UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

SISTEMA DE GERENCIAMENTO E ACOMPANHAMENTO DE PROPOSTAS E REQUISITOS

RODRIGO FELIPE MORITZ PETTERS

BLUMENAU
2007

2007/1-18

RODRIGO FELIPE MORITZ PETTERS

SISTEMA DE GERENCIAMENTO E ACOMPANHAMENTO DE PROPOSTAS E REQUISITOS

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 Sistemas de Informação - Bacharelado.

Prof. Wilson Pedro Carli, Mestre - Orientador

BLUMENAU
2007

2007/1-18

SISTEMA DE GERENCIAMENTO E ACOMPANHAMENTO DE PROPOSTAS E REQUISITOS

Por

RODRIGO FELIPE MORITZ PETTERS

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

Presidente:	Prof. Wilson Pedro Carli, Mestre – Orientador, FURB
Membro:	Prof. Adilson Vahldick, Especialista – FURB
Membro:	Prof. Paulo Roberto Dias, Mestre – FURB

Blumenau, 04 de julho de 2007

Dedico este trabalho à minha namorada e guia espiritual por estar sempre ao meu lado.

AGRADECIMENTOS

Ao Grande Mistério, por ser imenso amor e luz.

À minha família, que mesmo longe, sempre esteve presente. Especialmente meu pai que acreditou em mim e na conclusão do curso.

Aos meus colegas da Totall.com S.A., pelo auxílio que recebi para a realização deste trabalho.

Ao meu orientador, Wilson Pedro Carli, por ter me aceito como orientado e acreditado na conclusão deste trabalho.

Obrigado!

A linguagem é o único instrumento da ciência.

Samuel Johnson

RESUMO

Este trabalho apresenta a especificação e desenvolvimento de uma aplicação web para gerenciamento e acompanhamento de propostas de desenvolvimento e manutenção de softwares, possibilitando a interação entre cliente e analista. Demonstra-se também como é possível aos clientes, utilizando técnicas e tecnologias como AJAX e Flash através de um sistema desenvolvido em PHP, incluírem e acompanharem propostas para obterem resultado com maior autonomia e precisão.

Palavras-chave: Requisitos. Casos de Uso. Web 2.0. AJAX. PostgreSQL.

ABSTRACT

This work presents the development and specification of a web application for follow and management the development and maintenance proposals of softwares, allowing interaction between client and analyst. It is also demonstrated how it’s possible to the clients, using techniques like AJAX and Flash through a system developed on PHP, include and follow proposals to obtain results with more autonomy and precision.

Key-words: Requirement. Use Cases. Web 2.0. AJAX. PostgreSQL.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

AJAX – *Asyncronous JavaScript and XML*

ANSI – *American National Standards Institute*

API – *Application Programming Interface*

BSD – *Berkeley Software Distribution*

CASE – *Computer Aided Systems Engineering*

CGI – *Common Gateway Interface*

CSS – *Cascading Style Sheets*

DAO – *Data Access Object*

DBA – *Data Base Administrator*

DHTML – *Dynamic Hypertext Markup Language*

DOM – *Document Object Model*

EA – *Enterprise Architect*

GNU – *General Public License*

HTML – *HyperText Markup Language*

HTTP – *Hypertext Transfer Protocol*

IDE – *Integrated Development Environment*

MER – Modelo Entidade-Relacionamento

MVCC – *Multi-Version Concurrency Control*

PHP – *Personal Hypertext Preprocessor*

PHP/FI – *PHP/Forms Interpreter*

SGBD – Sistema Gerenciador de Banco de Dados

SQL – *Structured Query Language*

SSL – *Secure Sockets Layer*

SVG – *Scalable Vector Graphics*

UML – *Unified Modeling Language*

XHTML – *eXtensible Hypertext Markup Language*

XML – *eXtensible Markup Language*

XHR – *XMLHttpRequest*

XSLT – *eXtensible Stylesheet Language Transformations*

WAMP5 – *Windows Apache MySQL PHP 5*

WWW – *World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

Alinhar o que é solicitado pelo cliente com o que lhe é entregue tem sido o desafio de empresas de vários segmentos, principalmente o de desenvolvimento de softwares. É muito comum encontrar uma desordem entre o que é pedido pelo cliente e o que é realizado pela empresa, principalmente, devido ao grau de complexidade e precisão que os pedidos precisam ter.

Outro problema que está atrelado a este é a questão do tempo. Mais do que receber o que quer, o cliente precisa saber quando vai receber. Desta forma, é possível prever melhor os custos com implantação, treinamento e outros. Tal quadro foi observado em uma empresa de informática de pequeno porte em Blumenau no estado de Santa Catarina. Na mesma, através de entrevistas com especialistas na área de desenvolvimento, observou-se que entre as solicitações dos clientes e a definição das tarefas, existem algumas falhas de comunicação e de informações que prejudicam o andamento dos trabalhos, resultando em desgaste nas relações entre os clientes e a empresa.

Contudo, essas considerações a respeito de novos conceitos organizacionais, novas formas de relacionamentos com clientes e fornecedores e novas vitrines de interação com o mercado têm um ponto comum: a Internet (BARBIERI, 2001).

E é através da Internet que as empresas vêm estreitando laços com seus clientes. É muito comum que uma empresa hoje em dia tenha um web site e que nele exista uma área de relacionamento entre empresa e cliente. Este relacionamento torna-se mais estreito, à medida que a tecnologia se torna mais flexível e dinâmica.

Dentro das tecnologias existentes, uma linguagem que possui uma ampla difusão no desenvolvimento de sites, e que além de gratuita possui uma gama de documentações, é o *Personal Hypertext Preprocessor* (PHP). Esta, aliada a um conjunto de tecnologias que se

destaca na Web 2.0[1], que é o *Asyncronous JavaScript and eXtensible Markup Language* (AJAX), que por sua vez nada mais é que uma forma mais ágil de trocar informações entre o servidor do web site e o browser do cliente. Esta agilidade permite que as aplicações sejam mais flexíveis e dinâmicas.

Baseando-se nesses fatos desenvolveu-se uma aplicação web em PHP, que utiliza interface AJAX e que faz o intermediário entre o Analista de Requisitos – representando a empresa – e o cliente. A tarefa principal desta aplicação é fazer com que clientes possam incluir e acompanhar propostas e que estas possam ser avaliadas pelos analistas. Com isso, tenciona-se que as execuções estejam em consenso e que exista máxima aproximação entre os resultados esperados e os obtidos, protegendo assim ambas as partes.

1.1 OBJETIVOS DO TRABALHO

O objetivo geral do presente trabalho é desenvolver uma aplicação web em PHP para ser mediadora entre o Analista de Requisitos e o cliente, permitindo assim, que ambos interajam e aprovem cada passo do projeto através de requisitos e diagramas de casos de uso.

Os objetivos específicos do trabalho são:

a) implementar um aplicativo que faça a interação do analista de requisitos e o cliente em relação as propostas de desenvolvimento;
b) permitir que o analista relacione os requisitos e desenhe diagramas de casos de uso na proposta;
c) interagir com o cliente quanto a aprovação a cada alteração nos requisitos ou diagramas;
d) permitir que ambos possam acompanhar o andamento da proposta.

[1] Representação utilizada para designar a segunda geração da *World Wide Web* (WWW).

1.2 ESTRUTURA DO TRABALHO

No capítulo 1, apresenta-se a introdução do trabalho e os objetivos a serem atingidos.

No capítulo 2, apresenta-se definições da linguagem PHP, apresentação da Web 2.0, detalhes da técnica AJAX, do banco de dados PostgreSQL, do processo de Análise de Requisitos, de como é o processo de solicitação de requisitos atualmente e trabalhos correlatos.

No capítulo 3, apresenta-se a especificação da aplicação web e detalhes de seu desenvolvimento e funcionamento.

No capítulo 4 são apresentadas as conclusões do trabalho e as sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Inicialmente conceitos técnicos sobre a linguagem PHP são apresentados, prosseguindo com a visão geral de AJAX e do conceito Web 2.0 e do banco de dados PostgreSQL. Também são apresentados conceitos de análise de requisitos, um estudo de como funciona o processo atual de solicitação de propostas e são descritos alguns trabalhos correlatos.

2.1 DEFINIÇÕES DA LINGUAGEM PHP

PHP é uma linguagem de script o*pen source* de uso geral, muito utilizada e especialmente guarnecida para o desenvolvimento de aplicações web embutível dentro do *HyperText Markup Language* (HTML) (PHP.NET, 2007).

A primeira versão da linguagem foi desenvolvida por Rasmus Lerdof em 1994 como um *Common Gateway Interface* (CGI) escrito na linguagem C que permitia a interpretação de um número limitado de comandos. Esse sistema foi denominado *Personal Home Page Tools*. Com seu relativo êxito, a linguagem evoluiu para, além de interpretar comandos, processar formulários. Nesta etapa, recebeu o nome *PHP/Forms Interpreter* (PHP/FI).

A seguinte grande contribuição à linguagem foi realizada a meados de 97 quando se voltou a programar o analisador sintático (CRIARWEB.COM, 2007). Isso se deu em parte com a ajuda de dois programadores israelitas pertencentes ao Technion, o Instituto Israelita de Tecnologia: Zeev Suraski e Andi Gutmans. Essa nova versão recebeu o nome de PHP3.

Ao reescrever o núcleo, foi criado o Zend Engine, que é mantido oficialmente pela empresa Zend em conjunto com a comunidade PHP. Em maio de 2000 veio a público a versão 4, e em julho de 2004, a versão 5, onde a principal mudança foi uma nova *Application Programming Interface* (API) para orientação a objetos provida pelo Zend Engine 2.

> “Trata-se de uma linguagem extremamente modularizada, o que a torna ideal para instalação e uso em servidores web. [...] É muito parecida, em tipos de dados, sintaxe e mesmo funções, com a linguagem C e com a C++. [...] Pode ser, dependendo da configuração do servidor, embutida no código HTML. Além disso, destaca-se a extrema facilidade com que PHP lida com servidores de base de dados, como MySQL, Firebird, PostgreSQL, Microsoft SQL Server e Oracle. Existem versões do PHP disponíveis para os seguintes sistemas operacionais: Windows, Linux, FreeBSD, Mac OS, OS/2, AS/400, Novell Netware, RISC OS, IRIX e Solaris”.
> (WIKIPEDIA, 2007a)

Segundo o PROJECTO SIEP (2004) uma das maiores vantagens que o PHP possui é o fato de fornecer um vasto leque de recursos para fazer acesso a mais de vinte servidores de bases de dados. Desta forma, não é preciso programar nenhuma classe ou função específica para lidar com a conexão com o banco de dados: essa classe já vem nativamente na linguagem na forma de extensão.

2.1.1 Novidades do PHP5

Segundo Gutmans, Bakken e Rethans (2005), somente o tempo irá dizer se a versão 5 do PHP terá tanto sucesso quanto os seus dois predecessores (PHP3 e PHP4). As novas funcionalidades e modificações objetivam livrar o PHP de quaisquer fraquezas que possa ter tido e garantir que ele permaneça na liderança como a melhor linguagem para elaboração de scripts do mundo.

Gutmans, Bakken e Rethans (2005), listam como principais novidades no PHP5 os seguintes itens:

a) novo modelo orientado a objetos:
   - implementação de interfaces,
   - operador *instaceof*, que permite comparação de um objeto a uma classe,
   - métodos e classes finais,
   - clonagem de objetos,
   - constantes de classes,

- métodos e membros estáticos,
- classes e métodos abstratos,
- indicação do tipo de classe,
- iteradores,
- função *_autoload()*, que permite centralizar as inclusões de arquivos necessárias em cada arquivo de script;

b) tratamento de exceções;

c) *foreach* com referências;

d) valores padrões para parâmetros por-referência;

e) novidades e aprimoramentos de extensão à outras linguagens e bibliotecas, como por exemplo *eXtensible Markup Language* (XML), *Document Object Model* (DOM) e Perl;

f) novo gerenciador de memória.

2.2 WEB 2.0

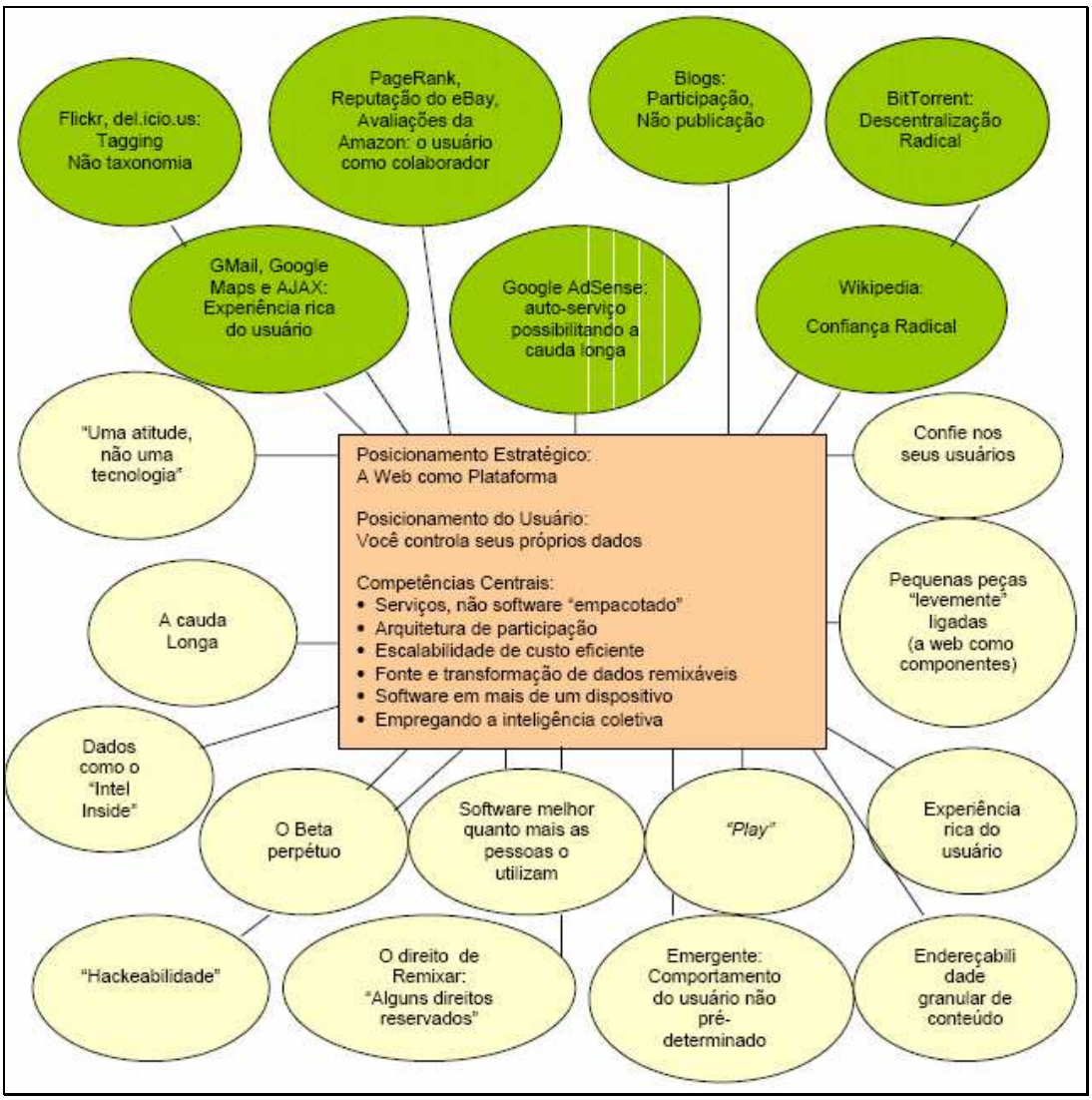
O termo Web 2.0 é utilizado para descrever a segunda geração da *World Wide Web* (WWW) - tendência que reforça o conceito de troca de informações e colaboração dos internautas com sites e serviços virtuais. A idéia é que o ambiente on-line se torne mais dinâmico e que os usuários colaborem para a organização de conteúdo (FOLHAONLINE, 2006).

> A Web 2.0 pressupõe um ambiente no qual muitos componentes se entrecruzam e se relacionam de uma nova forma. O primeiro deles é a busca por informações de maior relevância. Sites como o Google, Yahoo e outros são os maiores exemplos disto [...] A revolução aqui é obter resultados mais relevantes, a partir daquilo que realmente é buscado pelo usuário. Uma possibilidade é por meio de formas colaborativas nas quais os usuários fazem o ordenamento da informação (TIINSIDE, 2007).

Segundo a INFOEXAME (2007), 120 mil *blogs* surgem por dia - 1,4 por segundo. 1,5 milhões de *posts* são feitos todo dia nos *blogs* – 17 por segundo sendo que 35% dos *posts*

realizados em fevereiro do ano passado usaram *tags*. Mais ou menos 12% do tráfego na web americana é gerado por sites web 2.0 sendo que 7,1 milhões de artigos integram a Wikipedia.

Como muitos conceitos importantes, o de Web 2.0 não tem fronteiras rígidas, mas, pelo contrário, um centro gravitacional. Pode-se visualizar a Web 2.0 como um conjunto de princípios e práticas que interligam um verdadeiro sistema solar de sites que demonstram alguns ou todos esses princípios e que estão a distâncias variadas do centro, como observado na figura 1 (O'REILLY, 2005).



Fonte: O'REILLY, 2005.

Figura 1: Mapa de noções Web 2.0.

2.3 AJAX

AJAX é o uso sistemático de JavaScript e XML (e derivados) para tornar o navegador mais interativo com o usuário, utilizando-se de solicitações assíncronas de informações. AJAX não é somente um novo modelo, é também uma iniciativa na construção de aplicações web mais dinâmicas e criativas.

Desta forma, AJAX não é uma tecnologia, mas várias tecnologias trabalhando juntas, cada qual fazendo sua parte, oferecendo novas funcionalidades. A idéia é utilizar JavaScript para transformar as páginas em aplicações, de modo que não precise recarregar a tela cada vez que o usuário clicar em alguma coisa. Pode-se recarregar apenas a área que precisa ser alterada pela ação realizada (FERREIRA, 2007).

Segundo Garrett (2005), AJAX incorpora em seu modelo:

a) apresentação baseada em padrões, usando eX*tensible Hypertext Markup Language* XHTML e *Cascading Style Sheets* (CSS);

b) exposição e interação dinâmica usando o DOM;

c) intercâmbio e manipulação de dados usando XML e eX*tensible Stylesheet Language Transformations* (XSLT);

d) recuperação assíncrona de dados usando o objeto *XMLHttpRequest* (XHR);

e) JavaScript articulando todas elas.

Segundo Gois (2006), no AJAX, grande parte da codificação do aplicativo web fica no cliente, o que faz com que as aplicações se tornem mais responsivas e suportem uma maior variedade de formas de interação com o usuário como *drag-and-drop*, menus sensíveis ao contexto e atualizações dinâmicas de interfaces.

No modelo clássico de aplicação web, a interface com o usuário dispara uma solicitação *Hypertext Transfer Protocol* (HTTP) para o servidor web. O servidor processa os dados solicitados e então retorna uma página HTML para o cliente. A idéia na Web 2.0, com

o uso do AJAX, é dar maior transparência nas requisições cliente-servidor, dando maior agilidade para o usuário, para que ele não tenha que esperar cada requisição ao servidor. Para isso basta possuir algum dos navegadores modernos, ou seja, lançados após 2001.

2.3.1 Implementação do AJAX

Segundo Almeida (2006), o que difere a tecnologia AJAX da *Dynamic Hypertext Markup Language* (DHTML) é o uso do objeto XHR. Este objeto permite a um código JavaScript fazer o envio de dados e receber uma resposta de um servidor sem a necessidade de recarregar todo o código da página web. As versões anteriores ao Internet Explorer 7, que suportavam AJAX, permitiam primeiramente a implementação do ActiveX Microsoft.XMLHTTP e posteriormente do Msxml2.XMLHTTP.

Um exemplo de uma função JavaScript para criar um objeto XHR que pode ser usado na maior variedade possível de browsers pode ser observado na figura 2. Código-fonte retirado do desenvolvimento do trabalho.

```
 3  function XHR() {
 4      var ajax;
 5      try {
 6          ajax = new XMLHttpRequest();
 7      } catch(ee) {
 8          try {
 9              ajax = new ActiveXObject("Msxml2.XMLHTTP");
10          } catch(e) {
11              try {
12                  ajax = new ActiveXObject("Microsoft.XMLHTTP");
13              } catch(E) {
14                  ajax = false;
15              }
16          }
17      }
18      return ajax;
19  }
```

Figura 2: Implementação do AJAX.

Segundo W3C (2007), a classe XHR utilizada pelos browsers deve implementar a interface mostrada na figura 3, onde existem seis atributos – `onreadystatechange`, `readyState`, `responseText`, `responseXML`, `status` e `statusText` e seis métodos – `open`, `setRequestHeader`, `send`, `abort`, `getAllResponseHeaders` e `getResponseHeader`.

```
interface XMLHttpRequest {
           attribute EventListener   onreadystatechange;
  readonly attribute unsigned short  readyState;
  void               open(in DOMString method, in DOMString url);
  void               open(in DOMString method, in DOMString url, in boolean async);
  void               open(in DOMString method, in DOMString url, in boolean async, in DOMString user);
  void               open(in DOMString method, in DOMString url, in boolean async, in DOMString user, in DOMString password)
  void               setRequestHeader(in DOMString header, in DOMString value);
  void               send();
  void               send(in DOMString data);
  void               send(in Document data);
  void               abort();
  DOMString          getAllResponseHeaders();
  DOMString          getResponseHeader(in DOMString header);
  readonly attribute DOMString       responseText;
  readonly attribute Document        responseXML;
  readonly attribute unsigned short  status;
  readonly attribute DOMString       statusText;
};
```

Fonte: W3C, 2007.

Figura 3: Interface XHR.

Na figura 4, retirada do código-fonte do aplicativo desenvolvido neste trabalho, é possível observar a implementação de uma função que faz requisição assíncrona cliente-servidor, instanciando uma classe XHR. Nesta função o primeiro `if`, que está na linha 26, serve para verificar se o browser suporta DOM. Em seguida o objeto AJAX é instanciado a partir da classe XHR. O método open é chamado passando o método de envio de informações (“GET”, “POST”, “HEAD”, “PUT”, “DELETE”, “OPTIONS”), o link a ser submetido, e o tipo de requisição, informando se é assíncrona ou não.

O próximo passo é vincular uma função ao evento `onreadystatechange`, que é chamado a cada mudança de estado do atributo `readyState`. Neste evento está sendo verificado dois estados do objeto AJAX: 1 e 4. O estado 1 indica o início da transmissão, neste momento é mostrada para o usuário uma mensagem “Carregando...”. O estado 4 indica o término da transmissão. Caso o processo tenha tido sucesso - `status` igual a 200, o resultado da requisição do servidor é obtido através do atributo `responseText`.

Para que se inicie o processo é necessário acionar o método send. Se a requisição aberta no método open não for assíncrona, o método send só retorna após a transmissão de dados for concluída. Caso contrário o método send apenas dispara o processo.

```
25  function requisicaoXHR(url, destino, foco) {
26      if(document.getElementById) {
27          var ajax = XHR();
28          ajax.open("GET", url, true);
29          ajax.onreadystatechange = function() {
30              if(ajax.readyState == 1) {
31                  destino.innerHTML = "<h2>Carregando...</h2>";
32              }
33              if(ajax.readyState == 4) {
34                  if(ajax.status == 200) {
35                      destino.innerHTML = ajax.responseText;
36                  }
37                  else {
38                      destino.innerHTML = "<h2>[AJAX] Erro: " + ajax.status + "</h2>";
39                  }
40              }
41          }
42          ajax.send();
43      }
44  }
```

Figura 4: Requisição AJAX.

2.4 POSTGRESQL

O PostgreSQL é um Sistema Gerenciador de Banco de Dados (SGBD) objeto-relacional de código aberto, com mais de 15 anos de desenvolvimento. É extremamente robusto e confiável, além de ser flexível e rico em recursos. Ele é considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados. A equipe de desenvolvimento do PostgreSQL sempre teve uma grande preocupação em manter a compatibilidade com os padrões SQL92/SQL99 (OSLEI, 2004).

Segundo Biazus (2003a), pela riqueza de recursos e conformidade com os padrões, ele é um SGBD muito adequado para o estudo universitário do modelo relacional, além de ser uma ótima opção para empresas implementarem soluções de alta confiabilidade sem altos

custos de licenciamento. É um programa distribuído sob a licença *Berkeley Software Distribution* (BSD), o que torna o seu código-fonte disponível e o seu uso livre para aplicações comerciais ou não. O PostgreSQL foi implementado em diversos ambientes de produção no mundo, entre eles, um bom exemplo do seu potencial é o banco de dados que armazena os registros de domínio .org, mantido pela empresa Afilias.

Apesar de não ter a mesma fama que seus similares proprietários (Oracle, SQL Server e DB2), nos últimos anos o PostgreSQL obteve um reconhecimento maior dentre os outros bancos de dados. Isso porque possui várias características que, segundo Amorim e Silva Jr. (2005), o torna um SGBD robusto e flexível. Como exemplo de suas inovações pode-se citar o fato de ser um dos primeiros a introduzir muitos conceitos de objeto-relacional. Outras características que se pode citar são as seguintes:

a) implementação de integridade referencial;
b) suporte a transações;
c) suporte a subconsultas;
d) suporte aos padrões *American National Standards Institute* (ANSI) *Structured Query Language* (SQL) 89, 92 e 99;
e) implementa *Multi-Version Concurrency Control* MVCC, ou seja, resolve problemas relacionados a acessos concorrentes no banco de dados, o que poderia causar inconsistências nas informações armazenadas;
f) a possibilidade de criar funções definidas pelo usuário e também a criação de gatilhos (*triggers*);
g) esquemas (s*chemas*)
h) uso de herança de tabelas;
i) possibilidade de fazer backup on-line;
j) facilidades no uso dos objetos do banco;
k) possibilidade para o usuário definir novos tipos de dados;
l) possuir suporte ou grande integração com várias linguagens como PHP, Java, Perl, Python, Delphi, C++,Visual Basic, e outras;
m) integração com várias linguagens de programação para a criação de funções, na qual se pode utilizar desde o plpgSQL, muito semelhante ao PL/SQL, ou outras linguagens como C, plJava, plTcl, plPerl, plPython e plRuby, dentre outras;

n) conexão por interfaces nativas;

o) conexões *Secure Sockets Layer* (SSL).

É importante também frisar que o PostgreSQL pode ser instalado em plataformas Unix, como o Linux, FreeBSD, NetBSD, HPUX. Algumas distribuições do sistema operacional Linux, por exemplo, o Conectiva e o Red Hat já possuem o PostgreSQL incluso no pacote. Segundo Biazus (2003b), a partir da versão 8.0, o PostgreSQL apresenta suporte nativo também para a plataforma Windows, não precisando mais da camada de compatibilidade binária entre Linux e Windows chamada Cygwin. Por essas características, ele tem se tornado tão popular nos dias de hoje.

2.5 ANÁLISE DE REQUISITOS

De acordo com Rocha et al (2001), independentemente da área de aplicação, do tamanho do projeto ou de sua complexidade, o processo de desenvolvimento de software possui três fases genéricas: a definição, o desenvolvimento e a manutenção. E é dentro da fase de definição que se encontra a etapa de Análise de Requisitos.

Ainda segundo Rocha et al (2001), a fase de definição tem como objetivo definir quais informações serão processadas, quais funções e desempenhos são desejados, quais interfaces devem ser estabelecidas, quais restrições do projeto e critérios de validação são necessárias.

> Pode-se concluir que o primeiro passo em qualquer processo de desenvolvimento é descobrir o que o cliente quer e documentar os requisitos. [...] A análise é o processo de dividir as coisas em componentes que possam ser melhor entendidos (PFLEEGER, 2004).

Conforme mostra-se em Fernandes e Teixeira (2004), a especificação dos requisitos funcionais e não funcionais contempla as seguintes tarefas:

a) levantamento dos requisitos pelo exame de documentos, observação, entrevistas, reuniões, sessões conjuntas de especificação, workshop, e outros;

b) definição dos requisitos funcionais em termos de: propósito do produto (qual o problema ou oportunidade de negócio a que ele atende), quem vai ter acesso à solução

(pessoas, organizações, sistemas, intrusos), o que o produto vai fazer (objetos e componentes de que vai tratar), onde vai ser usado, regras de negócio principais que devem ser seguidas, por que o produto vai ser usado e como vai ser usado pelos clientes;

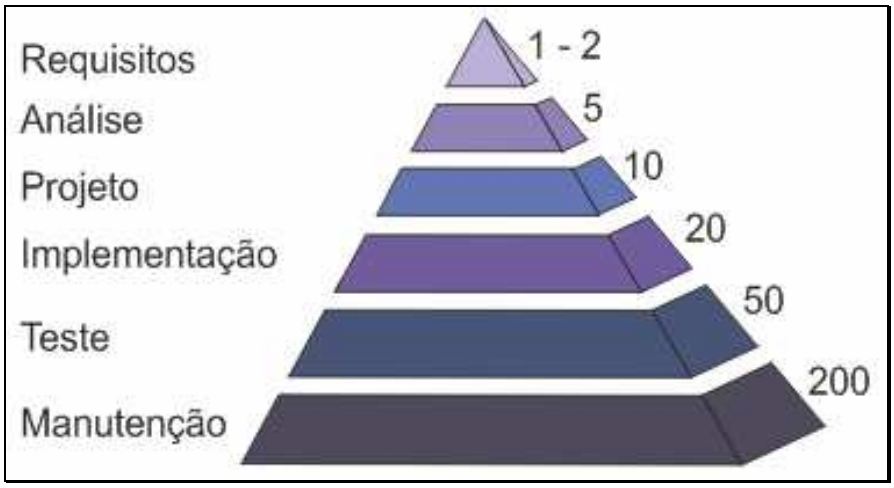
c) identificação e definição dos requisitos não funcionais em termos de: estética do produto, usabilidade, desempenho, aspectos operacionais, manutenibilidade, portabilidade, requerimentos de segurança, aspectos políticos, atendimento a aspectos legais e regulatórios, escalabilidade, confiabilidade, e outros;

d) identificar e definir restrições ao produto e ao projeto;

e) especificar os requisitos usando técnicas de engenharia de software, como diagramas de contexto, lista de eventos, modelo entidade-relacionamento, casos de uso, modelos de classe, técnicas de especificação de programas, e outros;

f) projeta, em alto nível, a arquitetura do software.

Segundo Magela (2006), a análise de requisitos é responsável por garantir a consistência e a qualidade dos requisitos especificados. Pode-se observar na figura 5 a Pirâmide de Propagação de Erro. Ela mostra a importância da análise de requisitos. Os números que aparecem à direita da pirâmide representam a proporção do custo - em valores monetários ou tempo em minutos. Ou seja, omitir ou especificar erroneamente um requisito custa 20 vezes mais se sua descoberta ou correção for realizada na fase de implementação e 200 vezes mais se for descoberta na manutenção.

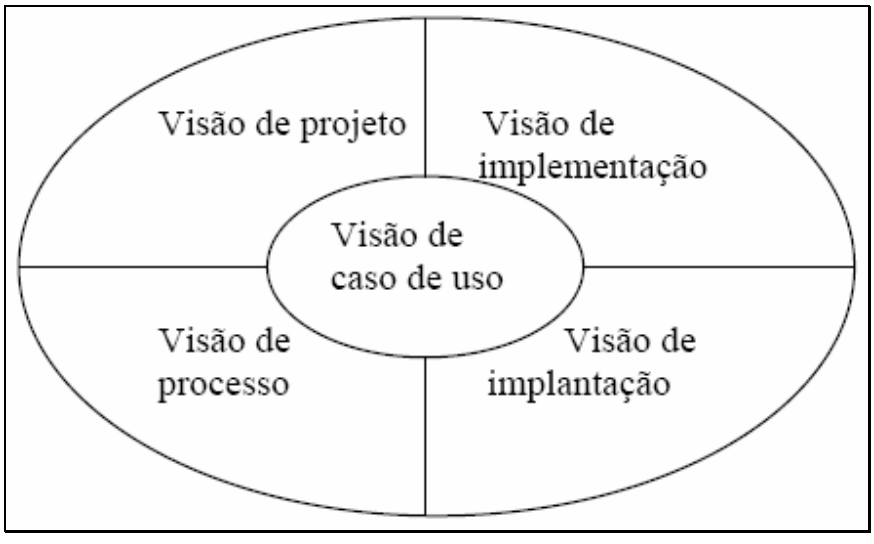


Fonte: MAGELA, 2006.

Figura 5: Pirâmide de Propagação de Erro.

2.5.1 Diagrama de casos de uso

A visão do caso de uso mostra conceitualmente o conjunto de funções que o sistema deve executar para atender aos requisitos do cliente, servindo como um contrato entre o cliente e o desenvolvedor. O sistema é visto sob a perspectiva do usuário, por isso a visão de caso de uso ocupa uma posição central, como pode ser visto na figura 6, que é a base para as demais visões, essencial para análise, desenho, implementação, testes e plano do desenvolvimento do sistema (LIMA, 2005).



Fonte: BEZERRA, 2003.

Figura 6: Visões de um sistema de software.

Segundo Cockburn (2005), um caso de uso captura um contrato entre os *stakeholders* de um sistema sobre seu comportamento. O caso de uso descreve o comportamento do sistema sob diversas condições conforme o sistema responde a uma requisição de um *stakeholder*, chamado ator primário. O ator primário inicia uma interação com o sistema para atingir algum objetivo. O sistema responde, protegendo os interesses de todos os *stakeholders*.

2.6 PROCESSO ATUAL

Observando uma empresa de informática de pequeno porte da cidade de Blumenau, no estado de Santa Catarina, pôde-se observar que a falta de um processo bem definido para solicitação de pequenas melhorias no sistema por parte do cliente pode levar a gastos financeiros desnecessários, bem como problemas de comunicação e maior lentidão na prestação do serviço.

No processo atual, que pode ser observado na figura 7, o cliente faz a solicitação da proposta por e-mail para a equipe responsável pela área comercial. Um dos atendentes comerciais entra em contato com o cliente por telefone para entender melhor o serviço que o cliente gostaria de receber. Após este entendimento, o atendente comercial manda um e-mail para o analista que gera um documento contendo os requisitos funcionais e não funcionais.

Utilizando-se de ferramentas gratuitas de modelagem de dados, o analista desenha um diagrama de casos de uso e o adiciona ao documento juntamente com os requisitos. Esse documento é passado para o atendente comercial. Eventualmente, o analista pode passar alguma planilha para o atendente comercial explicando os custos e as possibilidades de tempo para a execução do processo.

Com o auxílio dessa documentação, o atendente comercial manda um e-mail com a proposta para o cliente. Se este aprovar manda uma resposta positiva na forma de e-mail.

Neste caso, o atendente elabora um documento de contrato e envia para o cliente. Este por sua vez o assina e com isso pode-se dar início ao desenvolvimento do que foi solicitado.

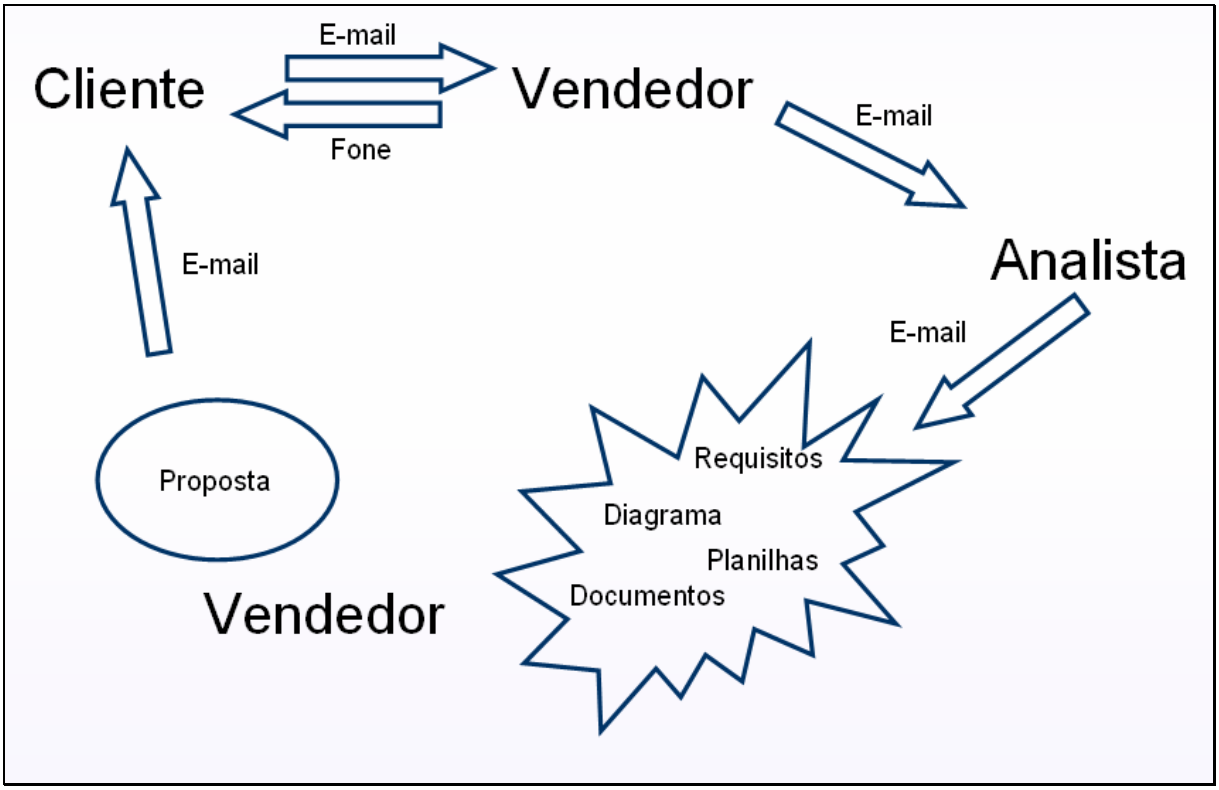


Figura 7: Processo atual.

2.7 TRABALHOS CORRELATOS

Fachini (2005), desenvolveu um sistema de informação, também em PHP orientado a objetos, para acompanhamento de chamados técnicos e *workflow* via web tornando assim mais fácil e ágil os processos de registro de chamados e atividades internas. O aplicativo disponibiliza estas funcionalidades para os funcionários e clientes na Internet, agregando valores ao atendimento de cliente e proporcionando agilidade, segurança e confiabilidade.

Góis (2006), implementou uma aplicação web para monitoramento gráfico de ambientes físicos para controle de acesso e segurança. Seu principal objetivo foi monitorar ambientes de forma gráfica com um aplicativo *web-based* devido à necessidade de utilização de interfaces com muitos recursos de interação. Utilizou-se de AJAX e da linguagem *Scalable*

Vector Graphics (SVG) para possibilitar que as páginas web, normalmente estáticas, tivessem uma interatividade comparável à de aplicativos desktop.

Reinert (2006), desenvolveu uma ferramenta de *workflow* para auxiliar no controle e execução das atividades que envolvem um processo de software. O objetivo principal foi promover o fluxo organizado de troca de informações entre os envolvidos no processo, tendo como conseqüência um aumento na produtividade e melhoria na qualidade do processo de desenvolvimento.

Razente, Barioni, Traina e Traina Jr. (2006), implementaram um sistema de busca de imagens médicas por conteúdo usando o banco de dados PostgreSQL. Neste trabalho, apontam como características positivas a possibilidade de uso de índices para pesquisa em grandes volumes de dados e baixo custo na implantação do sistema.

Marisco, Philips e Pereira (2004), desenvolveram um protótipo de Mapa para web interativo que utilize códigos padronizados e fontes abertos, tais como XML, SVG, DOM e linguagens de script ECMAScript/JavaScript e PHP, bem como o PostgreSQL e sua extensão, o PostGIS, para disseminar informações relacionadas ao cadastro imobiliário urbano municipal. Para isso, utilizaram-se de dados fornecidos pela Prefeitura Municipal de São José – Santa Catarina, referentes ao bairro Campinas.

Ainda Marisco, Philips e Pereira (2004), utilizando-se do modelo computacional Cliente/Servidor, projetaram o protótipo do mapa para web com códigos padronizados, de forma a permitir que o usuário possa visualizar o mapa para web sem a necessidade de instalar programas de computador específico, a não ser o *plug-in* Adobe SVG Viewer 3.0, em seu navegador. No intuito de atender aos preceitos de um bom projeto cartográfico para a web, procurou-se obedecer às regras da tradução gráfica e implementar diferentes funcionalidades de interação, das quais destacam-se as legendas interativas, a simbolização e a escala dinâmica. Dos resultados alcançados pode-se concluir que é altamente recomendada a

utilização dos códigos padronizados e das tecnologias códigos-fonte abertos em projetos de mapas para web interativos.

Gielow (2003), desenvolveu uma ferramenta para cálculo dos *Use Case Points* a partir da leitura de diagramas gerados pela ferramenta *Computer Aided Systems Engineering* (CASE) da empresa Rational Rose. Tendo em vista que a modelagem de casos de uso é muito utilizada na análise orientada a objetos para capturar e descrever os requisitos do sistema. E que estes, por sua vez, podem servir como medição do tamanho e complexidade do sistema.

3 DESENVOLVIMENTO DO TRABALHO

De acordo com os objetivos deste trabalho, foi desenvolvido um sistema para gerenciamento e acompanhamento de propostas e requisitos. Tal sistema consiste em um aplicativo web escrito na linguagem PHP. Utiliza-se como banco de dados o PostgreSQL. E para tornar a interface mais interativa, utiliza-se de JavaScript e ActionScript.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O Quadro 1 apresenta os requisitos funcionais previstos para o sistema e sua rastreabilidade, ou seja, vinculação com o(s) caso(s) de uso associado(s).

Requisitos Funcionais	**Caso de Uso**
RF01: O sistema deverá permitir aos usuários efetuarem login	UC01, UC02, UC03, UC04, UC05 e UC06
RF02: O sistema deverá permitir ao administrador cadastrar e manter os cadastros de analistas.	UC03
RF03: O sistema deverá permitir ao analista cadastrar e manter os cadastros de clientes.	UC03
RF04: O sistema deverá permitir ao cliente manter seus dados cadastrais.	UC02
RF05: O sistema deverá permitir ao analista visualizar todas as propostas de seus clientes.	UC05
RF06: O sistema deverá permitir ao cliente visualizar suas propostas.	UC04 e UC06
RF07: O sistema deverá permitir ao cliente incluir nova proposta.	UC04
RF08: O sistema deverá permitir ao cliente cancelar suas propostas.	UC06

RF09: O sistema deverá permitir ao analista editar os requisitos funcionais e não funcionais da proposta cadastrada pelo cliente e demais informações anexadas.	UC05
RF10: O sistema deverá permitir aos analistas editar um diagrama de casos de uso para cada proposta.	UC05
RF11: O sistema deverá permitir aos clientes incluírem e excluírem arquivos em anexo a proposta.	UC04 e UC06
RF12: O sistema deverá permitir aos clientes aprovar os requisitos informados pelo analista, o que resulta na aprovação da proposta.	UC06

Quadro 1: Requisitos funcionais

O Quadro 2 lista os requisitos não funcionais previstos para o sistema.

Requisitos Não Funcionais
RNF01: O sistema deverá utilizar a linguagem PHP
RNF02: A interface deve ser flexível e dinâmica, através de AJAX
RNF03: O banco de dados a ser utilizado é PostgreSQL
RNF04: Para a tela de diagramação deverá ser utilizada a linguagem ActionScript

Quadro 2: Requisitos não-funcionais

3.2 ESPECIFICAÇÃO

Nesta seção é apresentada a especificação do problema através de modelos e diagramas que representem logicamente o trabalho desenvolvido. Também são citadas técnicas e ferramentas utilizadas para fazer a especificação.

Para a especificação foi utilizada a linguagem visual *Unified Modeling Language* (UML). Foram escolhidos para tal os diagramas de casos de uso, de atividades e de classes, desenvolvidos com a tela de diagramação deste trabalho e com a ferramenta *Enterprise Architect* (EA). O Modelo Entidade-relacionamento foi construído utilizando-se a ferramenta *PowerDesigner*.

3.2.1 Regras de negócio

Na especificação do trabalho foi constatada a necessidade de se implementar as seguintes regras de negócio:

a) RN01 – Inclusão de um novo usuário: um administrador somente pode cadastrar analistas. Um analista somente pode cadastrar clientes;
b) RN02 – Estados da proposta: quando uma proposta é incluída pelo Cliente ela fica com estado de aberta. Caso todos os requisitos estejam analisados, cancelados ou aprovados, a proposta tem seu estado alterado para analisada. Caso todos os requisitos estejam cancelados ou aprovados, a proposta tem seu estado alterado para aprovada. Caso o cliente feche negócio a proposta tem seu estado alterado para fechada. Caso algum de seus requisitos esteja como aberto, a proposta tem

seu estado alterado para aberta. Caso o cliente cancela a proposta, ela tem seu estado alterado para cancelada;

c) RN03 – Inclusão de requisitos: quando um requisito é incluído pelo Analista ele fica com estado de analisado.

3.2.2 Diagramas de casos de uso

A seguir, nas figuras 8, 9 e 10, são apresentados os modelos dos diagramas de casos de uso do sistema.

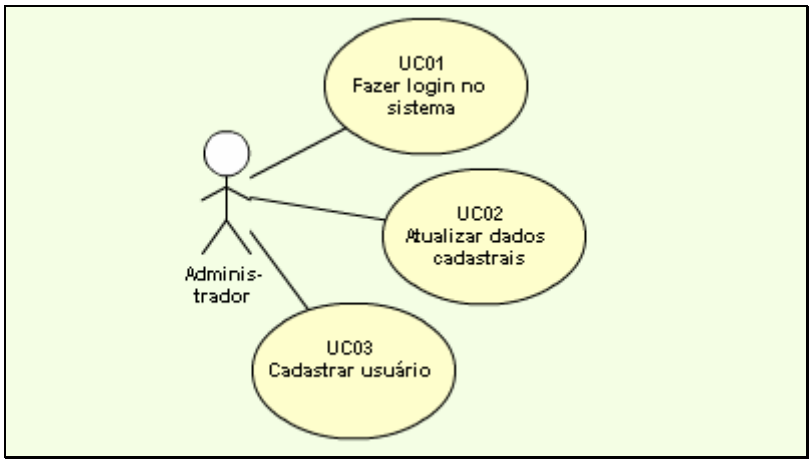


Figura 8: Diagrama de casos de uso para a visão do administrador

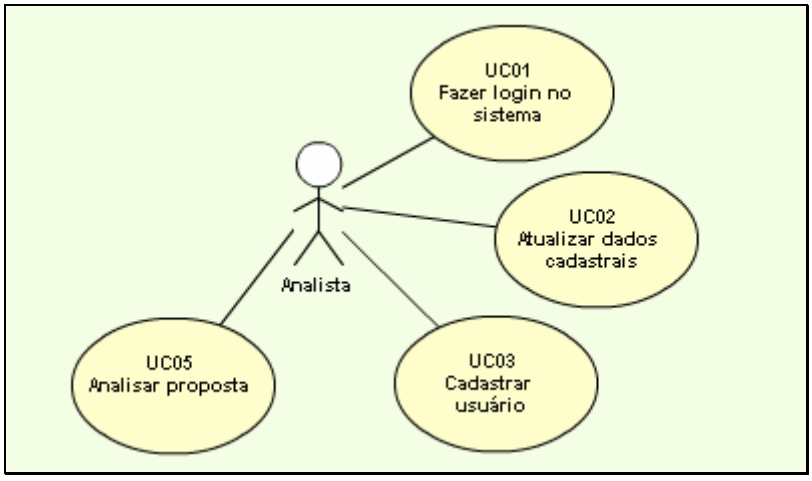


Figura 9: Diagrama de casos de uso para a visão do analista

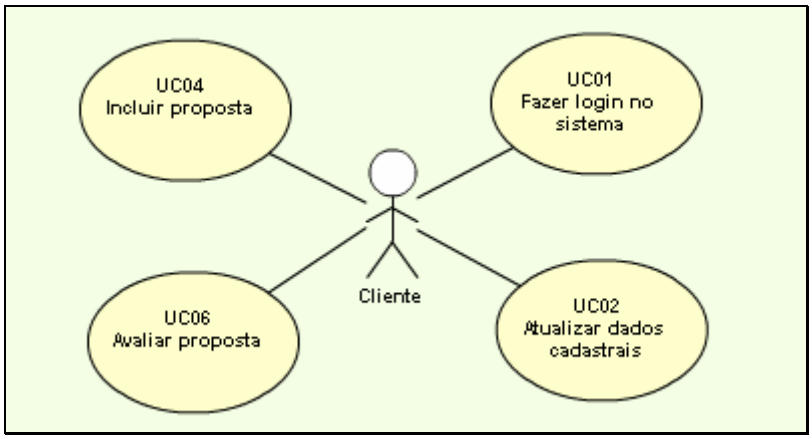


Figura 10: Diagrama de casos de uso para a visão do cliente

3.2.3 Descrição dos cenários

A seguir são apresentadas as descrições dos casos de uso do sistema.

UC01: Fazer login no sistema

Pré-condições: Usuário deve estar cadastrado no sistema.

Pós-condições: Usuário possui acesso à tela principal, à tela de usuários e à tela de visualização de proposta.

Fluxo principal:

1. Usuário informa e-mail e senha.
2. O sistema autentica as informações.

Fluxos alternativos:

2a. Usuário não cadastrado.

1. O sistema mostra mensagem informando que usuário não está cadastrado.
2. Reinicia o fluxo principal.

UC02: Atualizar dados cadastrais

Pré-condições: O Usuário está identificado pelo sistema.

Pós-condições: Atributos senha, fone, contato e forma de contato alterados do usuário.

Fluxo principal:

1. O Usuário preenche os dados de identificação do usuário: senha e confirmação de senha.
2. O Usuário preenche os dados do contato: fone, contato e forma de contato.
3. O sistema grava os dados alterados.

Fluxos alternativos:

1a. Usuário preenche dados complementares.

1. Se o Usuário for Administrador ou Analista ele pode alterar os campos da empresa, informando razão social, CNPJ e fone. Pode ainda alterar os dados do responsável técnico, informando nome e CPF.
2. Retoma o fluxo principal no passo 2.

Fluxos de exceção:

3a. Aviso ao gravar dados.

1. Caso o Usuário tente gravar seus dados sem informar os campos senha, confirmação, fone de contato e celular, o sistema apresenta uma mensagem solicitando preenchimento.

UC03: Cadastrar usuário

Pré-condições: O Usuário está identificado pelo sistema e deve ser Administrador ou Analista.

Pós-condições: Usuário criado.

Fluxo principal:

1. O Usuário seleciona através da lista de usuários a opção para incluir um novo usuário.
2. O Usuário preenche os dados de identificação do novo usuário: senha e confirmação de senha.
3. O Usuário informa os dados da empresa: razão social, CNPJ e fone.
4. O Usuário informa os dados do responsável técnico: nome e CPF.
5. O Usuário preenche os dados do contato: fone, contato e forma de contato.
6. O sistema grava os dados alterados.

Fluxos de exceção:

3a. Aviso ao gravar dados.

1. Caso o Usuário tente gravar seus dados sem informar os campos senha, confirmação, fone de contato e celular, o sistema apresenta uma mensagem solicitando preenchimento.

UC04 – Incluir proposta

Pré-condições: O Cliente está identificado pelo sistema.

Pós-condições: A proposta incluída com estado de acordo com a RN02.

Fluxo principal:

1. O Cliente solicita inclusão de uma proposta.
2. O Cliente preenche os campos da proposta: título e descrição.
3. O sistema grava os dados alterados.

Fluxos alternativos:

2a. Cliente anexa arquivos.

1. O Cliente anexa contratos, documentos e planilhas.
2. Retoma o fluxo principal no passo 2.

Fluxos de exceção:

3a. Aviso ao gravar dados.

1. Caso o Cliente tente gravar os dados sem informar os campos título e descrição, o sistema apresenta uma mensagem solicitando preenchimento.

UC05 – Analisar proposta

Pré-condições: O Analista está identificado pelo sistema. A proposta está cadastrada pelo Cliente.

Pós-condições: A proposta está apta a ser avaliada pelo Cliente.

Fluxo principal:

1. O Analista visualiza as propostas de seus clientes.
2. O Analista altera título e descrição da proposta.
3. O Analista inclui requisitos funcionais informando sequência, descrição, tempo e custo.
4. O Analista inclui requisitos não funcionais informando sequência, descrição e custo. Seu estado é alterado conforme RN03.
5. O sistema grava os dados alterando o estado da proposta de acordo com a RN02.

Fluxos alternativos:

4a. Analista anexa arquivos.

1. O Analista anexa contratos, documentos e planilhas.
2. Retoma o fluxo principal no passo 4.

4b. Analista anexa arquivos.

1. O Analista anexa contratos, documentos e planilhas.
2. Retoma o fluxo principal no passo 4.

Fluxos de exceção:

5a. Aviso ao gravar dados.

1. Caso o Analista tente gravar uma proposta sem que o título e a descrição estejam preenchidos ou o Analista tente incluir algum requisito não informando qualquer um

dos seus campos (Seqüência, Descrição, Tempo e Custo), o sistema apresenta uma mensagem de aviso.

UC06 – Avaliar proposta

Pré-condições: O Cliente está identificado pelo sistema. A proposta deverá ter sido analisada pelo Analista.

Pós-condições: A proposta está apta para o desenvolvimento.

Fluxo principal:

1. O Cliente visualiza suas próprias propostas.
2. O Cliente escolhe a proposta ser avaliada.
3. O Cliente aprova os requisitos avaliados pelo analista.
4. O sistema grava os dados alterando o estado da proposta para aprovada de acordo com a RN02.
5. O Cliente fecha negócio aprovando a proposta.
6. O sistema grava os dados alterando o estado da proposta para fechada de acordo com a RN02.

Fluxos alternativos:

2a. Cliente cancela proposta.

1. O Cliente solicita cancelamento de proposta.
2. O sistema grava os dados alterando o estado da proposta para cancelada de acordo com a RN02.

3a. Cliente rejeita requisitos.

3. O Cliente rejeita algum dos requisitos.
4. O sistema grava os dados alterando o estado da proposta para aberta de acordo com a RN02.
5. O caso de uso é encerrado.

3.2.4 Diagramas de atividades

Na figura 11, tem-se a representação do diagrama de atividades, onde demonstra-se todo o processo de inclusão, acompanhamento e aprovação de uma proposta na visão do analista e do cliente. Assim sendo, o cliente inclui uma proposta. O analista deve incluir os requisitos funcionais e não funcionais desta proposta e desenhar o diagrama de casos de uso.

Após este processo, o cliente deve avaliar os requisitos informados pelo analista, podendo aprovar ou reabrir. É permitido ao cliente cancelar a proposta e em caso de aprovação, é necessária uma confirmação para que se “feche o negócio”.

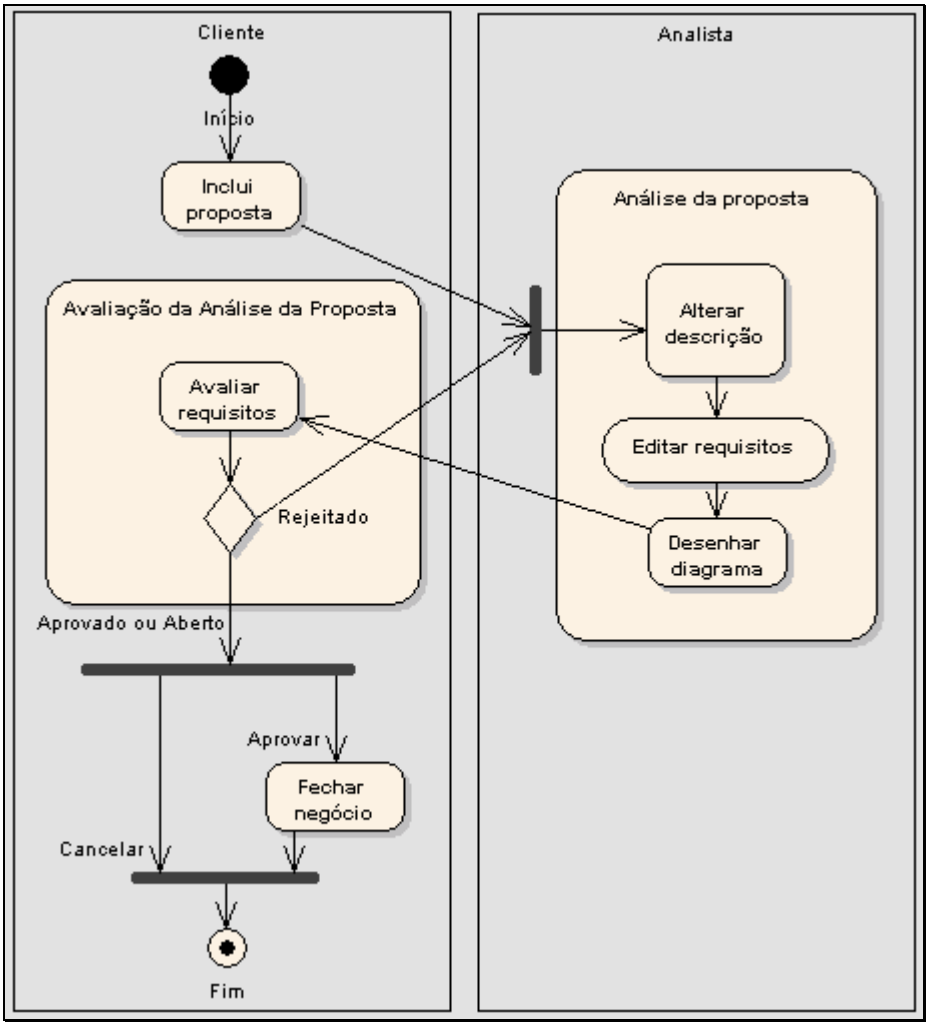


Figura 11: Diagrama de atividades do processo completo

3.2.5 Diagramas de classes

Os diagramas de classes do sistema foram divididos em 6 pacotes, conforme demonstrado na figura 12: `Dados`, `Diversos`, `DAO`, `ComandosPG`, `Atualiza` e `Consulta`.

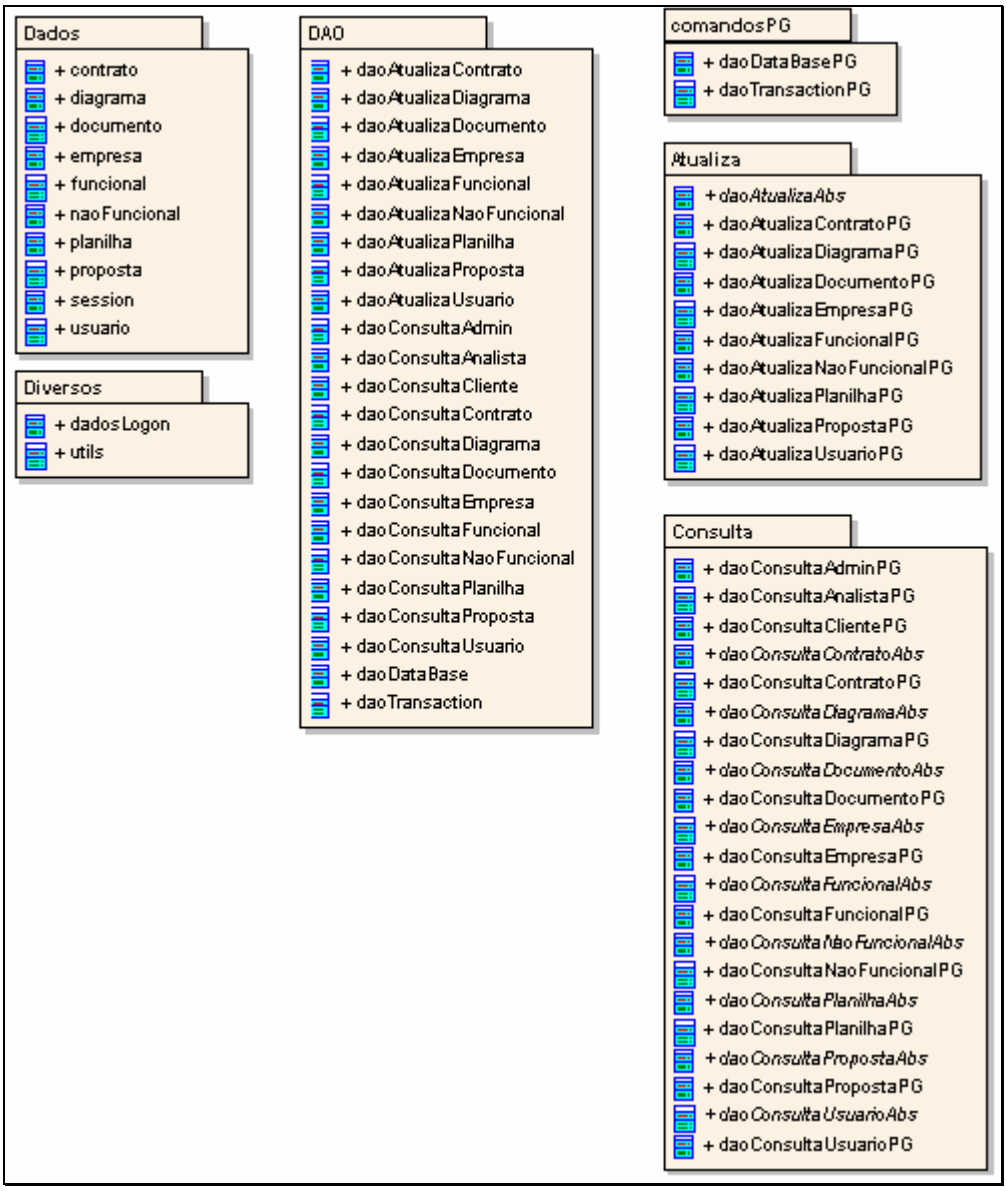


Figura 12: Pacotes do diagrama de classes

O pacote Dados que é demonstrado na figura 13 é responsável pelas classes que armazenam os dados. Este pacote é composto pelas seguintes classes:

a) `empresa`: é um registro da tabela de empresas (t_emp);

b) `usuario`: é um registro da tabela de usuários (t_usu);

c) `proposta`: é um registro da tabela de propostas (t_pro);

d) `funcional`: é um registro da tabela de requisitos funcionais (t_fun);

e) `naoFuncional`: é um registro da tabela de requisitos não funcionais (t_rnf);

f) `diagrama`: é um registro da tabela de diagramas (t_dia);

g) `contrato`: é um registro da tabela de contratos (t_con);

h) `documento`: é um registro da tabela de documentos (t_doc);

i) `planilha`: é um registro da tabela de planilhas (t_pla);

j) `session`: responsável por criar, deletar e validar sessões PHP. Armazena dados do usuário logado na sessão.

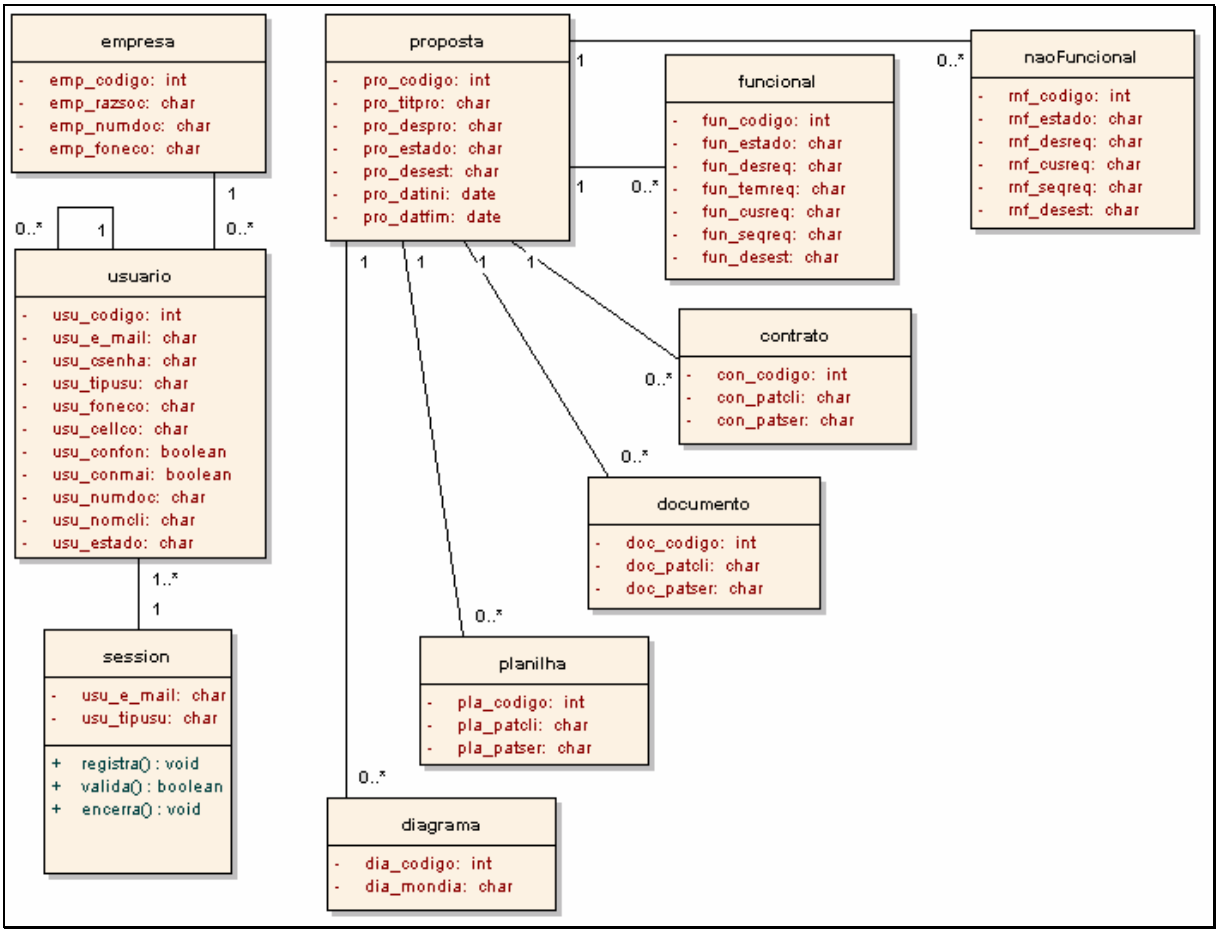


Figura 13: Diagrama de classes do pacote Dados.

O pacote `ComandosPG` que é demonstrado na figura 14 é responsável pelas classes que utilizam comandos PostgreSQL para efetuar consultas, transações, entre outras ações. Este pacote é composto pelas seguintes classes:

a) `daoDataBasePG`: responsável por gerenciar as conexões com o banco, bem como, efetuar consultas;

b) `daoTransactionPG`: responsável por gerenciar transações com o banco de dados.

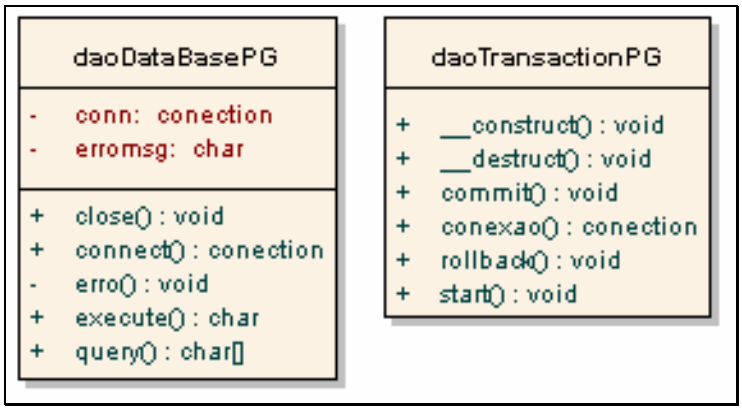


Figura 14: Diagrama de classes do pacote ComandosPG.

O pacote `Atualiza` que é demonstrado na figura 15 é responsável pelas classes que montam comandos para atualização das tabelas. Nestas classes existem comandos no padrão ANSI e específicos para o banco PostgreSQL. Este pacote é composto pelas seguintes classes:

a) `daoAtualizaAbs`: possui funções genéricas para atualização de uma tabela;

b) `daoAtualizaUsuarioPG`: possui funções específicas para montar comandos de atualização (*insert, update*) da tabela de usuários (t_usu) em PostgreSQL;

c) `daoAtualizaEmpresaPG`: possui funções específicas para montar comandos de atualização (*insert, update*) da tabela de empresas (t_emp) em PostgreSQL; também é capaz de retornar o valor da próxima chave primária desta tabela;

d) `daoAtualizaPropostaPG`: possui funções específicas para montar comandos de atualização (*insert, update*) da tabela de propostas (t_pro) em PostgreSQL; também é capaz de retornar o valor da próxima chave primária desta tabela;

e) `daoAtualizaFuncionalPG`: possui funções específicas para montar comandos de atualização (*insert, update*) da tabela de requisitos funcionais (t_fun) em PostgreSQL;

f) `daoAtualizaNaoFuncionalPG`: possui funções específicas para montar comandos de atualização (*insert, update*) da tabela de requisitos não funcionais (t_rnf) em PostgreSQL;

g) `daoAtualizaDiagramaPG`: possui funções específicas para montar comandos de atualização (*insert, update*) da tabela de diagramas (t_dia) em PostgreSQL;

h) `daoAtualizaContratoPG`: possui funções específicas para montar comandos de atualização (*insert, update, delete*) da tabela de contratos (t_con) em PostgreSQL;

i) `daoAtualizaDocumentoPG`: possui funções específicas para montar comandos de atualização (*insert, update, delete*) da tabela de documentos (t_doc) em PostgreSQL;

j) `daoAtualizaPlanilhaPG`: possui funções específicas para montar comandos de atualização (*insert, update, delete*) da tabela de planilhas (t_pla) em PostgreSQL.

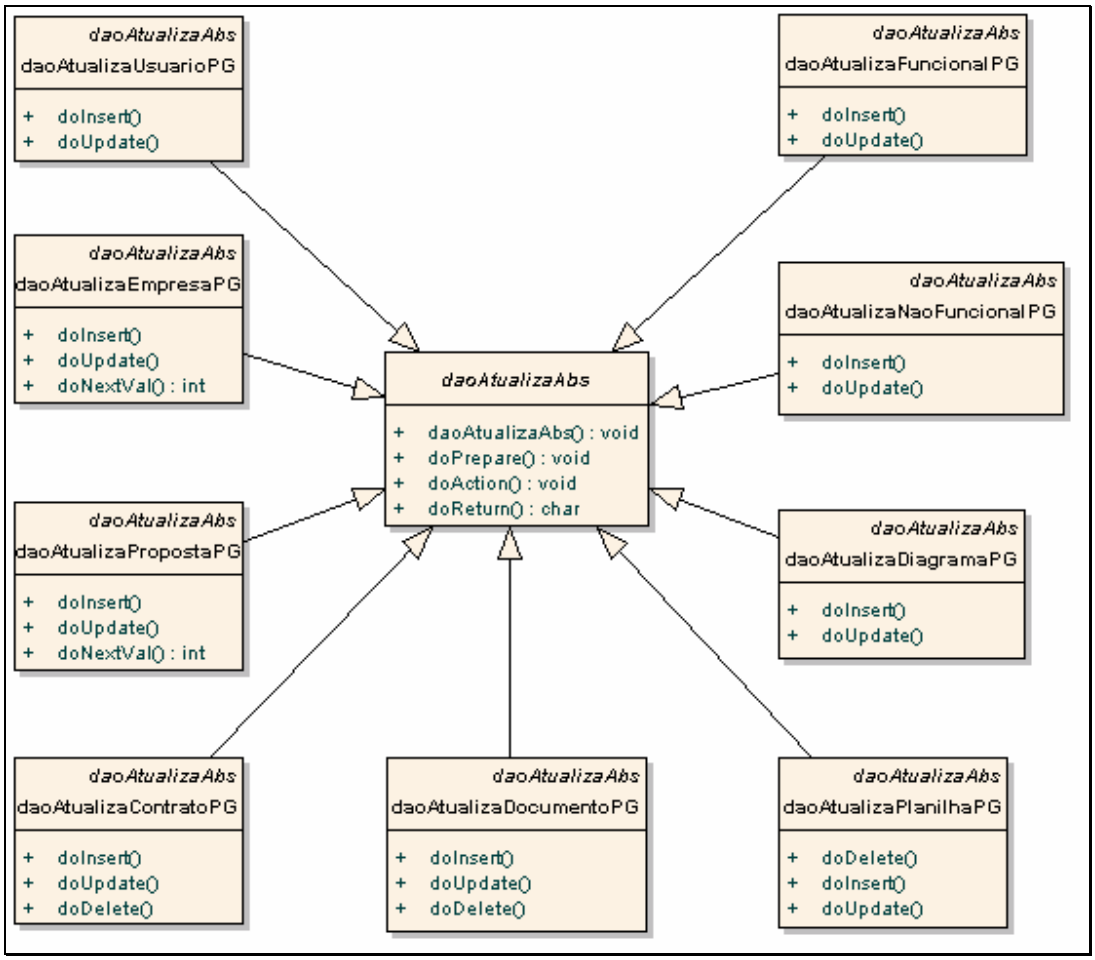


Figura 15: Diagrama de classes do pacote Atualiza.

O pacote `Consulta` que é demonstrado na figura 16 é responsável pelas classes que montam comandos para consulta das tabelas. Este pacote é composto pelas seguintes classes:

a) `daoConsultaUsuarioAbs`: classe responsável por efetuar a consulta (*select*) e manipulação do resultado da tabela de usuários (t_usu);

b) `daoConsultaUsuarioPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de usuários;

c) `daoConsultaAdminPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de usuários do tipo administrador;

d) `daoConsultaAnalistaPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de usuários do tipo analista;

e) `daoConsultaClientePG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de usuários do tipo cliente;

f) `daoConsultaEmpresaAbs`: classe responsável por efetuar a consulta (*select*) e manipulação do resultado da tabela de empresas (t_emp);

g) `daoConsultaEmpresaPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de empresa;

h) `daoConsultaPropostaAbs`: classe responsável por efetuar a consulta (*select*) e manipulação do resultado da tabela de propostas (t_pro);

i) `daoConsultaPropostaPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de proposta;

j) `daoConsultaFuncionalAbs`: classe responsável por efetuar a consulta (*select*) e manipulação do resultado da tabela de requisitos funcionais (t_fun);

k) `daoConsultaFuncionalPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de requisitos funcionais;

l) `daoConsultaNaoFuncionalAbs`: classe responsável por efetuar a consulta (*select*) e manipulação do resultado da tabela de requisitos não funcionais (t_usu);

m) `daoConsultaNaoFuncionalPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de requisitos não funcionais;

n) `daoConsultaDiagramaAbs`: classe responsável por efetuar a consulta (*select*) e manipulação do resultado da tabela de diagramas (t_dia);

o) `daoConsultaDiagramaPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de diagramas;

p) `daoConsultaContratoAbs`: classe responsável por efetuar a consulta (*select*) e manipulação do resultado da tabela de contratos (t_con);

q) `daoConsultaContratoPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de contratos;

r) `daoConsultaDocumentoAbs`: classe responsável por efetuar a consulta (*select*) e manipulação do resultado da tabela de documentos (t_doc);

s) `daoConsultaDocumentoPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de documentos;

t) `daoConsultaPlanilhaAbs`: classe responsável por efetuar a consulta (*select*) e manipulação do resultado da tabela de planilhas (t_pla);

u) `daoConsultaPlanilhaPG`: classe que constrói comando de consulta em PostgreSQL para obter uma lista ou um registro de planilhas.

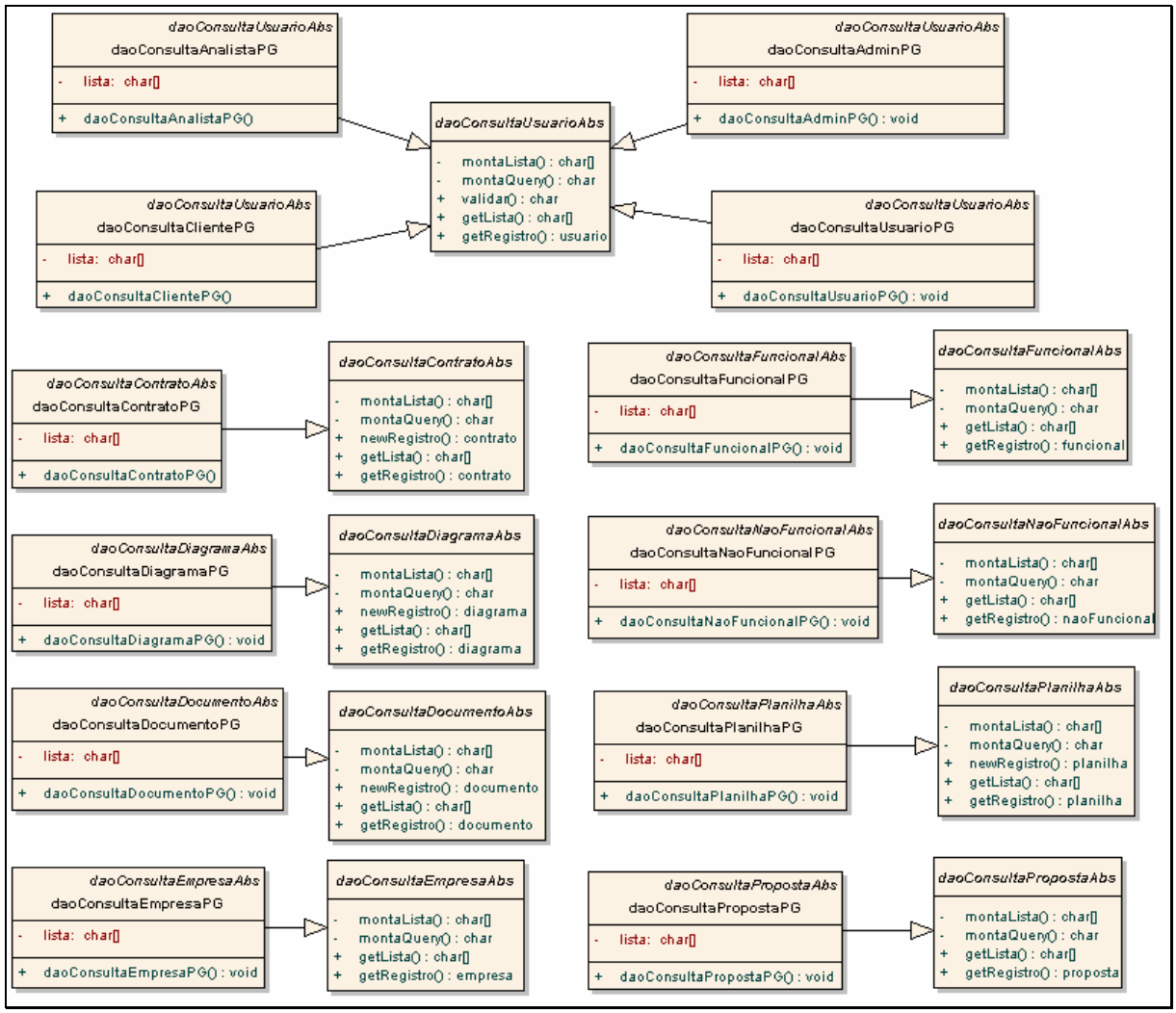


Figura 16: Diagrama de classes do pacote Consulta.

A figura 17 apresenta o pacote DAO onde estão as fábricas de objetos capazes de distinguir o banco de dados utilizado para execução do processo. Para isso utilizou-se o padrão de projeto *Data Access Object* (DAO) que visa tornar a camada de negócios mais independente da camada de persistência e do local onde os dados são persistidos. Todas as classes desse pacote têm funcionamento semelhante. Elas “fabricando” um objeto da classe que tem o mesmo nome acrescido do prefixo informado no atributo “banco_dados” da classe Util::dadosLogon.

Figura 17: Diagrama de classes do pacote DAO.

O pacote `Diversos`, que é demonstrado na figura 18 é responsável pelas classes que possuem funções utilitárias. Este pacote é composto pelas seguintes classes:

a) `utils`: possui métodos diversos escritos em PHP para resolver problemas pontuais, tais como, redirecionamento de página, manipulação de *strings* e gerenciamento de diretórios para *upload*;

b) `dadosLogon`: armazena, através de uma constante, o banco de dados escolhido em "tempo de desenvolvimento".

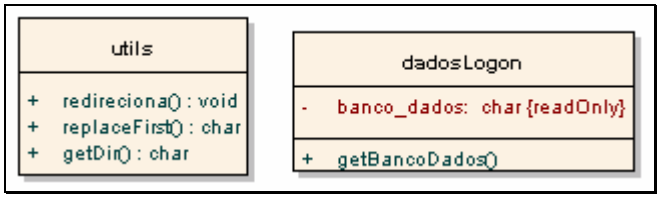


Figura 18: Diagrama de classes do pacote Diversos.

3.2.6 Modelo entidade-relacionamento

Na figura 19 é apresentado o Modelo Entidade-Relacionamento (MER) do sistema. Este diagrama foi obtido através de engenharia reversa na base do sistema.

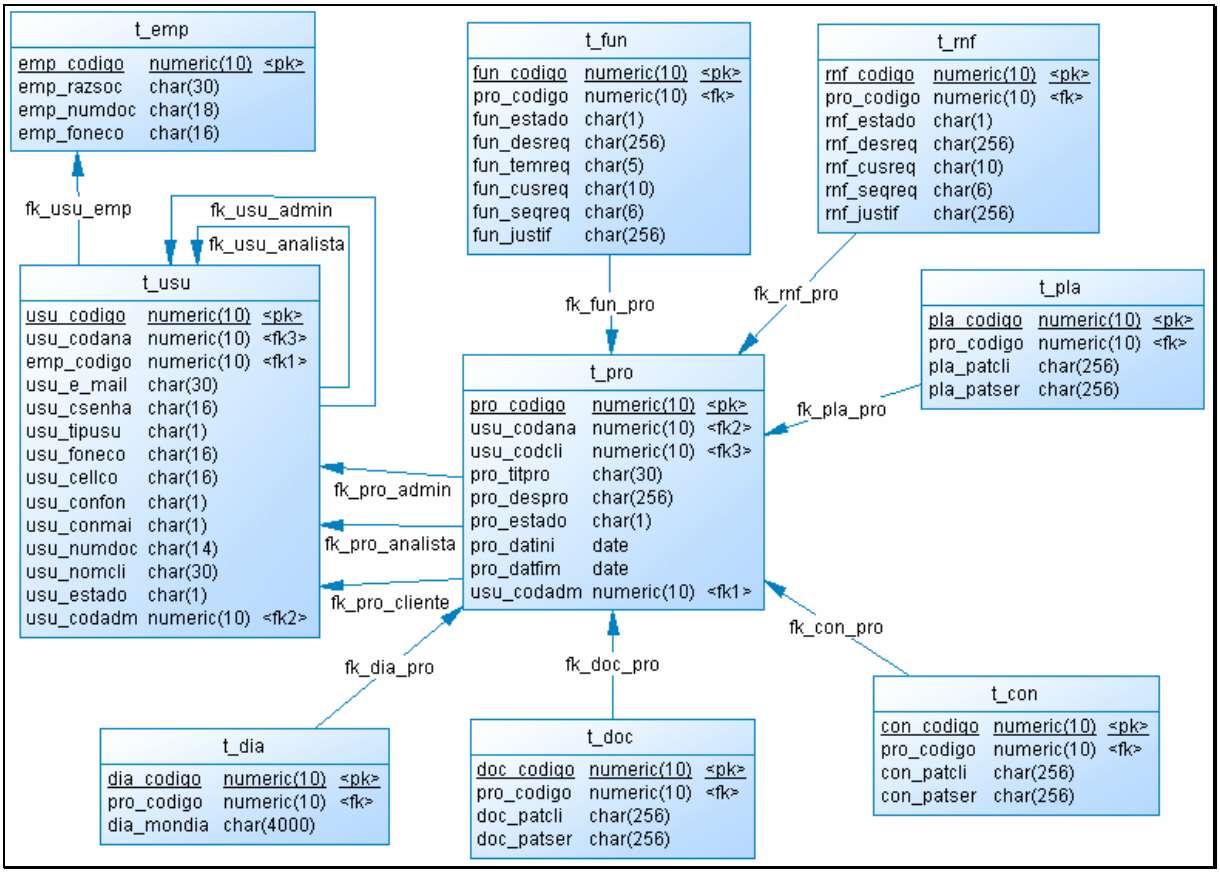


Figura 19: Modelo Entidade-Relacionamento Físico.

3.2.7 Dicionário de dados

O dicionário de dados desenvolvido para especificar o sistema é apresentado a seguir nos quadros 3, 4, 5, 6, 7, 8, 9 e 10.

t_emp – A tabela t_emp serve para armazenar as empresas vinculadas aos usuários.			
Campo	**Descrição**	**Tipo**	**Tamanho**
emp_codigo	Código da empresa	numeric	10
emp_razsoc	Razão Social	character	30
emp_numdoc	CNPJ da empresa	character	18
emp_foneco	Telefone de contato da empresa	character	16

Quadro 3 – Tabela de Empresas.

t_usu – A tabela t_usu serve para armazenar os usuários.			
Campo	**Descrição**	**Tipo**	**Tamanho**
usu_codigo	Código do usuário	numeric	10
usu_codadm	Código do usuário administrador	numeric	10
usu_codana	Código do usuário analista	numeric	10
emp_codigo	Código da empresa	numeric	10
usu_e_mail	E-mail do usuário	character	30
usu_csenha	Senha do usuário	character	16
usu_tipusu	Tipo de usuário: A - Analista C - Cliente O - Administrador	character	1
usu_foneco	Telefone do usuário	character	16
usu_cellco	Celular do usuário	character	16
usu_confon	Contato por telefone: T - Verdadeiro F - Falso	character	1
usu_conmai	Contato por e-mail: T - Verdadeiro F - Falso	character	1
usu_numdoc	CPF do usuário	character	14
usu_nomcli	Nome do usuário	character	30
usu_estado	Estado do usuário: A - Ativado E - Excluído	character	1

Quadro 4 – Tabela de Usuários.

t_pro – A tabela t_pro serve para armazenar as propostas.			
Campo	**Descrição**	**Tipo**	**Tamanho**
pro_codigo	Código da proposta	numeric	10
usu_codadm	Código do usuário administrador	numeric	10
usu_codana	Código do usuário analista	numeric	10
usu_codcli	Código do usuário cliente	numeric	10
pro_titpro	Titulo da proposta	character	30
pro_despro	Descrição da proposta	character	256
pro_estado	Estado da proposta: A - Aberta N - Analisada C - Cancelada P - Aprovada F - Fechada	character	1
pro_datini	Data em que foi criada a proposta	date	
pro_datfim	Data da conclusão da proposta	date	

Quadro 5 – Tabela de Propostas.

t_fun – A tabela t_fun serve para armazenar os requisitos funcionais.			
Campo	**Campo**	**Campo**	**Campo**
fun_codigo	Código do requisito funcional	numeric	10
pro_codigo	Código da proposta	numeric	10
fun_estado	Estado do requisito funcional: A - Aberto N - Analisado C - Cancelado P - Aprovado	character	1
fun_desreq	Descrição do requisito funcional	character	256
fun_temreq	Tempo estimado para desenvolvimento do requisito funcional	character	5
fun_cusreq	Custo estimado para o desenvolvimento do requisito funcional	character	10
fun_seqreq	Seqüência do requisito funcional	character	6

Quadro 6 – Tabela de Requisitos Funcionais.

t_rnf – A tabela t_rnf serve para armazenar os requisitos não funcionais.			
Campo	**Campo**	**Campo**	**Campo**
rnf_codigo	Código do requisito não funcional	numeric	10
pro_codigo	Código da proposta	numeric	10
rnf_estado	Estado do requisito não funcional: A - Aberto N - Analisado C - Cancelado P - Aprovado	character	1
rnf_desreq	Descrição do requisito não funcional	character	256
rnf_cusreq	Custo estimado para o desenvolvimento do requisito não funcional	character	10
rnf_seqreq	Seqüência do requisito não funcional	character	6

Quadro 7 – Tabela de Requisitos Não Funcionais.

t_dia – A tabela t_dia serve para armazenar os diagramas de casos de uso.			
Campo	**Campo**	**Campo**	**Campo**
dia_codigo	Código do diagrama	numeric	10
pro_codigo	Código da proposta	numeric	10
dia_mondia	Dados para montagem do diagrama	character	4000

Quadro 8 – Tabela de Diagramas.

t_con – A tabela t_con serve para armazenar os contratos anexados à proposta.			
Campo	**Campo**	**Campo**	**Campo**
con_codigo	Código do contrato	numeric	10
pro_codigo	Código da proposta	numeric	10
con_patcli	Caminho do arquivo no cliente	character	256
con_patser	Caminho do arquivo no servidor	character	256

Quadro 9 – Tabela de Contratos.

t_doc – A tabela t_doc serve para armazenar os documentos anexados à proposta.			
Campo	**Campo**	**Campo**	**Campo**
doc_codigo	Código do documento	Numeric	10
pro_codigo	Código da proposta	Numeric	10
doc_patcli	Caminho do arquivo no cliente	Character	256
doc_patser	Caminho do arquivo no servidor	Character	256

Quadro 10 – Tabela de Documentos.

t_pla – A tabela t_pla serve para armazenar as planilhas anexadas à proposta.			
Campo	**Campo**	**Campo**	**Campo**
pla_codigo	Código da planilha	Numeric	10
pro_codigo	Código da proposta	Numeric	10
pla_patcli	Caminho do arquivo no cliente	Character	256
pla_patser	Caminho do arquivo no servidor	Character	256

Quadro 10 – Tabela de Planilhas.

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas as ferramentas e técnicas utilizadas para a implementação do sistema. Também é feita uma abordagem do funcionamento do sistema através de um estudo de caso.

3.3.1 Técnicas e ferramentas utilizadas

Nesta seção são apresentadas técnicas e ferramentas utilizadas no desenvolvimento do sistema.

3.3.1.1 Smarty

Segundo Fachini (2005), Smarty é um sistema de *templates* para PHP. Mais especificamente, ele fornece uma maneira fácil de controlar a separação da aplicação lógica e o conteúdo de sua apresentação.

Com sua utilização, é possível separar as regras de negócio das regras que definem a apresentação. Isso proporciona maior reusabilidade dos códigos. De forma prática, é possível que exista uma alteração na regra-de-negócio sem que isso interfira no layout da página. O contrário também é válido.

Um dos aspectos únicos do Smarty é seu sistema de compilação de *templates*. O Smarty lê os arquivos de *templates* e cria scripts PHP a partir deles. Uma vez criados, eles são executados sem a necessidade de uma outra compilação do *template*. (SMARTY TEMPLATE ENGINE, 2005).

No quadro 11 apresenta-se um trecho de código da implementação do aplicativo web escrito em Smarty. As *tags* que delimitam o código-fonte do Smarty são por padrão “{}”, mas podem ser trocadas. Esse mesmo trecho é apresentado no quadro 12 utilizando diretamente o PHP.

```
<select id="combo_usuarios"
        name="Lista de Clientes"
        class="input_padrao"
        onchange="CarregaUsuario(this.value);"
        {if $UsuarioCliente}DISABLED{/if} />
  {if $Usuario->get_usu_tipusu() ne "C"}
    {html_options values="" output="-- Novo --"}
  {/if}
  {foreach from=$ListaUsuarios item=itemusuario}
    {html_options values=$itemusuario->get_usu_e_mail()
                  selected=$UsuarioSelecionado
                  output=$itemusuario->get_usu_e_mail()}
  {/foreach}
</select>
```

Quadro 11 – Exemplo de código-fonte utilizando Smarty.

```
<select id="combo_usuarios"
        name="Lista de Clientes"
        class="input_padrao"
        onchange="CarregaUsuario(this.value);"
        <?PHP if ($UsuarioCliente) { echo 'DISABLED'; } ?> />
  <?PHP
  if ($Usuario->get_usu_tipusu() != "C" ) {
    echo '<options values="">-- Novo --</option>';
  }
  else {
     foreach from=$ListaUsuarios item=itemusuario {
        echo '<options values="'.$itemusuario->get_usu_e_mail().
                    '" selected='.$UsuarioSelecionado.'>'.
                       $itemusuario->get_usu_e_mail().'</option>';
     }
  }
  ?>
</select>
```

Quadro 12 – Exemplo de código-fonte utilizando PHP.

3.3.1.2 WAMP5

Windows Apache MySQL PHP 5 (WAMP5) é um software francês publicado sob a *General Public License* (GNU) criado por Romain Bourdon. É parte do projeto WAMPSERVER que é gerenciado pelo PHP *Team* (WAMP5, 2007).

É usado para instalar rapidamente no computador os softwares PHP, MySQL e Apache, disponibilizando suporte ao uso de scripts PHP localmente no Windows. Existem

vários *add-ons* para download no site oficial, como por exemplo, o *add-on* para ativar o uso de PHP4 ao invés do PHP5 (WIKIPEDIA, 2007b).

Segundo WAMP5 (2007), o WAMP5 é compatível com os sistemas operacionais Windows XP, NT, 2000 e 2003. Sua instalação é compacta, ou seja, todos os arquivos são movidos para dentro do diretório de instalação. Com exceção do arquivo “mywamp.ini” que fica na pasta do Windows. Todos os serviços instalados pelo WAMP5 possuem nomes próprios, ou seja, não interferem nos serviços que já estejam instalados no computador. Conforme mostrado na figura 20 que é a tela capturada do programa Serviços do Sistema Operacional onde o WAMP5 foi instalado.

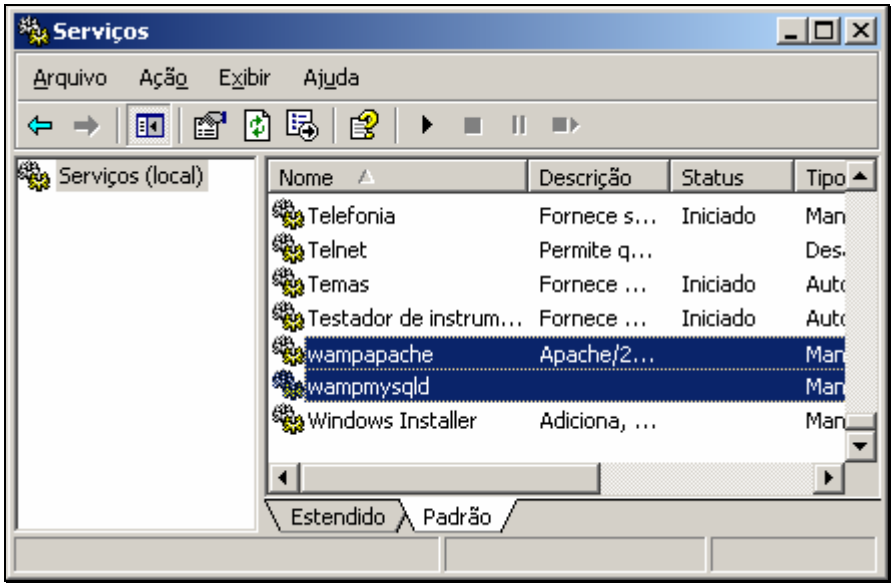


Figura 20: Serviços instalados pelo WAMP5.

Os softwares instalados pela versão WAMP5 utilizada no desenvolvimento da aplicação do presente trabalho foram:

a) Apache 2.2.4;

b) PHP 5.2.1;

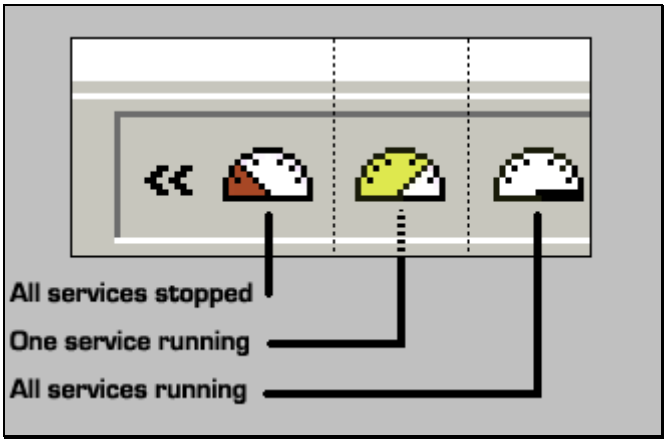
c) MySQL 5.0.27;

d) PHPmyadmin 2.9.2;

e) SQLitemanager 1.2.0;

f) WAMP *server service manager*.

Após a instalação, o *status* dos serviços podem ser visualizados através do ícone de bandeja do WAMP *server service manager*, conforme figura 21. E todos os recursos disponíveis pelo WAMP5 podem ser gerenciados a partir dessa ferramenta, conforme figura 22 que foi capturada no computador em que o WAMP5 foi instalado.



Fonte: WAMPSERVER, 2007.

Figura 21: Status dos serviços.

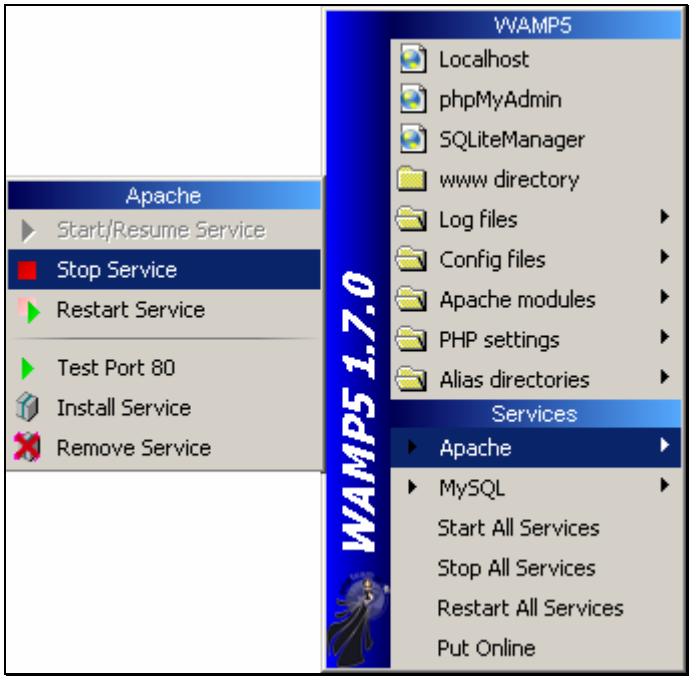


Figura 22: Recursos do WAMP5.

3.3.1.3 Dreamweaver 8

Macromedia Dreamweaver 8 é um editor profissional de HTML para desenhar, codificar e desenvolver web sites, páginas web, e aplicações web. A ferramenta proporciona duas visões distintas do código que está sendo desenvolvido sendo a primeira a do código-fonte e segunda, a do resultado visual do código-fonte, permitindo ao usuário programar em ambas (DREAMWEAVER, 2005).

3.3.1.4 Flash MX

Shockwave Flash, ou simplesmente Flash, é uma ferramenta de autoria e edição de imagens vetoriais com animação, som e interatividade. Baseada em imagens vetoriais, possibilita a criação de efeitos avançados em arquivos bastante pequenos (PINTO, 2000).

Segundo Pinto (2000), ao contrário dos *bitmaps* que são formados por *pixels*, as imagens vetoriais são aquelas criadas por cálculos matemáticos executados pelo computador. Isto significa que os arquivos que contêm essas imagens armazenam somente as fórmulas matemáticas que representam formas, curvas e cores, portanto são muito pequenos.

Flash inclui muitos recursos que o tornam uma ferramenta versátil, porém de fácil uso. São eles componentes visuais do tipo *drag-and-drop*, efeitos especiais que podem ser associados a objetos, e ainda a possibilidade de programação em ActionScript.

ActionScript é a linguagem que se usa para dar maior interatividade às aplicações Flash. Seus scripts permitem que sejam executadas ações dentro do ambiente Macromedia Flash Player, ou seja, na máquina de quem estiver rodando o arquivo Flash compilado.

3.3.1.5 PGAdmin III

PGAdmin III é um amplo gerenciador de base de dados PostgreSQL para sistemas operacionais Linux e Windows. É gratuito e seu projeto é gerenciado pelo pgAdmin *Development Team*. Suporta as versões do PostgreSQL superiores à 7.3 (PGADMIN, 2007).

3.3.1.6 Upload assíncrono

Para se fazer *upload* assíncrono de arquivos foi utilizada uma biblioteca JavaScript. Seu funcionamento consiste em criar um iframe temporário invisível para submeter o arquivo para o servidor (MICOX, 2007).

3.3.2 Operacionalidade da implementação

A operacionalidade da implementação será demonstrada através da simulação de um caso de utilização do sistema. A página de login deve ser implantada no web site da empresa que quer ter seus clientes criando propostas, conforme figura 23.

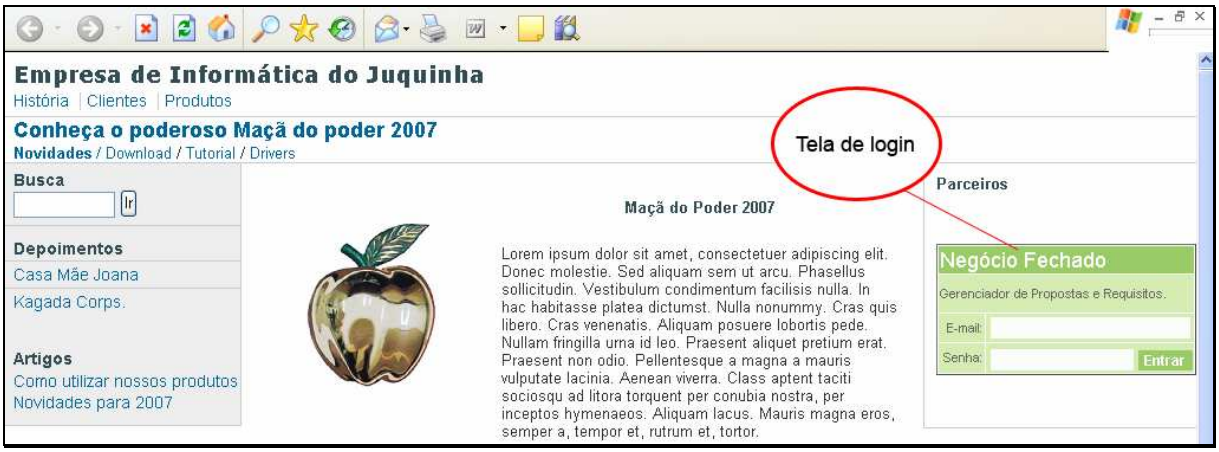


Figura 23: Tela de login implantada.

Para essa empresa será criado, através da intervenção de um *DBA*, um usuário administrador. Ele poderá se conectar ao sistema se deparando com a tela ilustrada pela figura 24. No quadro azul superior aparece o usuário que efetuou o login e um botão para que o usuário efetue logoff, encerrando a sessão. Além disso, existe um botão *home* para que o usuário volte para essa tela, caso ele esteja em outra; um botão ajuda que trás algumas informações úteis sobre o funcionamento do sistema. Na parte superior central aparece o nome do sistema. Na parte esquerda aparece o menu com suas duas opções: Cadastro de usuário e Acompanhamento de propostas.

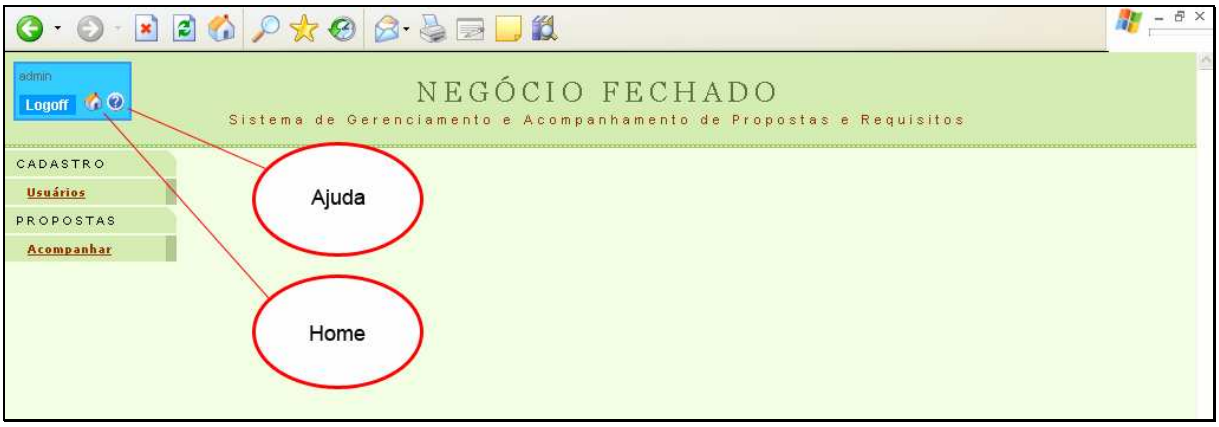


Figura 24: Tela principal.

O próximo passo para o administrador é cadastrar seus analistas. Isso se faz na tela de cadastro de usuários. Ao entrar nessa tela, por padrão, aparecem as informações do próprio usuário. Para que se cadastre um novo usuário é preciso escolher a opção “--Novo--“ na “Lista de Usuários”, conforme a figura 25. Após o preenchimento dos campos é necessário que o usuário clique no botão “Salvar”. Caso o usuário queira abandonar as alterações feitas naquele registro, pode-se clicar em “Cancelar”. Da mesma forma, os analistas podem cadastrar seus clientes.

Figura 25: Tela de cadastro de usuários.

A partir desse ponto o cliente que assume o papel principal. Cabe a ele entrar no sistema e cadastrar suas propostas. Isso pode ser feito entrando na tela de Acompanhamento de Propostas e clicando no botão “Incluir”, conforme figura 26.

Figura 26: Tela de acompanhamento de propostas vazia.

Na tela de inclusão de proposta, ilustrada na figura 27, o usuário deve informar um título e uma descrição, salvando o que foi informado através do botão “Salvar”. Caso o usuário deseje voltar para a tela de acompanhamento de propostas sem salvar as alterações

por ele feitas deve clicar em “Voltar”. É possível ainda anexar arquivos clicando no botão “Anexar”. Ao escolher essa opção o sistema salva as alterações feitas.

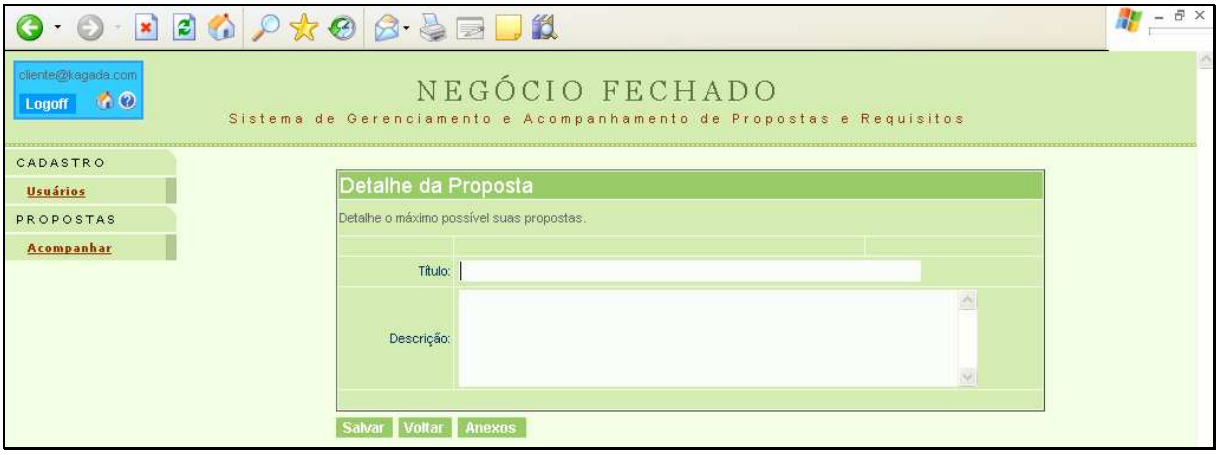


Figura 27: Tela de inclusão de propostas.

Na tela de anexos da proposta, demonstrada pela figura 28, o usuário pode fazer *upload* de arquivos compartilhando-os com o analista. Para adicionar é necessário clicar no botão “Mais”, fazendo com que aparece uma nova linha contendo um botão “Menos”, que serve para remover o arquivo, um botão “Procurar”, que abre uma tela de localização, e um botão “Upload” que envia o arquivo para o servidor.

Figura 28: Tela de anexos.

Uma vez criada a proposta pelo cliente, cabe ao analista observar na tela de acompanhamento, na coluna “Estado”, que a proposta está “Aberta”. Neste ponto ele pode acessar os detalhes da proposta clicando no hyperlink que existe na coluna “Título”. Como apresentado no figura 29.

Figura 29: Tela de acompanhamento com uma proposta.

Acessando a proposta, o analista deve inserir requisitos funcionais e não funcionais. Da mesma maneira que a tela de anexos, clicando no botão “Mais” aparece uma nova linha. Entretanto, nesta tela a linha é composta de um botão “Menos” para remoção do requisito, e demais campos para informar os dados do requisito. Como apresentado na figura 30 a seguir, todo requisito incluído pelo analista fica com o estado de “Analisado”. Se todos os requisitos estiverem “Analisados” a proposta também fica como “Analisada”. O analista pode anexar arquivos acessando a tela de anexos e criar um diagrama de casos de uso na tela apropriada, que é acessada pelo botão “Diagrama”.

Figura 30: Tela de detalhes da proposta para inclusão de requisitos.

Na tela de diagramas o analista pode desenhar um diagrama de casos de uso. Para isso basta clicar em um dos botões que indicam os componentes disponíveis (ator, balão simples e balão tracejado) e clicar no cenário, conforme indicado na figura 31. O balão tracejado não é um conceito próprio da UML. Ele foi criado nesta aplicação para indicar os casos de uso cuja ação não provém diretamente de um ator. A linha para interação necessita que depois de selecionada, o analista clique em dois objetos (origem e destino). É possível selecionar um componente criado e apagá-lo clicando no ícone “Lixeira”. Para apagar uma linha é necessário apenas remover um dos componentes que faz sua ligação. Clicando nos atores e balões é possível alterar seus textos. Para salvar, basta clicar em “Salvar”. Para cancelar as alterações é só clicar em “Cancelar”. Para limpar a tela removendo todos os componentes criados basta clicar em “Limpar”. A interação entre Flash e aplicação é feita através de uma classe chamada `LoadVars`, que quando instanciada permite que se envie e receba dados diretamente de uma página PHP.

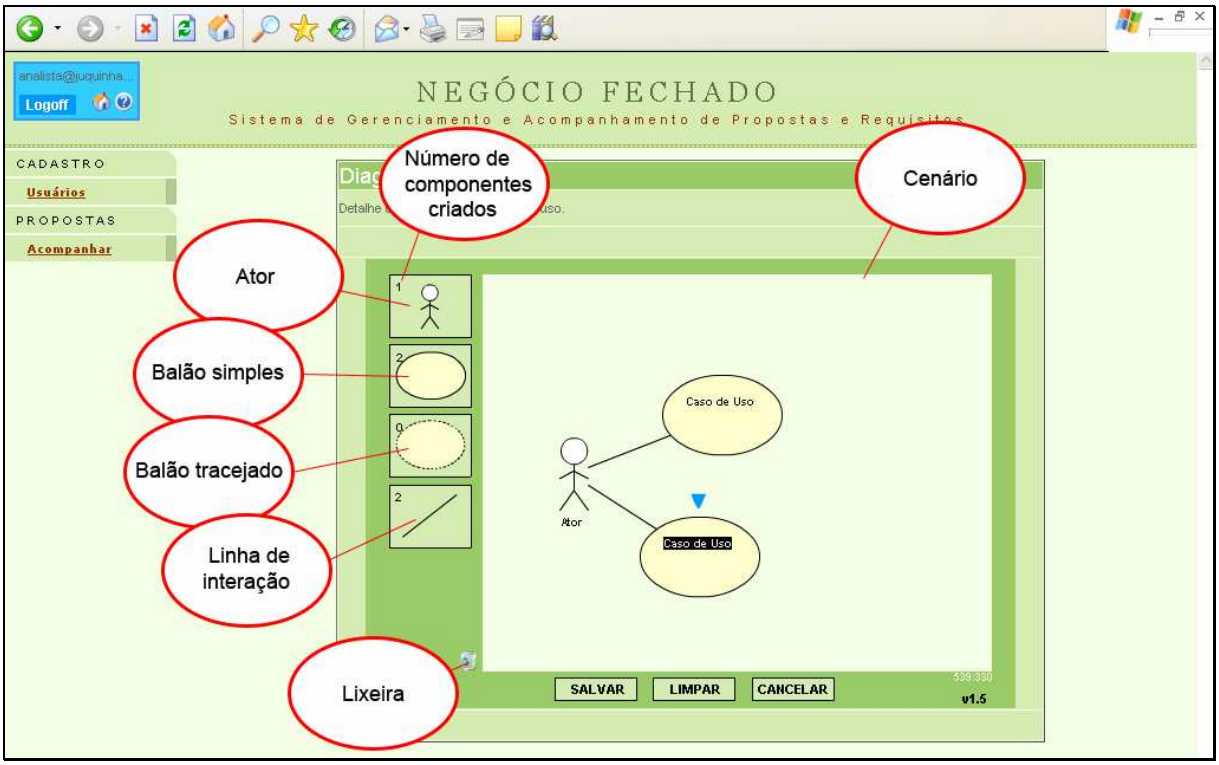


Figura 31: Tela para desenhar diagramas.

O cliente deve identificar que sua proposta está “Analisada” observando a tela de acompanhamento de propostas, conforme figura 32. Da mesma forma que o analista, o cliente deve acessar a proposta através do *hyperlink*.

Figura 32: Tela de acompanhamento de propostas com proposta analisada.

Nesta etapa, na tela de detalhes cabe ao cliente aprovar ou rejeitar os requisitos. Observa-se na figura 33 este processo. Se o cliente não estiver satisfeito com a proposta ele pode cancelá-la através do botão “Cancelar Proposta”. Do ponto de vista do cliente essa

operação é irreversível. Caso ele aceite todos os requisitos ele pode clicar em “Fechar Negócio”.

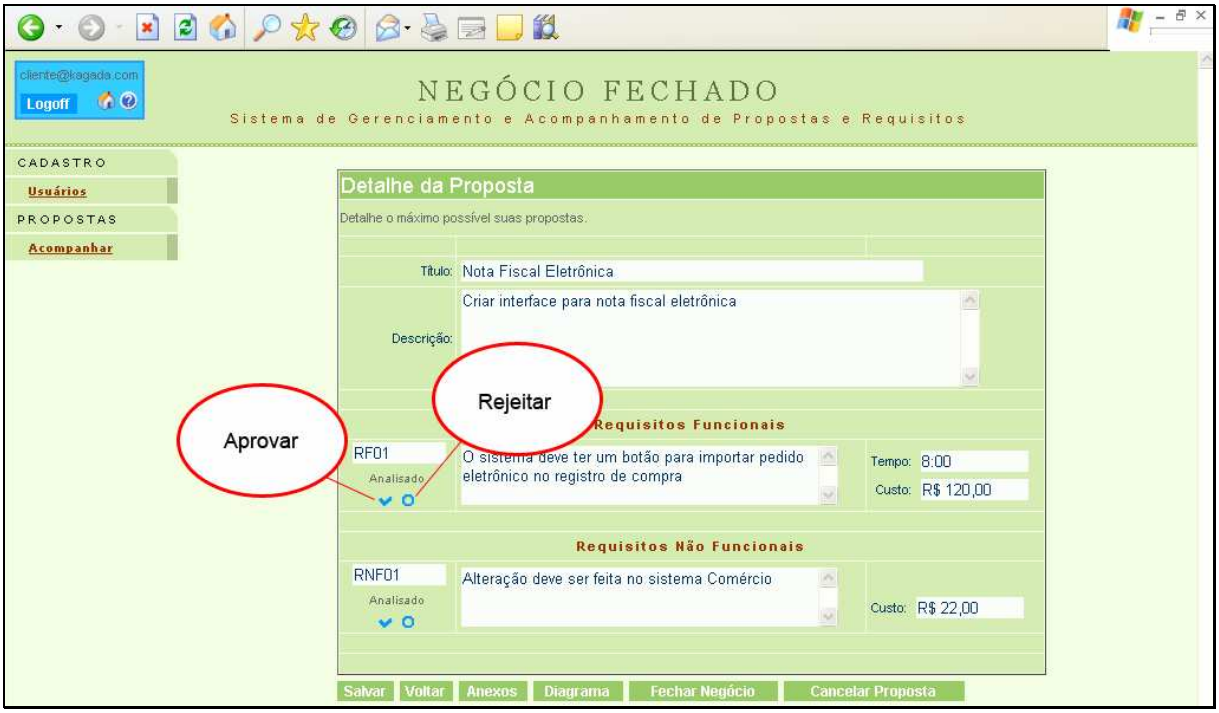


Figura 33: Tela de acompanhamento de propostas com proposta analisada.

Na tela de Aprovação de Proposta, conforme figura 34, acessada pelo botão “Fechar Negócio” da tela de detalhes da proposta, o cliente tem a possibilidade de clicar em “Aprovar”, fechando negócio com a empresa que lhe presta serviço, ou clicar em “Voltar”, voltando para a tela de detalhes da proposta. A operação de aprovar proposta é irreversível, é nesse momento que o negócio é fechado. E a proposta se torna apenas possível de ser lida.

Figura 34: Tela de acompanhamento de propostas com proposta analisada.

3.4 RESULTADOS E DISCUSSÃO

Dentro do que se objetivou desenvolver, nem todas as funcionalidades do PHP5 foram implementadas na ferramenta, mas as que foram implementadas se mostraram bastante úteis. Utilizando essas novas possibilidades, as alterações no PHP5, mostraram-se menos “abertas” para erros de desenvolvedores. Um exemplo disso seria a declaração *private* para variáveis, que não existia nas versões anteriores. Ou seja, uma vez declarada uma variável dentro de uma classe nada impediria que ela fosse acessada de qualquer lugar. No PHP5, se ela é *private*, apenas os métodos daquela classe podem acessá-la, qualquer outra tentativa resulta em erro.

Com relação ao trabalho apresentado por Fachini (2005), o uso de PHP5 possibilitou que a modelagem das classes fosse mais bem desenvolvida, tornando o modelo orientado a objetos do trabalho mais normalizado.

O uso de AJAX se mostrou eficaz em quase todas as áreas do trabalho, permitindo o envio de formulários inteiros, a carga de *combos* e abertura de telas sem que se tenha que

carregar a tela completamente. O único processo que se tentou fazer com AJAX que não se obteve sucesso foi o envio de arquivos. Por uma questão de segurança, os navegadores não permitem, por padrão, deixar que o JavaScript acesse recursos externos, impossibilitando que scripts AJAX encontrem o arquivo que se deseja enviar.

Para que se conseguisse enviar arquivos utilizou-se a técnica de requisições assíncronas através de componentes iframes. Conforme W3C (2006), existe uma interface chamada File, que serve para gerenciar *upload* de múltiplos arquivos, que poderia também resolver essa carência do AJAX.

O uso de Flash se mostrou bastante eficiente para problemas de desenho. Devido à sua portabilidade, códigos disponíveis na Internet, facilidade para se desenhar elementos e criar as interações. Um fator que pesa no uso de Flash é que a mesma não é uma ferramenta gratuita. Em comparação com a linguagem SVG utilizada por Gois (2006), o ActionScript se mostrou mais popular e com maior possibilidade de se encontrar fontes disponíveis na Internet, o que é um fator decisivo para se resolver problemas mais rapidamente.

Devido à base de dados não ser muito extensa, nem os recursos utilizados para manutenção de dados serem avançados, a utilização do banco de dados PostgreSQL foi muito relevante para o resultado final da aplicação. Com esse trabalho pôde-se observar apenas a facilidade de instalação no ambiente Windows, bem como a facilidade de manutenção utilizando a ferramenta PGAdmin III.

Ao final do desenvolvimento observou-se que a modelagem de dados do sistema poderia ser melhor. As tabelas de planilha, documento e contrato não precisavam estar normalizadas, já que elas possuem os mesmo atributos e métodos. Os comandos ANSI SQL poderiam estar descritos diretamente nas classes de consulta abstratas, já que é possível sobrescrever métodos no PHP5.

4 CONCLUSÕES

Como foi comprovado neste presente trabalho, é possível obter um resultado prático desenvolvendo sistemas web em PHP utilizando AJAX e Flash. Todas essas tecnologias estão amplamente difundidas e é possível encontrar milhares de códigos prontos tanto em sites brasileiros quanto em estrangeiros.

Com a utilização dessas tecnologias obteve-se êxito na implementação de um aplicativo que fizesse a interação entre *stakeholders* em relação às propostas de desenvolvimento. Também com sucesso permitiu-se que o analista relacionasse os requisitos, desenhasse diagrama de casos de uso na proposta e interagisse com o cliente a cada alteração nos requisitos.

O resultado final é um sistema web capaz de comportar a interação entre cliente e analista para que de um lado existam novas solicitações e de outro existam definições de custo e tempo, possibilitando a otimização na implementação de pequenas customizações.

Em termos de limitações, o sistema não possui relatórios nem gráficos que permitam ao cliente gerenciar custos e tempo. As propostas devem ser bem pontuais, ou seja, customizações do dia-a-dia, já que é apenas possível informar requisitos e um diagrama de casos de uso.

4.1 EXTENSÕES

Entre possíveis extensões para este trabalho, destacam-se:

a) utilização de outros bancos de dados, como por exemplo, MySQL, Oracle ou SQLServer, apenas criando as classes que contém o código SQL, já que o presente trabalho utiliza o conceito DAO;

b) criar funcionalidades novas como um novo tipo de diagrama ou novos recursos para o diagrama de casos de uso;

c) criar relatórios e gráficos com totalizadores de preço e tempo;

d) ampliar esse sistema para que o cliente pudesse acompanhar a execução de suas proposta;

e) implementar possibilidade de propostas pré-definidas para que o usuário não tenha que esperar a interação do analista;

f) edição de cenários através de mais uma tela na proposta, permitindo que o analista crie cenários relacionando os casos de uso e que tenham *constraints*, fluxo principal e fluxo de exceção;

g) versionamento das informações: caso exista alguma alteração da proposta por alguma das partes, essa alteração deve ser visível para a outra parte quando ela for reprocessar a proposta. Como sugestão, essa visualização da alteração poderia ser na forma de *log* ou indicação visual campo a campo.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Rondinely S. de. **AJAX:** O Objeto XMLHttpRequest. [S.l.], 2006. Disponível em: <http://www.hospedia.com.br/artigos/8/ajax/1/ajax_-_o_objeto_xmlhttprequest_-_parte_4.html>. Acesso em: 15 jun. 2007.

AMORIM Adeilton Queiroz. SILVA JR. Adelino. Ferreira da. **FISIOSYSTEM:** Sistema de gerenciamento da clínica Fisio Center. 2005. 60 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro Universitário de João Pessoa, João Pessoa.

BARBIERI, Carlos. **BI – Business Intelligence:** Modelagem & Tecnologia. Rio de Janeiro: Axcel Books do Brasil Editora, 2001. ISBN 85-7323-148-3.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML.** Rio de Janeiro: Campus, 2003.

BIAZUS, Diogo de Oliveira. **Introdução e Histórico.** [S.l.], 2003a. Disponível em: <http://www.postgresql.org.br/Introdu%C3%A7%C3%A3o_e_hist%C3%B3rico>. Acesso em: 15 jun. 2007.

_____. **PostgreSQL 8.0:** As Boas Novas do Elefante. SQL Magazine. DevMedia Group, 2003b. ISSN 1677918-5.

COCKBURN, Alistair. **Escrevendo Casos de Uso Eficazes:** Um guia prático para desenvolvedores de software. Tradução Roberto Vedoato. Porto Alegre: Bookman, 2005. ISBN 85-363-0457-X.

CRIARWEB.COM. **Breve História do PHP.** [S.l.], 2007. Disponível em: <http://www.criarweb.com/artigos/71.php >. Acesso em: 10 fev. 2007.

DREAMWEAVER. Macromedia Dreamweaver: help. Version 8. [S.l.], 2005. Documento eletrônico disponibilizado com o software Dreamweaver 8.

FACHINI, Paulo Luiz. **Sistema de Informação para Acompanhamento de Chamados e Workflow Via Web.** 2005. 65 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FERNANDES, Aguinaldo Aragon. TEIXEIRA, Descartes de Souza. **Fábrica de Software:** Implantação e Gestão de Operações. São Paulo: Atlas, 2004. ISBN 85-224-3690-8.

FERREIRA, Elcio. **Ajax para quem só ouviu falar.** [S.l.], 2007. Disponível em: <http://www.tableless.com.br/artigos/ajaxdemo/>. Acesso em: 8 fev. 2007.

FOLHAONLINE. **Entenda o que é a Web 2.0**. [S.l.], 2006. Disponível em: <http://www1.folha.uol.com.br/folha/informatica/ult124u20173.shtml>. Acesso em: 9 fev. 2007.

GARRETT, Jesse J. **Ajax:** A New Approach to Web Applications. [S.l.], 2005. Disponível em: <http://www.adaptivepath.com/publications/essays/archives/000385.php>. Acesso em: 8 fev. 2007.

GIELOW, Sandra Carla. **Ferramenta de Suporte ao Cálculo dos Use Case Points.** 2003. 70 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

GOIS, Marcus Vinícius Silva. **Ajax na Construção de uma Aplicação Web para Monitoramento de Ambientes em Plantas 2D.** 2006. 71 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

GUTMANS, Andi. BAKKEN Stig. RETHANS Derick. **PHP5 Power Programming.** São Paulo: Prentice Hall, 2005. ISBN 11-13-147149-X

INFOEXAME. **67 Serviços Imperdíveis da Web 2.0.** Maio, 2007.

LIMA, Adilson da Silva. **UML 2.0:** Do requisito à solução. São Paulo: Érica, 2005.

MAGELA, Rogério. **Engenharia de Software Aplicada:** Fundamentos. Rio de Janeiro: Alta Books, 2006.

MARISCO, Nelson. PHILIPS Jürgen. PEREIRA Humberto. Rafael. **Protótipo de Mapa Para Web Interativo:** Uma Abordagem. [S.l.], 2004. Disponível em: <http://www2.prudente.unesp.br/rbc/_pdf_56_2004/56_1_08.pdf>. Acesso em: 11 fev. 2007.

MICOX. **Upload Assíncrono:** Iframe como AJAX. [S.l.], 2007. Disponível em: <http://elmicox.blogspot.com/2007/03/upload-assncrono-iframe-como-ajax-1.html>. Acesso em: 18 jun. 2007.

O'REILLY, Tim. **O Que é Web 2.0:** Padrões de design e modelos de negócios para a nova geração de software. Tradução Miriam Medeiros. [S.l.], 2005. Disponível em: <http://pressdelete.files.wordpress.com/2006/12/o-que-e-web-20.pdf>. Acesso em: 15 jun. 2007.

OSLEI, Daniel. **Introdução ao PostgreSQL.** [S.l.], 2004. Disponível em: <http://www.crieseuwebsite.com/tutoriais/abretutorial.php?tutorial=118>. Acesso em: 9 fev. 2007.

PFLEEGER, Shari Lawrence. **Engenharia de Software:** Teoria e Prática. São Paulo: Prentice Hall, 2004. ISBN 85-87918-31-1.

PGADMIN. PGAdmin: help. Version 3. [S.l.], 2007. Documento eletrônico disponibilizado com o software PGAdmin III.

PHP.NET. **Manual do PHP.** [S.l.], 2007. Disponível em: <http://www.php.net/manual/pt_BR/>. Acesso em: 9 fev. 2007.

PINTO, Marcos José. **Flash 5:** A Nova Geração em Sites Interativos. São Paulo: Érica, 2000.

PROJECTO SIEP. **Tecnologias Aplicadas na Implementação.** Portugal, Braga: Universidade do Minho, 2004. Disponível em: <http://www.di.uminho.pt/~gepl/SIEP/tecnologias.htm>. Acesso em: 10 fev. 2007.

RAZENTE, Humberto. BARIONI, Maria Camila Nardini TRAINA, Agma. Juci Machado TRAINA JR, Caetano. **Recuperação de Imagens Médicas por Conteúdo em um Sistema de Gerenciamento de Banco de Dados de Código Livre.** X Congresso Brasileiro de Informática em Saúde. USP, 2006. Disponível em: <http://gbdi.icmc.usp.br/publicacoes/arquivos/2006_CBIS_Razente.pdf>. Acesso em: 11 fev. 2007.

REINERT, Roberto. **Sistema de Workflow para Modelagem e Execução de Processos de Software.** 2006. 75 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

ROCHA, Ana Regina Cavalcanti da. MALDONADO, José Carlos. WEBER, Kival Chaves. **Qualidade de Software:** Teoria e Prática. São Paulo: Prentice Hall, 2001. ISBN 85-87918-54-0.

SMARTY TEMPLATE ENGINE. **Documentation.** [S.l.], 2005. Disponível em: <http://smarty.php.net>. Acesso em 09 jun. 2007.

TIINSIDE. **Nasce a Geração C.** Ano 3. 21ª Edição. Jan./Fev., 2007.

W3C. **File Upload.** [S.l.], 2006. Disponível em: <http://www.w3.org/TR/2006/WD-file-upload-20061018/>. Acesso em 15 jun. 2007.

____. **The XMLHttpRequest Object.** [S.l.], 2007. Disponível em: <http://www.w3.org/TR/2007/WD-XMLHttpRequest-20070227/>. Acesso em 15 jun. 2007.

WAMP5. WAMP5: help. Version 1.7.0. [S.l.], 2007. Documento eletrônico disponibilizado com o software WAMP5.

WAMPSERVER. **Servers Web Four Windows.** [S.l.], 2007. Disponível em: <http://www.wampserver.com/en/presentation.php>. Acesso em 09 jun. 2007.

WIKIPEDIA. **PHP.** [S.l.], 2007a. Disponível em: <http://pt.wikipedia.org/wiki/Php>. Acesso em: 10 fev. 2007.

_____. **WAMP5.** [S.l.], 2007b. Disponível em: <http://pt.wikipedia.org/wiki/wamp5>. Acesso em: 10 jun. 2007.