"><? echo $Cache_Lite->get("cacheUsuario") ?></div></td>
  </tr>
</table>
```

Figura 26 – Fonte da aplicação utilizando *Cache Lite*

Para a comunicação usuário x sistema os *Web Design Patterns* foram mais aplicados, fornecendo ao *HelpDesk* um ambiente mais característico com o ambiente Web. Os *WebPatterns* foram aplicados para que a comunicação entre as classes dos sistema fosse feita de maneira mais correta otimizando o código fonte.

3.3.2 Operacionalidade da implementação

A seguir será apresentada uma operação realizada pelo solicitante através do sistema. Um solicitante encontra problemas em um software instalado em sua máquina. Para que seja resolvida esta situação, o mesmo precisa abrir um chamado pelo sistema de *HelpDesk* Web. Este chamado cairá para os atendentes que tomarão as iniciativas para a correção do problema.

3.3.2.1 Tela de *Login*

Responsável pela validação do usuário e verificação do tipo de usuário que está conectando (Administrador, Atendente, Solicitante Novo, Solicitante).

O Solicitante informa seu login e senha e clica em entrar. A página verifica se todos os campos foram informados e se estão válidos. Se o solicitante informar algum dado incorreto, o sistema retorna o problema em um campo de erros, senão encaminha para a página principal do sistema, conforme mostra a figura 27.

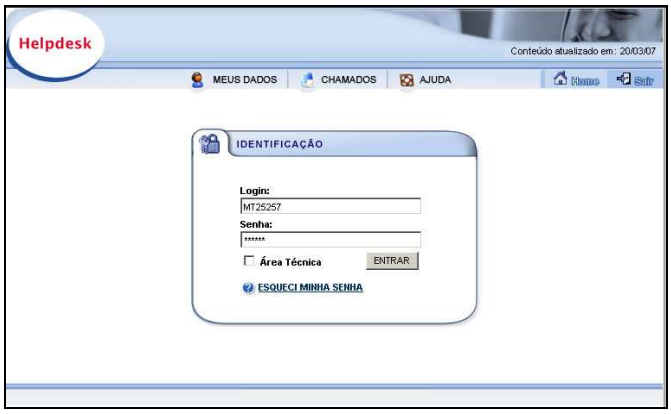


Figura 27 – Tela de *login* do *HelpDesk*

Nesta página foram aplicado os *Web Design Patterns* de formulário e mensagem de erro em formulários, deixando os mesmos mais uniformes a aplicação antiga, conforme comparação do quadro 5.

Padrão	Aplicação atual	Aplicação antiga
Formulário	• botão padronizado como campo de formulário; • mensagens do sistema mais condizente com o problema ocorrido.	• mensagens do sistema confusas; • seguia muito este padrão errando apenas nos detalhes apresentados na coluna ao lado.
Mensagem de erro	• todas as mensagens aparecem acima da área de identificação; • os retornos de erro aparecem somente após a submissão do formulário; • quando ocorre um retorno de erro, o foco do formulário vai para o campo que ocasionou o problema e sua tag muda a cor para vermelho.	• as mensagens aparecem como texto ou em caixas de alertas; • existem retornos antes de submeter o formulário e depois de submetido; • não caracteriza o campo que ocorreu o problema, mas retorna o foco para o mesmo.

Quadro 5 – Melhorias apresentadas na tela de login

A aplicação anterior apresentava os erros em caixas de alertas e no código HTML, este tipo de abordagem, além de confundir o usuário pode ser prejudicado pelo tipo de navegador que pode estar sem o *javascript* habilitado. A aplicação atual mostra o erro diretamente na página HTML em um local padrão para todas as mensagens do sistema para o usuário. Esta diferença esta apresentada na figura 28.

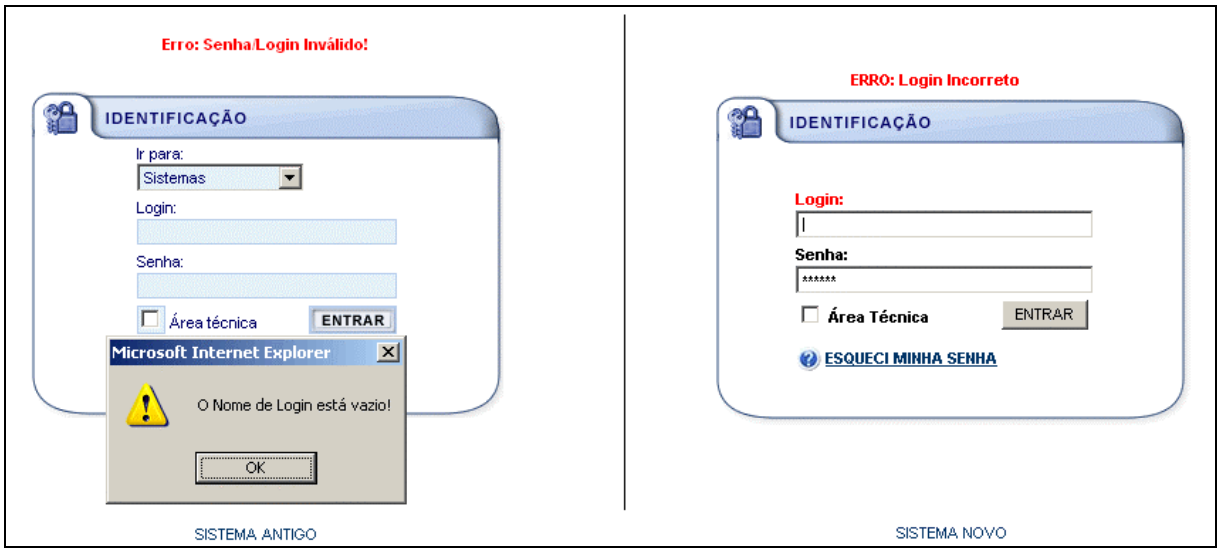


Figura 28 – Comparação da tela de *login* antiga com atual

3.3.3 Tela de Meus Chamados

O solicitante é encaminhado para a página de Meus Chamados, figura 19, que é considerada a *home page* da aplicação. Nela o usuário pode configurar como deseja ver as listagens e navegar em quase todo o sistema. Esta página traz a lista de todos os chamados que foram abertos pelo mesmo.

O solicitante pode escolher através do Status quais os chamados que serão listados na tela, configurando esta opção como padrão através de um *cookie*. Configura também o número de chamados apresentado por página, tornando-se sempre padrão.

A partir desta tela o solicitante tem um acesso rápido para a tela de abertura de chamados, sendo um botão "Abrir Chamados" ou pode entrar na página a partir do menu Chamados -> Abrir Chamados, conforme apresenta a figura 29.

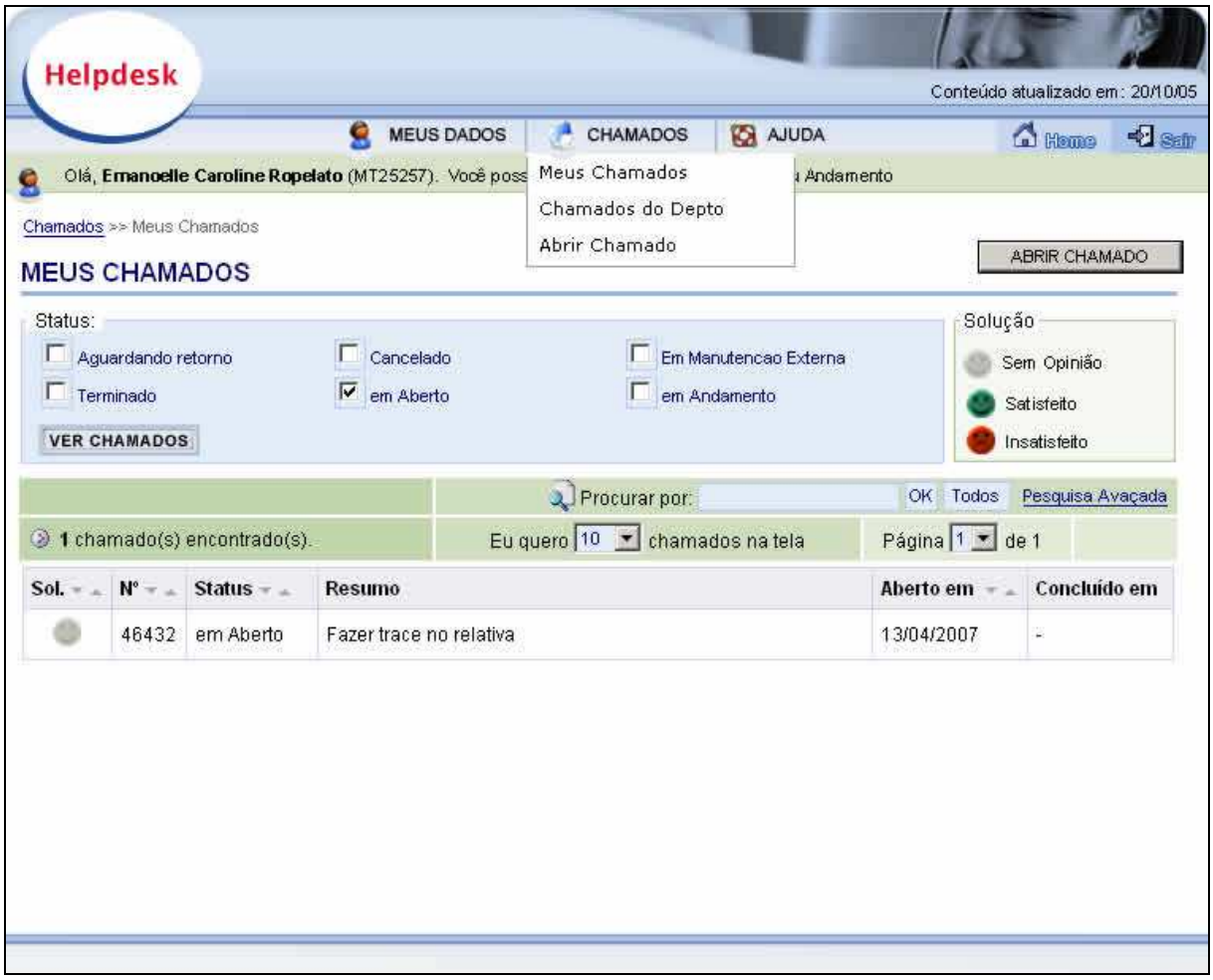


Figura 29 – Tela de Meus Chamados

Como todo o conteúdo interno, esta tela está dividida em topo, conteúdo e rodapé. O topo e o rodapé serão padrões para toda a aplicação formando para o usuário uma linha de raciocínio e aprendizagem. O conteúdo segue o mesmo raciocínio. O solicitante visualiza a hierarquia da página no sistema, abaixo o título da tela e ao lado um botão que determina uma ação. A próxima linha mostra uma seleção de status dos chamados que devem aparecer e uma legenda do grau de satisfação.

Antes da listagem de chamados, são apresentadas duas linhas, sendo uma própria para pesquisas e a outra voltada para a configuração e visualização da lista de chamado. A partir destas divisões o usuário já pode montar a linha de raciocínio para a utilização do site, uma vez que existe separação de pesquisas, consultas, títulos e localização.

Para alcançar o resultado citado, foram utilizados os padrões de *Breadcrumbs*, Área de Pesquisa e Menu *Fly-out*, a aplicação esta demonstrada no quadro 6.

Padrão	Aplicação atual	Aplicação antiga
Menu *Fly-out*	• mostra a mensagem diretamente na página; • as intenções da página estão melhor representadas. • lista todas as páginas que podem ser acessadas sem hierarquia; • o usuário consegue encontrar todos os itens diretamente no menu.	• lista todas as páginas que podem ser acessadas sem hierarquia; • o problema é que não segue um padrão e a qualquer momento pode ser alterado e desconfigurado.
Área de Pesquisa	• apresenta uma área exclusiva para procura de chamados; • a pesquisa foi colocada em um local visível e fácil utilização; • ao lado da caixa de pesquisa rápida existe um link para uma pesquisa avançada.	• apresenta uma caixa de pesquisa “perdida” entre as demais informações; • não apresenta um link de busca avançada.
Breadcrumbs	• localizado acima do título da página, apresenta o trajeto percorrido pelo usuário até a página atual; • o nível de hierarquia “Chamados” é etiquetado com um link de acesso para a página e o local atual é identificado através da cor cinza ; • o usuário sente-se mais informado e localizado dentro do sistema.	• não existe nenhum tipo de localização da área em que o usuário esta.

Quadro 6 – Melhorias apresentadas na tela de meus chamados

3.3.4 Tela de Abertura de Chamados

O solicitante entrou na tela que estava procurando. A partir dela ele pode abrir o chamado de seu software que está com problemas para a área responsável. Como o problema é no software, ele deverá escolher os campos conforme a figura 30 e enviar o formulário.

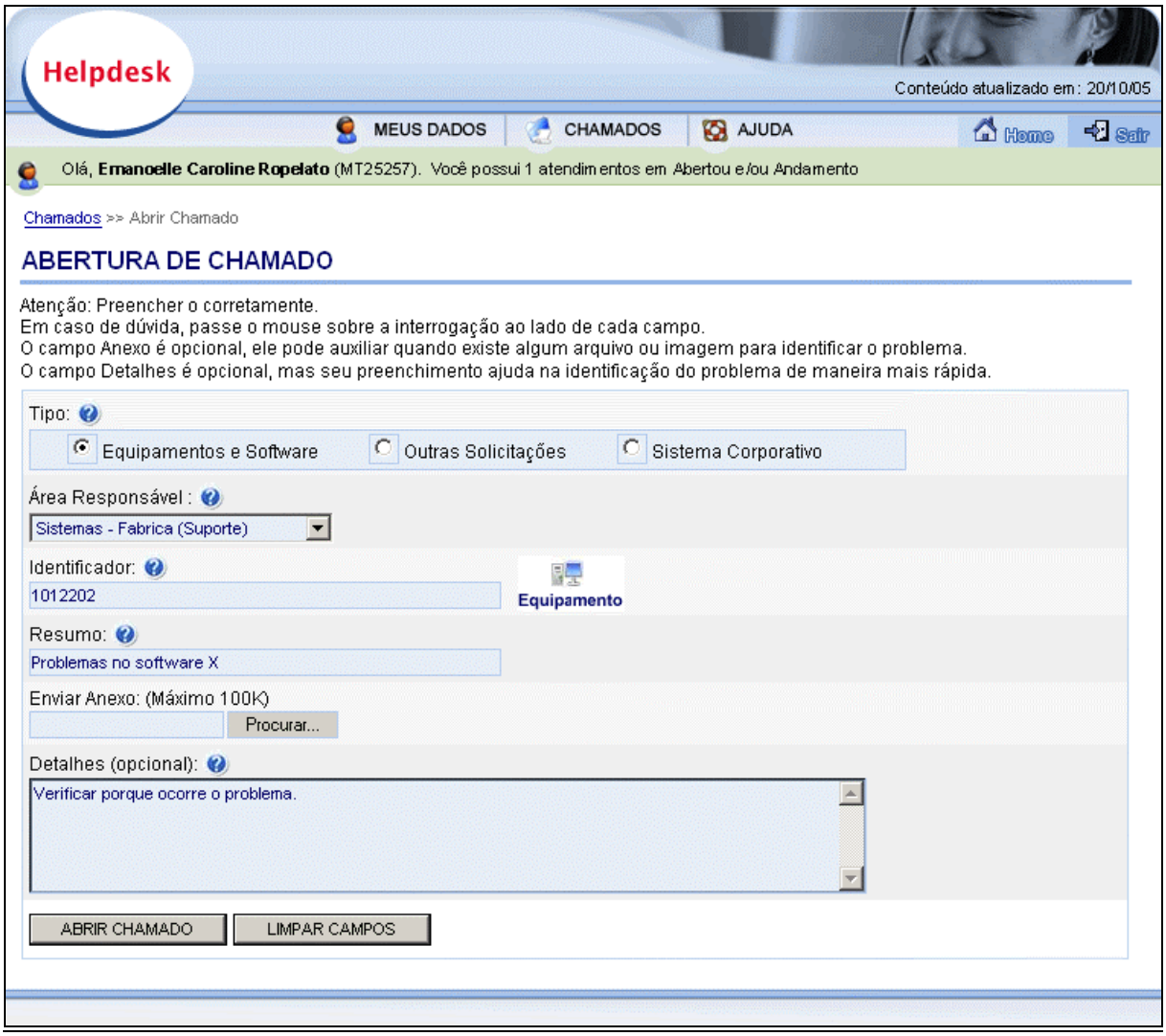


Figura 30 – Tela de Abertura de chamado

Quando o usuário tem problemas referentes ao equipamento ou seus softwares o identificador deve sempre ser o número do equipamento. Caso tenha problemas no sistema da empresa, o identificador será o número da tela que ocorreu o problema, se não se encaixa nestes o identificador será ele mesmo. Qualquer informação incorreta será apresentado o erro da mesma maneira que na tela de *login*.

A página segue a mesma padronização da anterior com as três divisões e foram aplicados os *Web Design Patterns* de formulário, mensagem de erro em formulários e *Breadcrumbs*, conforme quadros 7 e 8.

Padrão	Aplicação atual	Aplicação antiga
Formulário	• texto para utilização do formulário esta mais explicativo; • criado um texto sobre a importância do preenchimento dos campos opcionais; • o help disposto ao lado de cada item estão mais claros; • criados campos pré-definidos para prevenir erros durante a digitação; • acrescentado o botão “Limpar” que resseta o formulário para as informações iniciais; • a apresentação do formulário esta mais claro e prático para o preenchimento das informações.	• texto para utilização do formulário esta confuso; • não existe uma indicação de importância do preenchimento dos campos opcionais e por experiência o mesmo é pouco utilizado e muito necessário; • não apresenta um botão para limpar as informações digitadas;
Mensagem de erro	• todas as mensagens aparecem abaixo do título da página; • os retornos de erro aparecem somente após a submissão do formulário; • quando ocorre um retorno de erro, o foco do formulário vai para o campo que ocasionou o problema e sua tag muda a cor para vermelho; • as mensagens estão padronizadas e a identificação do problema esta mais simples.	• as mensagens aparecem como texto ou em caixas de alertas; • existem retornos antes de submeter o formulário e depois de submetido; • não caracteriza o campo que ocorreu o problema, mas retorna o foco para o mesmo.

Quadro 7 – Melhorias apresentadas na tela de abertura de chamados

Padrão	Aplicação atual	Aplicação antiga
Breadcrumbs	• localizado acima do título da página, apresenta o trajeto percorrido pelo usuário até a página atual; • o nível de hierarquia “Chamados” é etiquetado com um link de acesso para a página e o local atual é identificado através da cor cinza ; • o usuário sente-se mais informado e localizado dentro do sistema.	• não existe nenhum tipo de localização da área em que o usuário esta.

Quadro 8 – Melhorias apresentadas na tela de abertura de chamados - Continuação

A aplicação anterior apresentava o mesmo problema que a tela de *login*, erros em caixas de alertas e no código HTML. A atual foi corrigida e colocada no padrão de desenvolvimento. O formulário também ficou mais claro após a aplicação do *pattern* de formulários, conforme a figura 31.

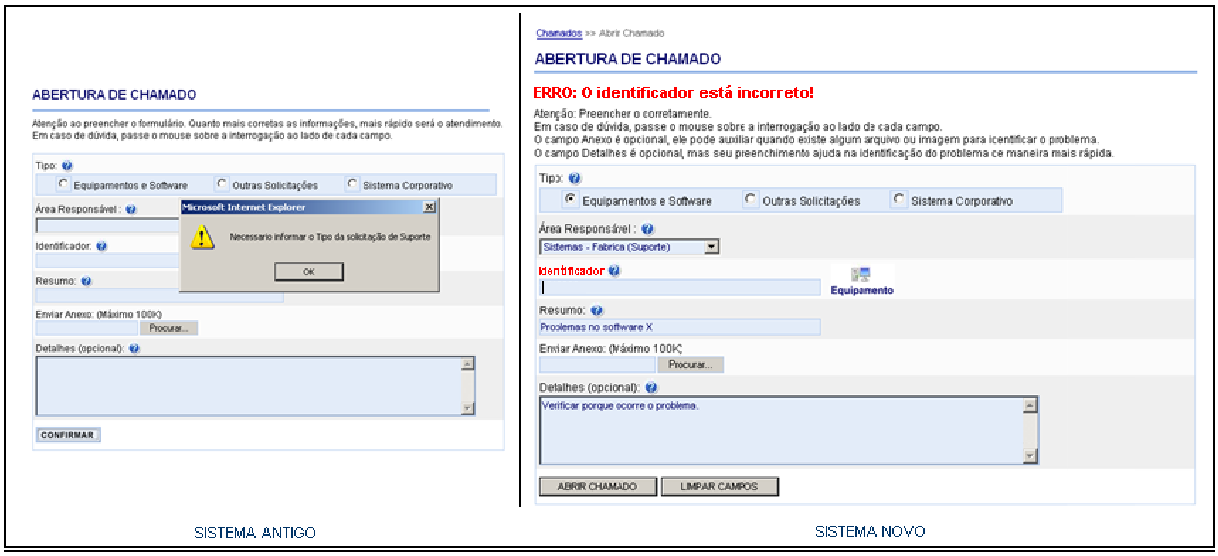


Figura 31 – Comparação da tela de abertura antiga com atual

3.3.5 Tela de Resposta

O solicitante abriu o chamado corretamente. O sistema apresenta a tela de resposta com uma mensagem de criação de chamado, encaminha um e-mail para o administrador da equipe responsável, um para o solicitante e o remete para a tela de Meus Chamados, conforme mostra a figura 32.

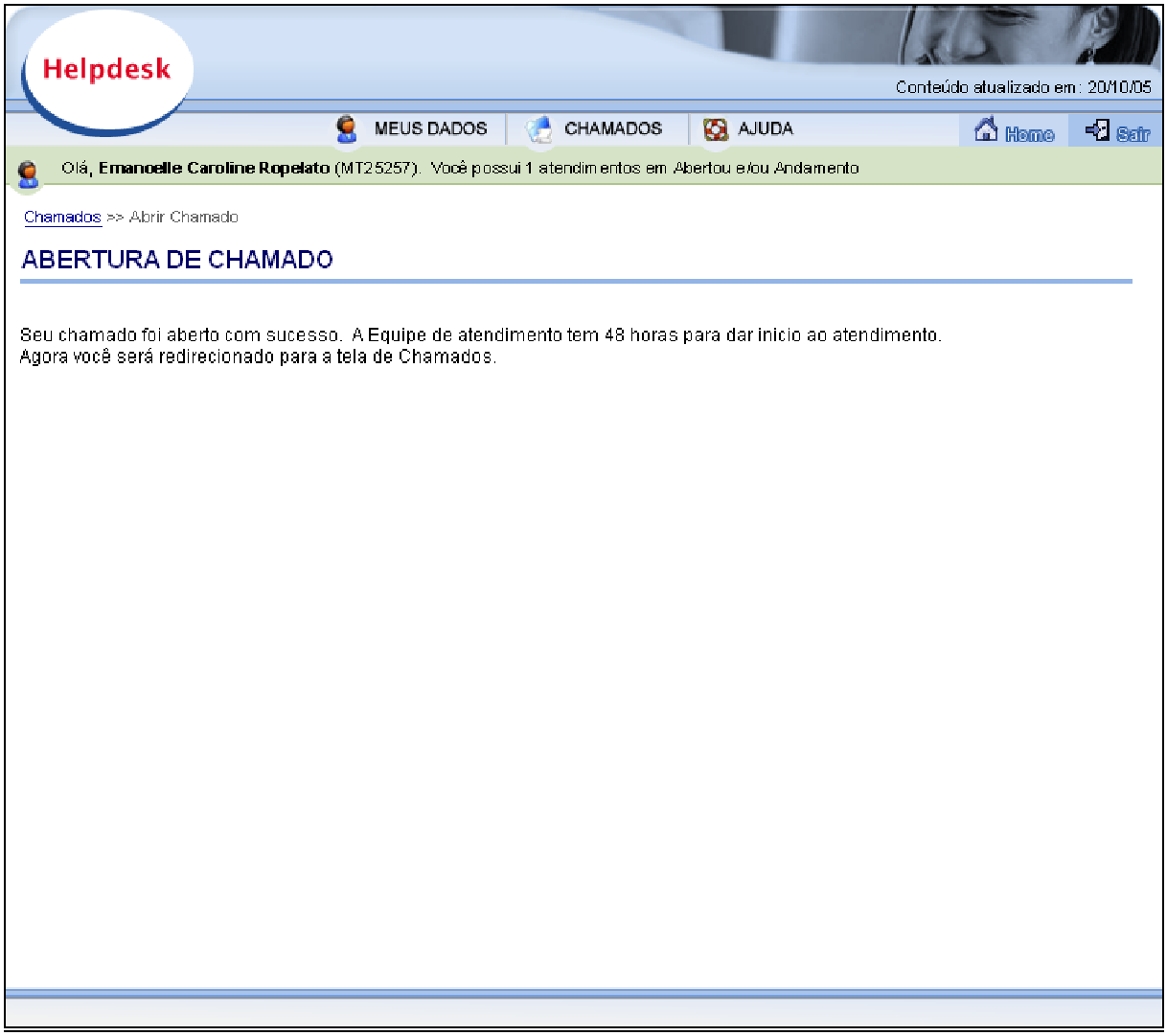


Figura 32 – Tela de resposta

Anteriormente não existia uma tela de resposta. Era apenas uma caixa de alerta, como mostra a figura 33, o que poderia causar problemas, dependendo do navegador e sua configuração. Na aplicação atual foram acrescentados *patterns* conforme quadro 9.

Padrão	Aplicação atual	Aplicação antiga
Mensagem de erro	• mostra a mensagem diretamente na página; • as intenções da página estão melhor representadas.	• texto para utilização do formulário esta confuso; • não existe uma indicação de importância do preenchimento dos campos opcionais e por experiência o mesmo é pouco utilizado e muito necessário; • não apresenta um botão para limpar as informações digitadas;
Breadcrumbs	• localizado acima do título da página, apresenta o trajeto percorrido pelo usuário até a página atual; • o nível de hierarquia “Chamados” é etiquetado com um link de acesso para a página e o local atual é identificado através da cor cinza ; • o usuário sente-se mais informado e localizado dentro do sistema.	• não existe nenhum conteúdo no momento de confirmação do chamado.

Quadro 9 – Melhorias apresentadas na tela de resposta

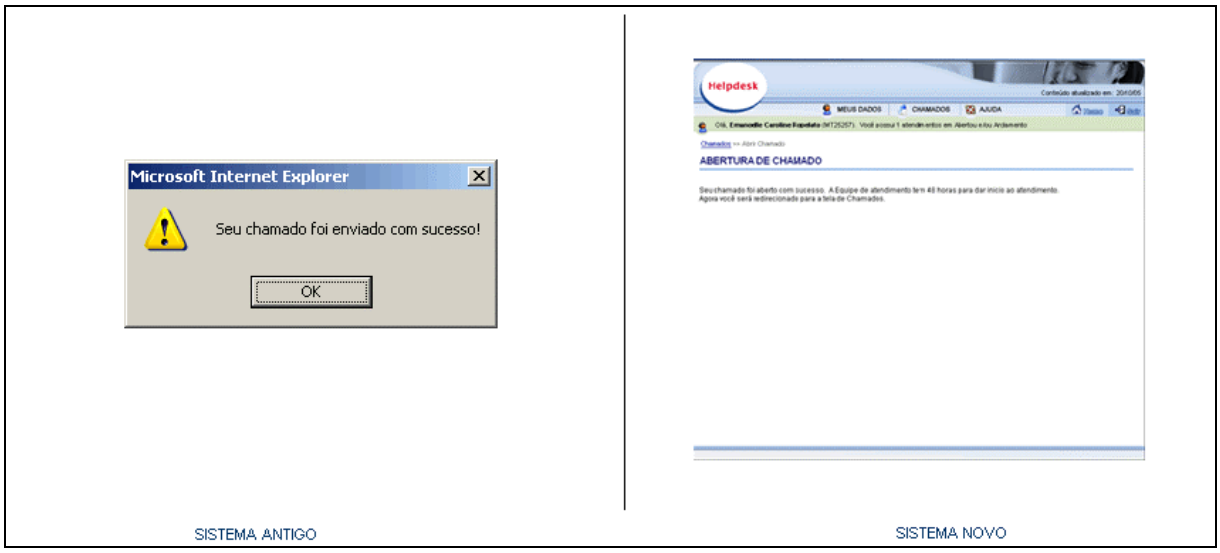


Figura 33 – Comparação da tela de Resposta antiga e atual

3.4 RESULTADOS E DISCUSSÃO

A aplicação desenvolvida neste trabalho trata-se de um sistema *HelpDesk* simples para a Web. O que tornou interessante foi o estudo e a implementação de padrões de desenvolvimento Web, tanto na análise, utilizando a engenharia Web como na implementação utilizando os padrões de projeto.

Trabalhos anteriores como de Sorroche e Lopes (2003), foram implementadas ferramentas utilizando *design patterns* especificamente para sistema desenvolvido em J2EE que por sinal é uma área de amplo estudo. Grott (2003) desenvolveu um software desktop com utilização de *design patterns* e por fim Almeida (2004) apresentou um sistema Web em PHP baseado em padrões de projetos, mais abrangentes e não os específicos para programação PHP como foram implementados neste trabalho.

O que pode ser notado, tanto neste trabalho como nos trabalhos correlatos é que a aplicação dos *patterns* sejam genéricos ou específicos como os *Web Design Patterns* mostraram o quanto é importante trabalhar com esses padrões reforçando a idéia de reuso de um bom código.

Neste trabalho, a estratégia foi trabalhar em uma aplicação já existente, com um código em PHP estruturado e desenvolvido sem nenhuma base de padrões ou mesmo engenharia Web. Conforme se iniciou os trabalhos a diferença entre utilizar um código bem modelado e seguindo padrões e um desenvolvimento sem planejamento era enorme. Enquanto na nova

aplicação via-se o desenvolvimento fluir adequadamente com a produtividade era possível notar que a outra aplicação estava totalmente amarrada. Outro aspecto foi à utilização da avaliação de heurística para Web, demonstrando a diferença entre a ferramenta obedecendo a heurística e a que esta totalmente fora do padrão, conforme os quadros 10 e 11.

Heurística	**Nova Aplicação**	**Aplicação Antiga**
Status do sistema	Ao visualizar um chamado o usuário é informado em qual aba de detalhe o mesmo se encontra.	Não havia identificação de qual detalhe estava sendo visualizado, o que prejudicava o entendimento das informações.
Compatibilidade do sistema com o mundo real	Os termos utilizados para identificar as páginas, elementos entre outros foram condizentes com o que os usuários estão acostumados. Um exemplo: o termo “Minha responsabilidade”, qualquer atendente já sabe que são os chamados que estão sob seu atendimento.	A aplicação estava condizente com este item, exceto pelas mensagens de erro que apresentava para os solicitantes.
Controle do usuário e liberdade	O usuário pode configurar como deseja ver a lista de chamados, quais tipos de chamados devem aparecer e podem utilizar qualquer botão do navegador que estão utilizando, uma vez o que sistema não desabilita nenhuma funcionalidade da ferramenta.	O usuário já tinha controle total sobre a aplicação, podendo configurar da mesma maneira que a aplicação atual.
Consistência e padrões	Foi criado um padrão de hierarquia da informação, padronizados o esquema de cores (vermelho para mensagem de erro, azul para *links*, etc.), o cabeçalho, os botões e os *links* utilizados em todas as páginas, a linguagem e o formato das mensagens de erro, facilitando o aprendizado da ferramenta.	Apesar de parecer, não havia coerência entre os tipos de elementos. Em alguns lugares os *links* estão sublinhados ou em outra cor. Os botões de formulário estão diferentes e as mensagens de erro as vezes aparecem na página outras vezes em caixas de alerta.
Prevenção de erros	Através dos *WebDesign Patterns* de formulários e mensagem de erro foram corrigidos problemas com *input* de informações incorretas. Foram aplicadas máscaras em campos pré-definidos e ajudas para que o usuário consiga entender o funcionamento da aplicação.	Não havia um forte controle de prevenção de erro, o que prejudicava um pouco o *input* das informações.

Quadro 10 – Comparação de heurística entre as ferramentas

Heurística	Nova Aplicação	Aplicação Antiga
Reconhecimento ao invés de lembrança	A aplicação permite que o usuário acesse facilmente informações a qualquer momento, já que ele sempre esta informado sobre sua localização dentro da aplicação, através do *patterns* de *Breadcrumbs*.	Não possui um caminho identificando para onde o usuário deve voltar para ver determinada informação.
Flexibilidade e eficiência no uso	Como é uma aplicação para empresa, a flexibilidade esta no modo que o usuário pode apresentar sua lista de chamados.	Segue o mesmo padrão da atual, uma vez que já apresentava características de personalização.
Estética e design minimalista	Procurou-se aplicar poucos efeitos visuais e manter uma estrutura *clean* e que não cansasse a visão do usuário, uma vez que deve trabalhar quase todo o expediente diante da ferramenta.	Apresenta alguns "enfeites" que cansam a visão do usuário, como botões feitos a partir de imagens ao invés de utilizar os botões de formulários.
Ajudar os usuários a reconhecer, diagnosticar e corrigir erros	Utilizando o *pattern* de mensagem de erro, a linguagem sistema X usuário ficou mais clara o que possibilitou uma posição mais correta do usuário durante o *input* de informações.	As mensagens estavam confusas, aumentando as dúvidas entre os usuários do sistema. Muitos não conseguiam abrir um chamado sozinho e entravam em contato via telefone com os atendentes.
Ajuda e documentação	A área de FAQ foi totalmente reformulada, visando atender as necessidades dos usuários. Existe um *help* a disposição do usuário.	O FAQ era ultrapassado, mas existia um *help* para consulta dos usuários.

Quadro 11 – Comparação de heurística entre as ferramentas

Abaixo se encontram exemplos da utilização de heurística no *HelpDesk*.

A aplicação da heurística deu-se juntamente com a aplicação dos Web *Design patterns*. Nesta área serão apresentados exemplos de utilização desta análise na ferramenta *HelpDesk*.

A figura 34 mostra a área mais escura como a área atual em que o usuário esta localizado, as demais podem ser acessada a qualquer momento.

Figura 34 – Heurística *status* do sistema

Para a heurística de compatibilidade do sistema com o mundo real procurou-se utilizar termos já utilizados entre os colaboradores da empresa mesmo antes de existir a aplicação. Portanto, os itens do menu foram baseados nas expressões do dia-a-dia da empresa, conforme mostra a figura 35.

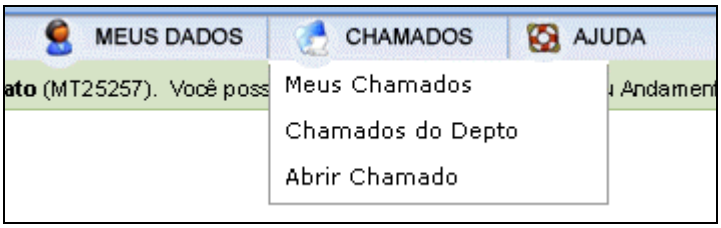


Figura 35 – Heurística compatibilidade do sistema

Outra preocupação foi criar uma padronização dos elementos da aplicação apresentando uma consistência e padrões. A figura 36 mostra alguns dos padrões criados.

Figura 37 – Heurística consistência e padrões

Na aplicação anterior ocorriam muitos problemas no cadastramento das informações e abertura de chamados, para que não ocorra mais foram padronizados alguns campos, conforme figura 37. Neste exemplo contempla-se a heurística de prevenção de erros.

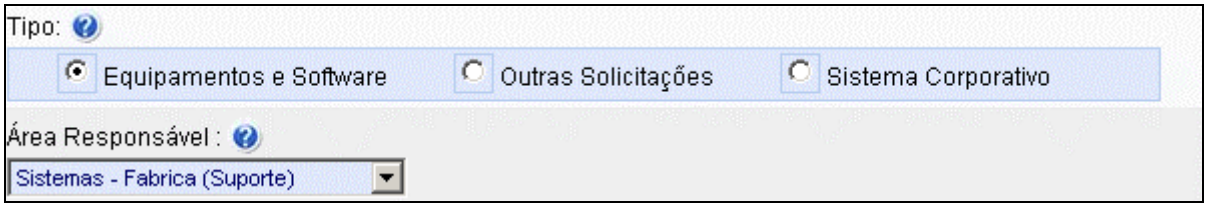


Figura 37 – Heurística prevenção de erros

O reconhecimento ao invés de lembrança esta aplicado na estrutura do novo sistema, já que o mesmo apresenta ao usuário sua localização atual no sistema e como voltar para informações anteriores, como mostra a figura 38.

Figura 38 – Heurística reconhecimento ao invés de lembrança

O usuário pode personalizar a página de Chamados conforme sua necessidade. Com isto, foi contemplada a área de flexibilidade e eficiência no uso, como apresenta a figura 39.

Figura 39 – Heurística flexibilidade e eficiência no uso

Outra solicitação da heurística, ajudar os usuários a reconhecer, diagnosticar e corrigir

erros foi atendida com as novas mensagens configuradas para o contato sistema X usuário. Uma demonstração desta aplicação pode ser vista na figura 40.

Figura 40 – Heurística ajudar os usuários a reconhecer

É muito importante salientar que o leiaute e a interface sofreram pequenas alterações. Como este sistema já está implantado para alguns usuários, a nova aplicação foi desenvolvida para não causar um grande impacto visual para o usuário, mas sim um impacto funcional. A grande mudança das interfaces foi a linha de aprendizagem, aonde se definiu as fontes que representariam os links, os textos, os destaques e ainda os tipos de botões, elementos de formulário, etc.

A reconstrução da ferramenta foi um sucesso, concretizando mais uma vez o resultado visto nos trabalhos correlatos e que trabalhar com um paradigma consistente e bem definido seguindo as metodologias de Padrões de Projeto e Engenharia Web, é possível desenvolver uma aplicação Web mais consistente e menos sujeito à falhas.

Em relação aos usuários que fizeram os testes, a grande maioria preferiu utilizar a nova ferramenta. Todos sentiram um ganho de performance muito grande durante as interações e ficaram mais familiarizamos com a nova interface, considerando a mesma mais intuitiva e prática para utilizar.

Com isto, pode-se concluir que a aplicação de *patterns* não é apenas proveitosa para os desenvolvedores, como também enchem os olhos dos usuários, o que torna o sistema muito mais utilizado e pode-se sentir um *feedback* por parte dos usuários.

4 CONCLUSÕES

A utilização de sistemas de *HelpDesk* tornou-se fundamental para as organizações gerenciarem chamados abertos pelos seus usuários, armazená-los documentá-los. Com isto foi implementada uma primeira versão deste sistema, apenas para nutrir a necessidade de um setor. Mas a aplicação foi tomando uma dimensão gigantesca e a estrutura já não suportava mais alterações.

Pensando nisto este trabalho foi posto em prática. Reconstruir a *WebApp* , mas agora de maneira correta, utilizando tecnologias que possibilitem uma boa estruturação tanto de código como de análise. Antes de iniciar qualquer desenvolvimento foram feitas análises que atendesse os requisitos que Pressman (2006) descreve em seus estudos. A seqüência desta análise foi similar à sugerida pela pirâmide de projetos Web. Depois de realizadas as análises trabalharam-se os padrões de projeto. Muita pesquisa e estudos foram feitos até encontrar os padrões que se adaptavam a aplicação e que atendiam aos requisitos.

A estrutura de base de dados também sofreu alteração, passando do Sistema de Gestão de Bases de Dados (SGBD) Oracle para MySQL, o que a tornou mais versátil, podendo ser instalada em qualquer servidor que tenha estes bancos.

Em relação aos objetivos definidos neste trabalho, chega-se à conclusão de que os mesmos foram alcançados, pois a aplicação mostrou um bom resultado após a aplicação dos estudos sugeridos. O objetivo da ferramenta não foi alterado, mas sua estrutura de programação esta totalmente renovada, com uma interface mais dinâmica, prática e objetiva.

Os benefícios da utilização de padrões observou-se em dois momentos distintos, no desenvolvimento do código fonte, quando poderia trabalhar *design* e programação separadamente e de uma maneira correta e no momento da avaliação da nova ferramenta com os usuários finais, sendo relatado os ganhos em usabilidade e performance da nova aplicação em relação a antiga. A aplicação da heurística foi quase uma conseqüência da utilização dos Web *Design patterns*, uma vez que os dois tem a mesma preocupação, validar os conteúdos divulgados na internet.

Os *WebPatterns* foram utilizados de maneira correta, focando muito no padrão funcional, voltado tanto para a iteração do usuário com o sistema como as iterações entre as classes da aplicação, proporcionando um ambiente de trabalho mais otimizado. O sistema ganhou em desempenho e confiabilidade, só que o grande ganho da utilização de *patterns* foi diretamente com o usuário, que viu sua ferramenta de trabalho ficar mais agradável e

intuitiva, trazendo todas as características boas que formam uma aplicação de ambiente Web.

Cada *pattern* utilizado no desenvolvimento da *WebApp* foi documentado e listado para que outros estudantes possam dar continuidade ao estudo e aplicações dos mesmos.

O que faltou foi um estudo mais aprofundado de outros *WebPatterns* que poderiam ter sido utilizados. Mesmo assim pode-se notar a partir das comparações apresentadas que as aplicações apresentam grandes diferenças e que sempre é bom fazer uma análise antes de iniciar uma programação, o que não é freqüente na Web.

Quanto às ferramentas e tecnologias utilizadas, muitas delas tiveram que ser estudadas e pesquisadas durante este trabalho, sendo que as mesmas auxiliaram e facilitaram bastante o desenvolvimento.

Algumas dificuldades e limitações foram encontradas no decorrer do desenvolvimento, uma delas é o sistema de abertura de chamados. Quem controla a abertura é um software que faz a sincronia entre Web e sistema. Este software não rodou na máquina de testes e como ele e proprietário não foi possível utilizá-lo.

Ao final deste trabalho concluiu-se que mesmo sendo uma *WebApp* é necessário aplicar os padrões de desenvolvimento e uma engenharia Web, para desenvolver softwares cada vez mais confiáveis e com estruturas que permitam a qualquer programador fazer uma manutenção sem perder boa parte do tempo. Outra coisa é pensar sempre no usuário final, que através destas práticas irá apreciar muito mais a aplicação que fará parte de seu dia-a-dia.

4.1 EXTENSÕES

Como sugestões para possíveis extensões ao trabalho desenvolvido citam-se:

a) estudar e aplicar outros *patterns* que não puderam ser aplicados agora e identificar as melhorias ocorridas;
b) acrescentar outras funcionalidades que existe no software desktop, inventário de máquinas e softwares, cadastro de máquinas, entre outros;
c) gerar relatórios de tempo de atendimento por equipe, total de chamados atendidos, equipamentos com problemas, entre outros, para auxiliar nas estatísticas e no gerenciamento dos chamados.

REFERÊNCIAS BIBLIOGRÁFICAS

ALECRIM, E. **Banco de dados MySQL e PostgreSQL**. [S.l.], 2007. Disponível em: < http://www.infowester.com/postgremysql.php>. Acesso em: 05 maio 2007.

ALLSOPP, J. **OnVoiceOver**. [S.l.], 2005. Disponível em: <http://www.onvoiceover.com/articles/webpatterns/>. Acesso em: 15 abr. 2007.

ALMEIDA, R. I. M. **Utilização de padrões de projeto no desenvolvimento de aplicações Web com PHP 5**. 2004. 41 f. Monografia (Bacharelado em Sistemas de Informação) - Centro Universitário Luterano de Palmas, Palmas. Disponível em: <http://www.ulbra-to.br/ensino/43020/artigos/relatorios2004-2/Arquivos/RicardoIshibashi_Estagio.pdf>. Acesso em: 20 abr. 2007.

AMSTEL, F. **Diagrama de navegação**. [S.l.], 2007. Disponível em: < http://www.usabilidoido.com.br/imagens/diagrama_navegacao.png>. Acesso em: 10 maio 2007.

BOLCHINI, D. **Web design patterns**: improving quality and performance in Web application design. Lugano: [s.n.], 2000

CONALLEN, J. **Desenvolvendo aplicações Web com UML**. 2. ed. Tradução Altair Dias Caldas de Moraes, Cláudio Belleza Dias. Rio de Janeiro: Campus, 2003.

GROTT, M.C. **Reutilização de soluções com patterns e frameworks na camada de negócio**. 2003. 114 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

GONÇALVES, R. F. et al. Uma proposta de processo de produção de aplicações Web. **Revista Produção**, [S.l.], v. 15, n. 3, p. 376-389, Set./Dez. 2005.

LOZANO, F. **Patterns e anti-patterns para PHP**. [S.l.], 2003. Disponível em: <http://www.lozano.eti.br/palestras/patters-php.pdf >. Acesso em: 01 maio 2007.

MACIEL, C. et al. **Avaliação Heurística de Sítios na Web**. In: VII ESCOLA DE INFORMÁTICA DO SBC - CENTROOESTE, 2004, Cuiabá. SUCESU-MT 2004 Conference: Sociedade do Conhecimento. Cuiabá: PAK Multimídia, 2004.

MACROMEDIA DreamWeaver. In: ADOBE, Resumo do DreamWeaver. [S.l.]: Adobe, [2007]. Disponível em: < http://www.adobe.com/br/products/dreamweaver/>. Acesso em: 20 maio 2007.

NIELSEN, J. **Designing Web usability**. [S.l.], 1994. Disponível em: < http://www.asktog.com/basics/firstPrinciples.html>. Acesso em: 8 junho 2007.

OLIVEIRA, L. **Wireframe, documento cada vez mais importante**. [S.l.], 2003. Disponível em: < http://webinsider.uol.com.br/index.php/2003/12/09/wireframe-documento-cada-vez-mais-importante/>. Acesso em: 05 maio 2007.

PRESSMAN, R. S. **Engenharia de software**. 6. ed. Tradução Rosângela Delloso Penteado. São Paulo: McGraw-Hill, 2006.

SAUVÉ, J.P. **Jacques philippe sauvé's home at UFCG**. [S.l.], 2006. Disponível em: < http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/arqu/mvc/_mvc.gif >. Acesso em: 01 maio 2007.

SORROCHE, R.; LOPES, M. C. Uso de design patterns e J2EE: um estudo de caso. In: SEMINÁRIO DE COMPUTAÇÃO, 12., 2003, Blumenau. **Anais eletrônicos**... Blumenau: FURB, 2003. Não Paginado. Disponível em: <http://www.inf.furb.br/seminco/2003/artigos/128-vf.pdf>. Acesso em: 28 abr. 2007.

XEXÉO, G. **Modelagem de Sistemas de Informação**. [S.l.], 2007. Disponível em: < http://ge.cos.ufrj.br/tikiwiki/tiki-download_file.php?fileId=1>. Acesso em: 07 maio 2007.