UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

VITOR CERVELIN JUNIOR

GERADOR DE FORMULÁRIOS HTML A PARTIR DE SCRIPTS SQL: ESTUDO DE CASO UTILIZANDO PHP

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2011

VITOR CERVELIN JUNIOR

GERADOR DE FORMULÁRIOS HTML A PARTIR DE SCRIPTS SQL: ESTUDO DE CASO UTILIZANDO PHP

Trabalho de diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – CSTADS - da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Msc. Fernando Schütz

MEDIANEIRA

2011

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Coordenação do Curso Superior de Tecnologia em
Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

Gerador de Formulários HTML a partir de Scripts SQL:
Estudo de Caso Utilizando PHP.

Por

Vitor Cervelin Junior

Este Trabalho de Diplomação (TD) foi apresentado às 13:10 h do dia 17 de junho de 2011 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, *Campus* Medianeira. O candidato foi argüido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. M.Sc. Fernando Schütz UTFPR – *Campus* Medianeira (Orientador)	Prof. M.Sc. Claudio Leones Bazzi UTFPR – *Campus* Medianeira (Convidado)
Prof. M.Sc. Romualdo Rubens de Freitas UTFPR – *Campus* Medianeira (Convidado)	Prof. M.Eg. Juliano Rodrigo Lamb UTFPR – *Campus* Medianeira (Responsável pelas atividades de TCC)

A folha de aprovação assinada encontra-se na Coordenação do Curso.

“Informação é poder, porém se tens tal domínio e não o divulgas, torna-te responsável pela ignorância alheia.”

Ivan Teorilang

RESUMO

Este trabalho apresenta um estudo sobre o desenvolvimento de um framework para desenvolvimento *WEB* ágil. Faz uma explanação sobre tecnologias utilizadas no meio de desenvolvimento *WEB*, mostrando breve histórico e conceitos sobre PHP, CSS, HTML, Ajax, programação orientada a objetos, recursos de desenvolvimento e o método de *scaffold*. Mostra ao final um exemplo de aplicação para testes desenvolvidas a partir de tecnologias já citadas, que tem como objetivo expor de forma prática os conceitos envolvidos no estudo, principalmente o método de *scaffold*ing de aplicação, para criação de *CRUD* com estilização visual amigável.

Palavras-chave: PHP, *scaffold*, desenvolvimento WEB

RESUMO EM LINGUA ESTRANGEIRA

This work presents a study on developing a framework for agile *WEB* development. Makes an explanation of technologies used in *WEB* development through, showing a brief history and concepts of PHP, CSS, HTML, Ajax, Object-oriented programming, resource development and the method of *Scaffold*. Shows an example of the end application testing developed from technologies already mentioned, which aims to expose a practical concepts involved in the study, especially the method of application of *scaffold*ing to create *CRUD* with visual stylization friendly.

Key-words: PHP, *scaffold*, WEB development.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	*Application Programming Interface*
COBOL	*Common Business Oriented Language*
CRUD	*Create Replace Update Delete*
CSS	*Cascading Style Sheets*
DHTML	*Dynamic HTML*
DNS	*Domain Name System*
ERP	*Enterprise Resource Planning*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
IMAP	*Internet Message Access Protocol*
JS	*JavaScript*
LDAP	*Lightweight Directory Access Protocol*
MVC	*Model View Controller*
NNTP	*Network News Transfer Protocol*
OOP	*Object-Oriented Programming*
PDF	Portable Document Format
PHP	*Personal Hypertext Preprocessor*
POP3	*Post Office Protocol*
RIA	*Rich Internet Application*
SGBD	*Sistema de Gerenciamento de Banco de Dados*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*

UTFPR	*Universidade Tecnológica Federal do Paraná*
W3C	*World Wide Web Consortium*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos desenvolvedores *WEB* têm buscado cada vez mais ferramentas e *softwares* para aumento de produtividade em ambiente de desenvolvimento. Para isso surgem *framewoks* cada vez mais completos e robustos para atender a demanda dos profissionais. Com um mecanismo de geração de formulários HTML a partir de scripts SQL os testes de aplicativos *WEB* tendem a ficar cada vez mais intuitivos, oferecendo ao desenvolvedor uma aplicação de testes mais próximas da etapa final de desenvolvimento de seus aplicativos.

O modelo clássico de aplicação *WEB* trabalha da seguinte forma: a maioria das ações do usuário na interface dispara uma solicitação HTTP para o servidor *WEB*. O servidor processa uma página, recuperando dados, realizando cálculos, conversando com vários sistemas lega dos, e então retorna uma página HTML para o cliente. É um modelo adaptado do uso original da *WEB* como um agente de hipertexto, porém o que faz a *WEB* boa para hipertexto não necessariamente a faz boa para aplicações de *software*. A Figura 1 apresenta o funcionamento da arquitetura *WEB*.

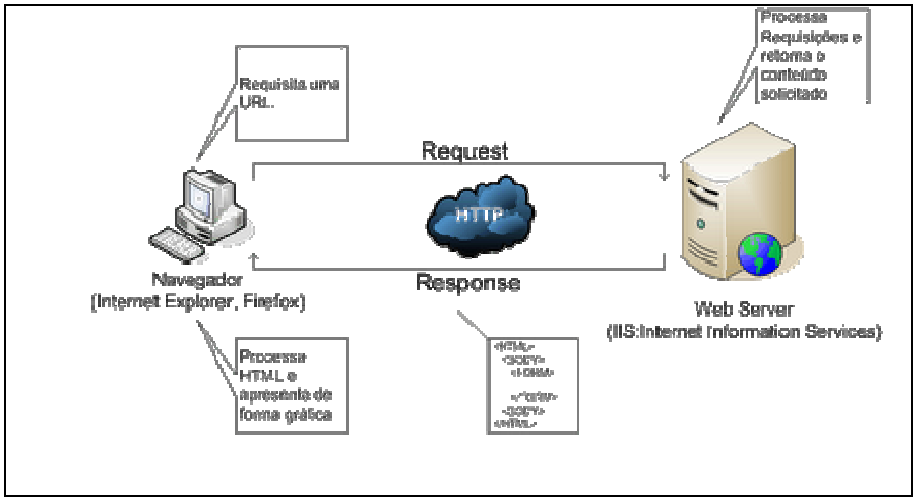

Figura 1 – Arquitetura WEB.
Fonte: Blog Cliente *WEB (*2008).

Com a popularização de sistemas que funcionam inteiramente na *WEB* e também com o aumento da velocidade das conexões banda larga, o problema da espera pelo envio e retorno da página inteira se tornou muito mais evidente para o usuário.

Para resolver esse problema surgiu o *Ajax (JavaScript e XML Assíncrono)*. As principais vantagens das aplicações que utilizam *Ajax* para determinadas requisições é que os dados trafegados pela rede são reduzidos e o usuário não precisa aguardar a página ser recarregada a cada interação com o servidor.

Em vez de carregar uma página *WEB*, no início da sessão, o navegador carrega uma ferramenta *Ajax*, escrita em *JavaScript* e geralmente colocada sobre um frame escondido. Esta ferramenta é responsável por renderizar a interface que o usuário vê e pela comunicação com o servidor em prol do usuário. A ferramenta *Ajax* permite que a interação do usuário com a aplicação aconteça assincronamente, independente da comunicação com o serviço, então o usuário nunca ficará de frente com uma janela branca do browser e um ícone de ampulheta, esperando pelo servidor para fazer algo.

Nos dias atuais o desenvolvimento de aplicações *WEB* utilizando *frameworks* se expandiu de tal forma que o uso deles para construção de sistemas voltados para *WEB* é uma necessidade comprovada pela comunidade de desenvolvedores e pelo meio corporativo. Projetos de pequeno e grande porte, nacionais e internacionais, muitos são os casos atribuídos aos modernos e ágeis *frameworks* para desenvolvimento *WEB*. Entre os casos de sucesso estão o *Twitter* utilizando o *Rails, e o SourceForge* utilizando *o Turbo Gears 2.*

O padrão *MVC* (*model, view, controller*) funciona muito bem para aplicações *WEB*, e é melhor aproveitado quando explorado por um *framework*. *Frameworks* implementam o padrão *MVC* de forma realmente padronizada, o explorando ao máximo (GILMORE, 2008).

Na área de desenvolvimento *WEB* existem vários *frameworks open-source* disponíveis para *download*, com suporte a *MVC*, *scaffold,* e vários outros recursos. Mas vários projetos desenvolvidos não necessitam necessariamente de um *framework* para sua produção, por serem pequenos projetos com pouquíssimas classes e linhas de códigos a serem escritas. Nessa parte entra um aplicativo *WEB* simples e de fácil utilização onde o desenvolvedor poderá gerar um *CRUD* com folhas de estilo aplicados para seu

software, onde basta apenas inserir algumas linhas de *script* SQL, não necessitando configurar todo um *framework* como *Cake PHP*, *Zend Framework* ou *Code Igniter* para iniciar seu desenvolvimento.

1.1 OBJETIVOS

Este trabalho, que abordará conceitos sobre RIA e programação para *WEB*, tem seus objetivos descritos nas seções que seguem.

1.1.1 Objetivo Geral

Projetar e desenvolver um *software WEB* de testes para desenvolvedores da área, utilizando linguagem de programação PHP.

1.1.2 Objetivos Específicos

- Desenvolver um referencial teórico sobre as tecnologias de desenvolvimento de *software* para *WEB*, técnicas para construção de Aplicações Ricas para Internet (RIA) e persistência de objetos.
- Desenvolver análise e projeto do sistema utilizando linguagem UML.
- Desenvolver um sistema *WEB* com a linguagem de programação PHP, para ser usado por profissionais da área, utilizando as tecnologias Ajax e CSS no desenvolvimento do sistema;
- Elaborar um plano de testes e testar a aplicação com desenvolvedores.

1.2 JUSTIFICATIVA

Atualmente os desenvolvedores *WEB* não contam com nenhum *software* rápido para geração de formulários HTML com estilo visual embutido, para testar suas aplicações *WEB* em ambiente de desenvolvimento. Com o *software* desenvolvido a comunidade de desenvolvedores *WEB* ganhará um aliado no desenvolvimento de *WEBApps*, transformando páginas de testes no mais próximo possível de páginas em ambiente de desenvolvimento.

1.3 DIVISÃO DO TRABALHO

Este trabalho é dividido em três capítulos. O primeiro capítulo faz uma introdução ao desenvolvimento ágil, além *frameworks***,** MVC e Ajax.

O segundo capítulo apresenta a linguagem PHP, Orientação a Objetos, linguagem JavaScript, AJAX, CSS(*Cascading Style Sheets)*, Scaffolding, HTML 5 e o MySql.

O terceiro capítulo apresenta o aplicativo desenvolvido no decorrer do trabalho, onde é apresentado a codificação a análise e os testes realizados.

2 REFERENCIAL TEÓRICO

Este capítulo aborda tópicos relacionados aos estudos realizados no decorrer do trabalho de conclusão de curso.

2.1 PHP

Neste capítulo será apresentada uma breve descrição da linguagem PHP, incluindo história da linguagem, usabilidade e orientação a objetos.

2.1.1 Histórico

A linguagem PHP é a representante do estereótipo de projeto *open source,* criada pra suprir as necessidades do desenvolvedor e aperfeiçoada pra se adequar às necessidades da sua crescente comunidade.

O que distingui o PHP de uma linguagem como *JavaScript*, por exemplo, é que em *JavaScript* o código é executado no lado do cliente e no PHP o código é executado no servidor gerando HTML que é então enviado para o cliente. O cliente receberia os resultados da execução desse script, mas não saberia como é o código fonte. (GILMORE, 2008).

Em junho de 2004 foi lançada a versão 5 do PHP, introduzindo um novo modelo de orientação a objetos, incluindo a reformulação dos construtores e adição de destrutores, visibilidade de acesso, abstração de objeto e interfaces de objetos (MILANI, 2010).

2.1.2 Características Gerais da Linguagem

Segundo Gilmore (2008), o PHP possui algumas características que o conduziram a uma posição de destaque no mercado de desenvolvimento, como a praticidade, poder, possibilidade e preço.

A linguagem PHP foi criada focada em praticidade. Afinal, a intenção original de Lerdorf, criador da linguagem PHP, não era desenvolver uma linguagem toda nova, mas resolver um problema que não tinha solução prontamente disponível. A constante evolução da linguagem ao longo dos anos tem como objetivo aumentar sua utilidade ao usuário. O resultado é uma linguagem que permite ao usuário construir aplicações poderosas mesmo com um mínimo de conhecimento (GILMORE, 2008).

Um exemplo de praticidade da linguagem pode ser visto no Quadro 1, que representa um script PHP completo em apenas uma linha, cujo propósito é exibir a data atual.

```
<?php echo date("j F, Y");?>
```

Quadro 1 – Script que escreve a data na página

Os desenvolvedores PHP têm mais de 180 bibliotecas à disposição, coletivamente contendo mais de 1.000 funções. Além de se conectar com bancos de dados, manipular informações de formulário e criar páginas dinamicamente, o PHP também pode:

- Avaliar uma senha por *guessability* (verificação do nível de segurança da senha) comparando-a a dicionários de linguagem e identificando padrões facilmente quebrados;
- Analisar as *strings* mais complexas usando POSIX e bibliotecas de expressão regular baseadas em Perl;
- Autenticar usuários confrontando credenciais de *login* aramazenadas em arquivos *flat*, bancos de dados e até mesmo no *Active Directory* da Microsoft;
- Comunicar-se por meio de vários protocolos, incluindo LDAP, IMAP, POP3, NNTP e DNS;
- Integrar-se fortemente com uma ampla lista de soluções de processamento de cartão de crédito. (MELONI, 2000).

A capacidade flexível do PHP de *string-parsing* oferece a usuários com diferentes habilidades a oportunidade de, não apenas imediatamente iniciar a execução de operações complexas de string, mas também, rapidamente, portar no PHP programas de funcionalidade similar (como Perl e Python). Além de mais de 85 funções de manipulação de string e formatos de expressão regular baseados em Perl são suportados. O PHP oferece ainda suporte a linguagem procedural e linguagem orientada a objeto. (CAMARGOS, MENEZES; 2008).

PHP é disponibilizado gratuitamente. Desde seu início, PHP tem sido gratuito sem restrições de redistribuição, modificação e uso. Recentemente, *softwares* que se adaptam a essas qualificações de licença aberta têm sido referidos como *software open source*. Algumas características de *softwares open source*:

- Livre de restrições de licença imposta pela maioria dos produtos comerciais; Embora existam algumas discrepâncias entre as variantes de licença, usuários são amplamente liberados para modificar, redistribuir e integrar *software* em outros produtos;
- Desenvolvimento aberto e processo de auditoria; pelo fato do código original ser disponibilizado gratuitamente para qualquer um examinar, problemas com segurança e *bugs* são rapidamente encontrados e consertados. (NIEDERAUER, 2011).

2.1.3 Vantagens

Entre as principais vantagens da utilização da linguagem PHP frente ao ASP, por exemplo, cita-se:

- Linguagem da fácil aprendizado;
- Desempenho e estabilidade excelentes;
- Código aberto;
- Suporte nos principais servidores do mercado;
- Suporta conexão com os principais bancos de dados do mercado;
- É multiplataforma;

- Suporta grande variedade de protocolos;
- Não precisa ser compilado, por ser uma linguagem interpretada.

2.2 ORIENTAÇÃO A OBJETOS

A orientação a objetos é um paradigma que representa toda uma filosofia para a construção de sistemas. Em vez de construir um sistema formado por um conjunto de procedimentos e variáveis nem sempre agrupadas de acordo com o contexto, como se fazia em linguagens estruturadas como COBOL, Clipper e Pascal, na orientação a objetos se utiliza uma ótica mais próxima do mundo real.

O nascimento da programação orientada a objeto representou uma importante mudança de paradigma em estratégia de desenvolvimento, refocalizando a atenção nos dados de uma aplicação em vez da sua lógica. Em outras palavras, OOP muda o foco dos eventos de procedimentos do programa para as entidades da vida real que ele, no fim das contas, modela.

Os conceitos fundamentais de OOP são: encapsulamento, herança e polimorfismo. Juntos, essas três idéias formam a base para o mais poderoso modelo de programação já desenvolvido. (DALL'OGLIO, 2007).

2.3 JAVASCRIPT

JavaScript é uma linguagem de *script* de programação. Não é compilada e sim interpretada, ou seja, pra que ela seja executada, é necessário que o navegador a interprete. Dessa maneira, é fundamental que o navegador seja compatível com a linguagem de *script.*

Chamada inicialmente de *LiveScript*, foi desenvolvida pela Netscape. Posteriormente, esta, em parceria com a SUN, empresa desenvolvedora da linguagem Java, hoje pertencente a ORACLE, criaram a linguagem de *script* baseada no próprio Java, surgindo o *JavaScript*. Portanto, o *Javascript* contém sintaxes e instruções semelhantes às do Java, porém, com uma manipulação muito mais simples e fácil. (SILVA, 2010).

É uma linguagem de marcação bem limitada. O *JavaScript* permite a interação com o HTML, tornando possível a manipulação de vários elementos e a criação de recursos diversos em uma página que o HTML, por si só, é incapaz de realizar.

Com isso, o *JavaScript* permite a execução de instruções de acordo com a intenção do programador. Podem-se inserir comandos para executar algo quando uma página for aberta ou fechada, pode-se criar uma validação de formulário para a verificação do correto preenchimento dele, efetuar contas, exibir mensagens na barra de status, criar pequenas animações, entre outros. (SILVA, 2005).

Como o *JavaScript* é uma linguagem interpretada seus códigos podem estar embutidos no próprio arquivo HTML. Além disso, o *JavaScript* contém menos objetos e comandos que o Java. O Quadro 2 apresenta um exemplo de código *JavaScript.*

```
window.onload = function(){
        var palheta = document.getElementById('palheta');
        var tds = palheta.getElementsByTagName('TD');

        for (var i = 0; tds.length; i++) {
                tds[i].onclick = function() {
                        if( window.getComputedStyle ) {
                          bg = document.defaultView.getComputedStyle(this, null).backgroundColor;
                        } else if( palheta.currentStyle ) {
                          bg = document.getElementById(this.id).currentStyle['backgroundColor'];
                        }
                        document.body.style.backgroundColor = bg;
                }
        }
}
```

Quadro 2 - Código JavaScript para alteração de background de páginas HTML.

2.4 AJAX

É um acrônimo desenvolvido muito recentemente por Jesse James Garret, da *Adaptive Path*, e significa *Asynchronous JavaScript and XML* (JavaScript e XML assíncrono). Porém, o AJAX é muito mais que apenas a junção de JavaScript com XML, é todo um conceito de navegação e atualização de páginas *WEB*. Algumas partes descritas na definição de AJAX não são novas,

as quais muitas vezes foram denominadas de DHTML (HTML Dinâmico) e *Script* Remoto.(GONÇALVES, 2007).

No começo existiam apenas páginas repletas de textos e nada mais, antes de surgirem os *browsers* gráficos. Com o passar do tempo, e a descoberta da *WEB* por mais e mais pessoas, deixando o ambiente acadêmico e passando a ser visto e utilizado por todos, passamos a ter uma forma mais estruturada e bonita de visualização, seja pelo uso de fontes diferentes, cores, imagens, tabelas, seja pela atualização dinâmica das páginas. Neste caso entram as linguagens de programação voltadas para *WEB*, tais como: PHP, ASP e Java, entre outros. (SOARES, 2007).

Atualmente, existem as mais variadas aplicações rodando na *WEB*, desde simples *sites* de notícias ou *blogs*, até portais complexos para negociação entre empresas, ou de empresa com pessoas físicas, chegando aos sistemas de gestão, conhecidos como ERP - *Enterprise Resource Planning*.

Mesmo com toda evolução da *WEB*, a forma que as páginas eram processadas ainda não havia mudado, e se manteve a mesma desde o início. Para que uma página seja processada, é preciso que os dados sejam enviados ao servidor e que ele retorne uma nova página pós-processamento, criando o efeito de *"refresh*" da página. Dessa forma o carregamento das páginas ficava muito lento, pois no pré-carregamento das páginas todo e qualquer dado necessário precisava ser enviado na primeira requisição do servidor para o cliente, ou toda vez que a página tiver de ser carregada.

Para solucionar o problema de desempenho nas páginas *WEB* surgiu o AJAX, que proporciona um mecanismo para separar o dado da aplicação, assim como no mundo das aplicações cliente/servidor ou em n-camadas, proporcionando mais velocidade, além de aplicações *WEB* mais limpas, claras, mais fáceis de desenvolver e gerir. (NIEDERAUER, 2007).

Entre as grandes empresas que utilizam o AJAX alguns exemplos de sucesso são o Google (buscas, mapas, *email*) e o Yahoo (*email*).

2.5 FOLHAS DE ESTILOS EM CASCATA (CSS)

Nesta seção será mostrada uma breve introdução a linguagem de folhas de estilo em cascata - CSS (*Cascading Syle Sheet*) e um pouco de sua história.

2.5.1 Introdução ao CSS

É uma linguagem de estilo utilizada para criar páginas *WEB* de uma maneira mais exata.

Através dessa linguagem são definidas as apresentações de documentos escritos em linguagem de marcação de texto como HTML ou XML. Com ele é possível fazer uma separação entre o formato e o conteúdo de um documento. (W3 Schools, 2011).

2.5.2 Versões do CSS

O objetivo inicial de CSS era separar o conteúdo da forma, e se cumpriu já com as primeiras especificações da linguagem. Entretanto, o objetivo de oferecer um controle total aos desenvolvedores sobre os elementos da página foi mais difícil de cobrir. As especificações anteriores das linguagens tinham muitas utilidades para aplicar estilos às páginas *WEB*, porém os desenvolvedores ainda continuavam usando diversos códigos improvisados para conseguir efeitos tão comuns ou tão desejados, como as bordas arredondadas ou a sombra de elementos na página, que só eram possíveis com auxilio de *software* editor de imagens, como *Adobe Fireworks*. (COLISSON, 2008).

O CSS 1 já significou um avanço considerável no projeto de páginas *WEB*, trazendo maior controle dos elementos da página. Porém, como ainda ficaram muitas coisas que os desenvolvedores desejavam fazer, mas que CSS não permitia especificar, estes faziam uso de truques para incrementar o estilo da página. A desvantagem da utilização destes truques é que muitas vezes implica alterar o conteúdo da página para incorporar novas tags HTML, que permitam aplicar estilos de uma maneira mais elaborada. Dada à necessidade de mudar o conteúdo, para alterar o desenho e fazer coisas que CSS não permitia, estava-se acabando com algum dos objetivos para os que CSS foi criado, que era o de separar por completo o conteúdo da forma. (BUDD, 2006).

CSS 2 incorporou algumas novidades, que hoje é utilizada habitualmente, porém CSS 3 ainda avança um pouco mais na direção de dar mais controle sobre os elementos da página.

A novidade mais importante que traz CSS 3 para os desenvolvedores *WEB* consiste na incorporação de novos mecanismos para manter um maior controle sobre o estilo com o qual se mostram os elementos das páginas, sem ter que recorrer as chamadas “gambiarras” que muitas vezes complicavam a exibição de páginas em navegadores sem suporte adequado como o “Internet Explorer”.

Com CSS 3 é possível aplicar estilos visuais iguais, ou até melhores, do que usando uma imagem desenhada em editores de imagens como Adobe Photoshop e Adobe Fireworks, além de manter uma velocidade de carregamento muito menor, pois todo efeito já vai estar acoplado a página, evitando carregamento de pesadíssimas imagens com seus efeitos visuais. (Site Tableless, 2011).

Algumas propriedades muito interessantes do CSS 3 e de fácil utilização é o *border-radius*, que arredonda cantos, que só era possível utilizando imagens com cantos sobrepostos, e também o *box-shadow* que aplica sombra no item desejado. No quadro 3 é apresentado um exemplo de código CSS.

```css
body {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
    font-family: verdana, arial, helvetica, sans-serif;
    color: #ccc;
    background-color: slategray;
    text-align: center;
}
```

Quadro 3 – Código CSS

A Figura 2 apresenta um exemplo de estilização CSS aplicada ao site, onde são mostrados os efeitos *border-radius* para arredondar cantos de quadros, e *box-shadow* para efeito de sombra sob caixas.

Figura 2 – Exemplo de aplicação de código CSS

2.6 SCAFFOLDING

O recurso de *scaffold* de aplicações é uma técnica que permite ao desenvolvedor definir e criar uma aplicação básica que possa inserir, selecionar, atualizar e excluir objetos (operações *CRUD – create, replace, update, delete*).

Ele permite que se tenha uma aplicação *CRUD* com total funcionamento em questão de poucos minutos. O uso de *scaffold* poupa o trabalho da criação da estrutura real, para acelerar o início de um projeto em etapas iniciais. *Scaffold* não tem a intenção de ser completamente flexível, mas sim uma forma temporária de fazer as coisas funcionarem até uma versão definitiva do aplicativo.

Scaffold é uma excelente maneira de iniciar o desenvolvimento de partes prematuras de uma aplicação *WEB*. Primeiras versões de esquemas de bases de dados tendem a sofrer mudanças, o que é algo perfeitamente normal nas etapas iniciais do projeto de aplicação. Isto tem um lado negativo: um desenvolvedor *WEB* não gosta de criar formulários que nunca virão a ser efetivamente usados. Para minimizar o esforço do desenvolvedor, o recurso de

scaffold é excelente. O *scaffold* analisa as tabelas da base de dados e cria listas padronizadas com botões de inserção, edição e exclusão, formulários padronizados para edição e visões padronizadas para visualização de um único registro da base de dados. (Site HTML Staff, 2011).

Um exemplo muito conhecido e utilizado por desenvolvedores *WEB* é o *phpScaffold.* A Figura 3 apresenta um *screenshot* de tela do *site* phpscaffold.com.

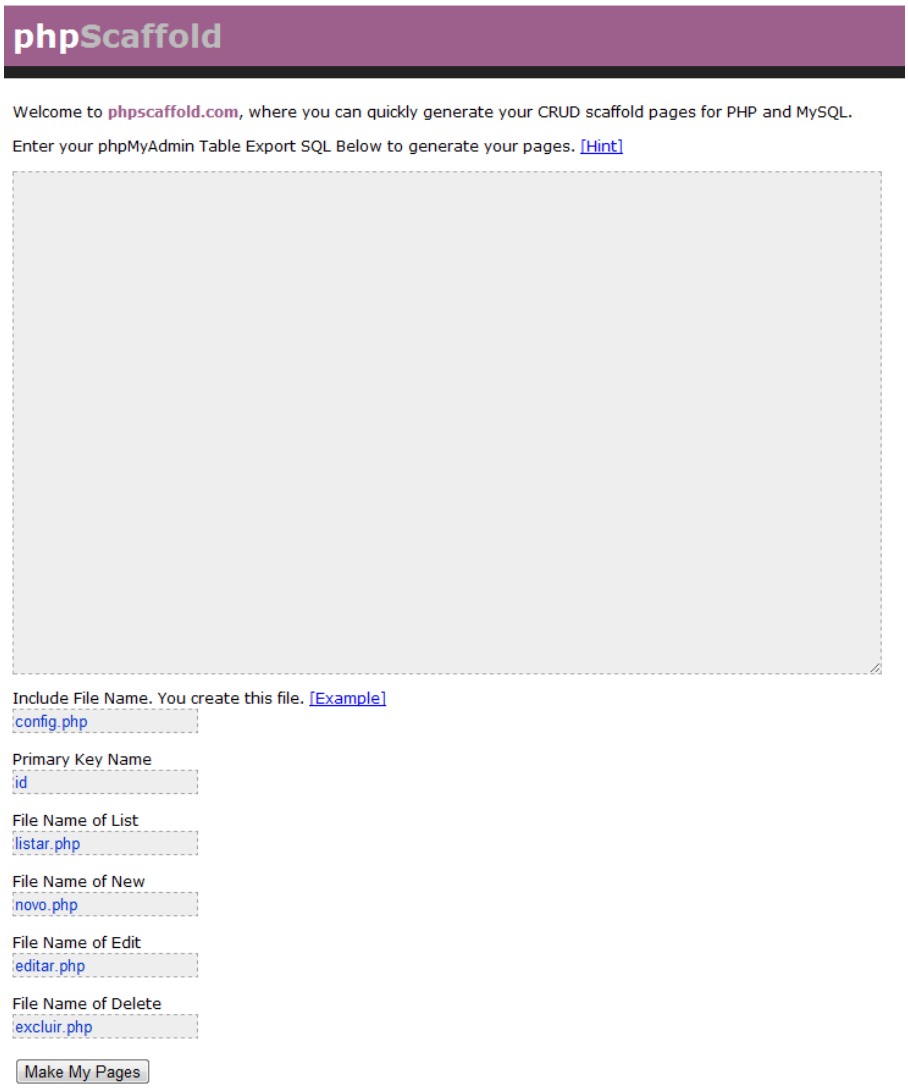

Figura 3 - *Screenshot* do site phpscaffold.com

2.7 HTML 5

HTML 5 (*Hypertext Markup Language*, versão 5) é a quinta versão da linguagem HTML. Esta nova versão traz consigo importantes mudanças quanto ao papel do HTML no mundo da *WEB*, trazendo novas funcionalidades como semântica e acessibilidade, com novos recursos antes só possíveis por meio de outras tecnologias, e trazendo uma importante disseminação dentre todos os novos navegadores de internet, tornando-o mais universal. (Site Tecmundo, 2011).

No início de 2008 o W3C(World Wide Web Consortium) – consórcio de empresas de tecnologia que coordena os padrões da internet quanto a linguagem – anunciou a primeira especificação do HTML 5. O HTML, que é responsável por organizar e formatar as páginas visitadas na internet está em sua versão 4.0.1 e continua evoluindo.

Após cinco anos de trabalho, desde 2008 esta ainda é apenas uma versão de testes do HTML 5, enquanto a versão final está prometida para 2012. Foram feitas grandes alterações, que incluem:

- Novas *API's*, entre elas uma para desenvolvimento de gráficos bidimensionais, chamada de "*Canvas*";
- Controle embutido de conteúdo multimídia;
- Aprimoramento do uso off-line;
- Melhoria na depuração de erros.

Esta evolução da linguagem padrão para *WEB* pode eliminar a necessidade de *plug-ins* para aplicações multimídia em navegadores, principalmente *Adobe Flash* e *Microsoft Silverlight*. Recentemente, Shantanu Narayen, diretor executivo da Adobe, afirmou em entrevista ao site de tecnologia IDGNOW(http://idgnow.uol.com.br), que o Flash não iria perder mercado, porém a versão 5 do HTML já está sendo chamada de "*Flash-killer*" (Assassino do Flash). Estas tecnologias precisarão se adaptar rapidamente para conseguir manter-se no mercado, tão populares quanto hoje.

Na avaliação do co-diretor de ferramentas da Mozilla, Ben Galbraith, as tecnologias viabilizadas pelo HTML 5 como o *Canvas* para desenhos 2D e o armazenamento de conteúdos no desktop, permitirão que o browser seja mais usado do que nunca.

Após dez anos sem atualizações, a forma como se escreve páginas na internet passa por uma boa transformação. O HTML 5 oferece uma experiência *WEB* totalmente diferente para usuários e embora exista um longo caminho para ser finalizado, muitos navegadores importantes, como Internet Explorer 9, Opera, Safari 4, Firefox 3.6 e Chrome já implementaram grandes partes da linguagem, incluindo *tags* de vídeo e suporte à tecnologia *Canvas*. Com a evolução da linguagem, os navegadores passam da categoria "mostradores" de páginas para um renderizador de "*WEB software*". O Quadro 4 apresenta um exemplo de código HTML 5.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo de Multimidia</title>
    <meta charset=UTF-8 />
  </head>
  <body>
        <article>
            <audio src="musica.mp3" controls></audio>
        </article>
        <article>
            <audio src="video.avi" controls></audio>
        </article>
  </body>
</html>
```

Quadro 4 – Código HTML 5

2.8 MYSQL

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (*Structured Query Language* - Linguagem de Consulta Estruturada) como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo, pertencendo atualmente à empresa Oracle, também dona da Sun Microsystems, a mesma que desenvolveu a linguagem Java. (GILMORE, 2008).

Entre os usuários do banco de dados MySQL estão: NASA, Friendster, Banco Bradesco, Dataprev, HP, Nokia, Sony, Lufthansa, U.S Army, US.

Federal Reserve Bank, Associated Press, Alcatel, Slashdot, Cisco Systems e outros.(WELLING, THOMSON, FURMANKIEWICZ; 2005).

O Quadro 5 demonstra um trecho de código *SQL*. No quadro é demonstrado o código para criação de uma tabela chamada *CLIENTE,* que irá conter os campos ID, que será do tipo inteiro, auto-incrementável e não nulo, além do campo NOME que será do tipo varchar de 100 caracteres não nulo, e também EMAIL outro varchar com 30 caracteres não nulo e ainda ENDERECO do tipo varchar de 30 caratecteres não nulo. O campo ID será criado como chave primária da tabela.

```
CREATE TABLE `cliente` (
  `id` int(10) NOT NULL auto_increment,
  `nome` varchar(100) collate latin1_general_ci NOT NULL,
  `email` varchar(100) collate latin1_general_ci NOT NULL,
  `endereco` varchar(30) collate latin1_general_ci NOT NULL,
  PRIMARY KEY  (`id`)
;
```

Quadro 5 – Exemplo de código *SQL*.

Na Figura 4 é mostrado um exemplo de modelo de banco de dados de um sistema de restaurante.

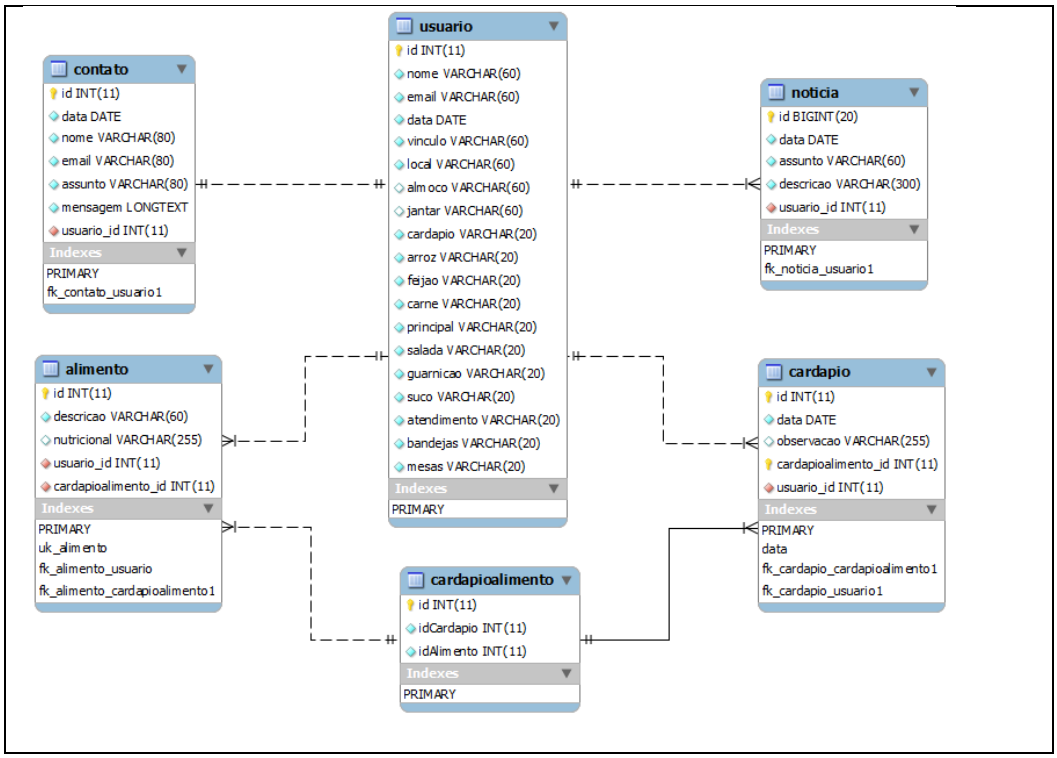

Figura 4 – MER Sistema de Restaurante

3 GERADOR DE FORMULÁRIO HTML

Nesta seção a seguir será apresentada a análise de requisitos e o sistema desenvolvido.

3.1 VISÃO GERAL DO SISTEMA

Trata-se de um sistema *WEB* que visa gerar formulários HTML a partir de *script* de tabelas de banco de dados desenvolvidas em SQL. O usuário do sistema utilizará uma página *WEB* em que escreverá o script SQL exportado do seu gerenciador de banco de dados, em que consta o banco de dados de seu sistema em desenvolvimento.

3.2 FERRAMENTAS UTILIZADAS

Para o desenvolvimento do sistema foram utilizadas as seguintes ferramentas:

- Netbeans 6.9: IDE (*Integrated Development Environment*), para codificação do sistema utilizando a linguagem PHP;
- StarUML 5.0: para o desenvolvimento de diagramas no padrão UML;
- WampServer: servidor local para execução do software em ambiente de desenvolvimento.

3.3 SUMÁRIO EXECUTIVO

O sistema será responsável por automatizar o processo de criação de formulários HTML referentes ao *CRUD* (*Create, Replace, Update, Delete*) do sistema desejado a partir do *script* SQL desenvolvido pelo programador.

O sistema será operado em ambiente *WEB*, onde o visitante terá total acesso as funcionalidades do sistema.

3.4 REQUISITOS FUNCIONAIS

Neste item será apresentado uma lista com os requisitos funcionais existentes no sistema.

- O sistema deverá permitir envio de script SQL para um formulário, onde o usuário informará o nome das páginas que serão geradas para listagem de registros, novos cadastros, edição de registros e exclusão; Além de opcionalmente escrever o caminho do arquivo de configuração do acesso ao banco de dados pelo PHP;
- O sistema deverá permitir que o usuário escolha o tema CSS pré definido na página para estilização de seus formulários a serem gerados;
- Usuário**:** visitante;
- Funcionalidades: Visitantes terão total acesso as funcionalidades do sistema;
- Ambiente**:** O sistema funcionará em ambiente *WEB*;
- Requisitos**:** Acesso a internet e navegador compatível com HTML 5 (Firefox 4, Chrome, Ópera, Internet Explorer 9).
- Atores: Visitante e Sistema.

3.5 CASOS DE USO

A Figura 5 apresenta o caso de uso de um visitante acessando o aplicativo.

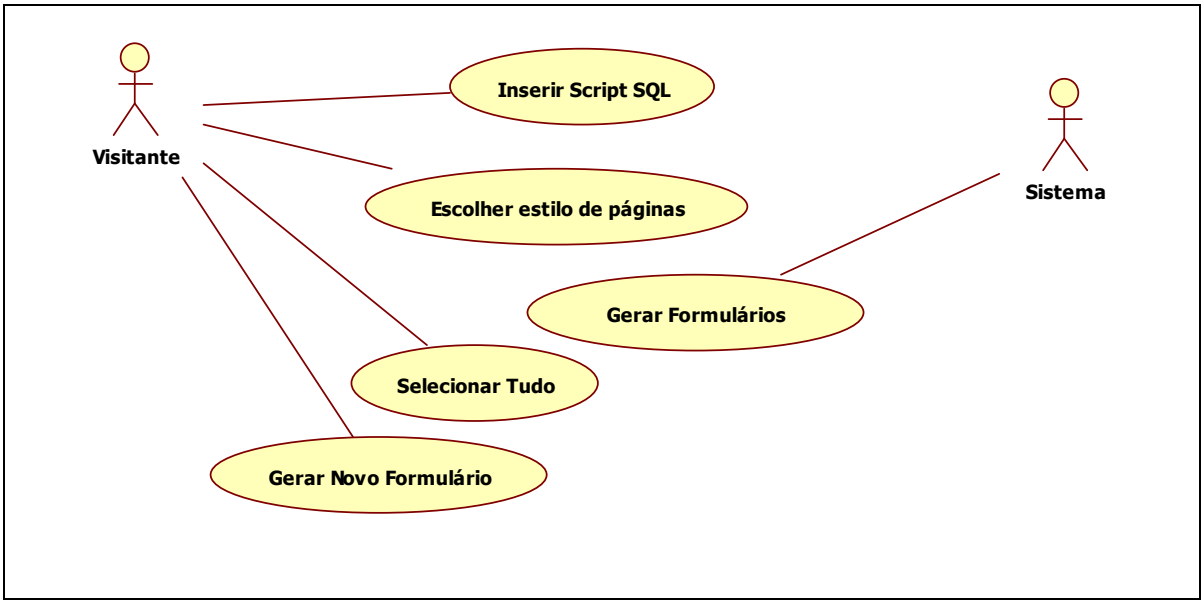

Figura 5 – Caso de Uso Visitante

3.6 DIAGRAMA DE CLASSE

Na Figura 6 é apresentado o diagrama de classe do sistema.

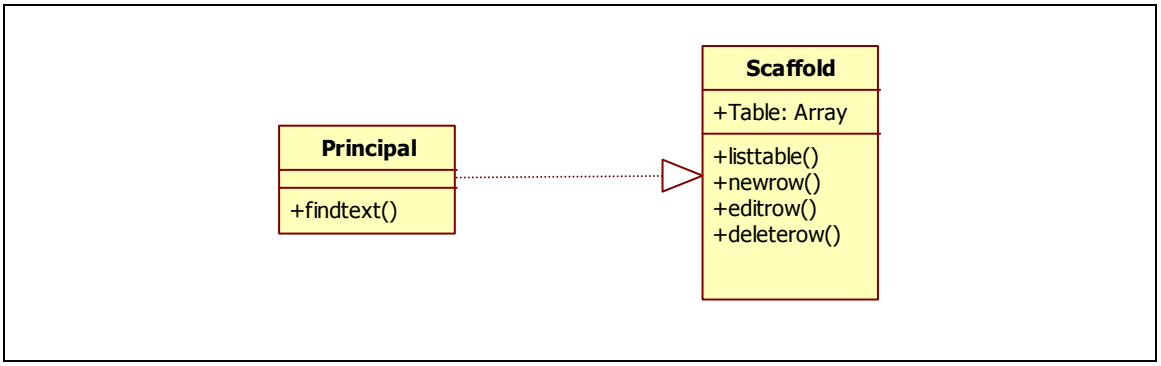

Figura 6 – Diagrama de Classe

3.7 DIAGRAMA DE SEQUÊNCIA

Na Figura 7 é demonstrado o diagrama de seqüência do sistema.

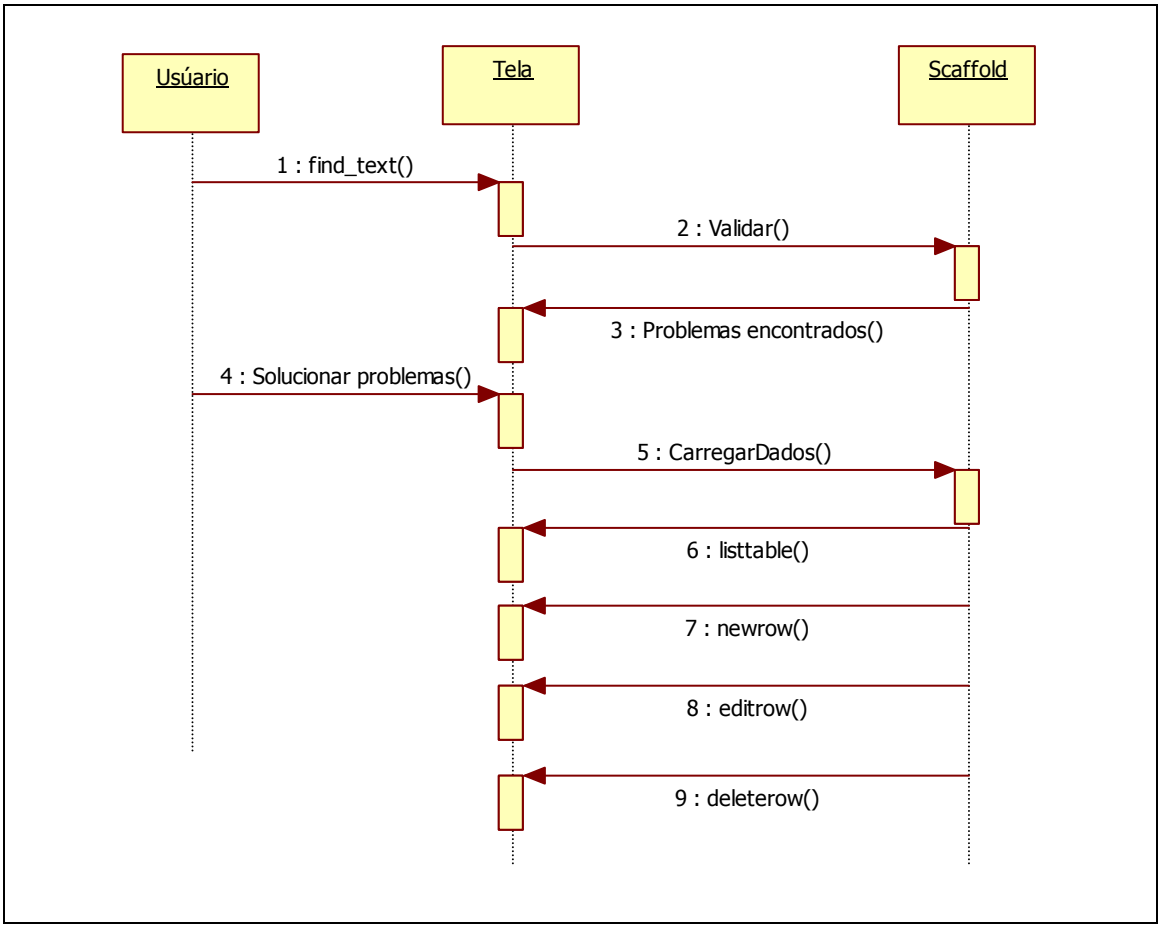

Figura 7 – Diagrama de Seqüência

3.8 TESTES

Por tratar-se de um sistema de pequeno porte, não foi necessária a utilização de um *framework* especifico para testes de software, sendo utilizado neste trabalho a forma tradicional de testes em PHP, com a inserção de instruções *"echo"* para verificação de passagem de *strings* ao decorrer do código, confirmando dessa forma se as variáveis ou instruções enviadas estariam chegando ao destino correto. A Figura 8 demonstra um teste simples de código, executado na função desenvolvida pra geração de uma página para inserção de um novo registro.

```
function newrow(){

            $return_string = '';
            echo "entrou aqui";
            $return_string .= "<? \n";
            $return_string .= "echo \"<!DOCTYPE html> \"; \n";
            echo "inseriu aqui";
            $return_string .= "echo \"<meta charset=UTF-08 /> \"; \n";
            if ($this->table['radio'] != '') $return_string .= "echo \"<style>{$this->table['radio']}</style>\"; \n";
            if ($this->table['include'] != '')  $return_string .= "include('{$this->table['include']}'); \n";
            $return_string .= "echo \" <center>\"; ";
            echo "retornando string";
             $return_string .= "echo \"<h1>Novo Registro</h1> \"; \n";
            $column_array = array();
            echo "passando pelo array";
            $text = '';

        foreach($this->table AS $key => $value) {
            echo "entrou aqui";
            if (is_array($value)) {
```

Figura 8 – *Screenshot* de tela de teste de código

Os resultados dos testes não influenciaram em nenhuma grande mudança na codificação do sistema. Os resultados foram positivos, conforme o esperado.

3.9 CODIFICAÇÃO

O sistema foi desenvolvido com linguagem PHP e efeitos com Ajax e JavaScript. O *layout* desenvolvido é simplista mas amigável, utilizando técnicas de *HTML* 5 e *CSS* 3.

A Figura 9 apresenta um *screenshot* da tela principal do sistema.

Figura 9 – *Screenshot* da tela principal do sistema.

No *index.php* são postados os scripts *SQL* das tabelas do banco de dados. O usuário poderá informar o caminho de seu arquivo de configuração na caixa de texto indicada como *config.php* e indicar a chave primária da tabela, geralmente identificada como *ID*.

3.9.1 Selecionando estilo de formulário

A escolha do estilo a ser usado no *CRUD* gerado é feita através de marcação de *radio-button* disponível. Por padrão, se nada for selecionado o *CRUD* será gerado sem estilização visual. A Figura 10 demonstra um a tela de seleção de estilo.

Figura 10 - *Screenshot* seleção de estilo.

O usuário também poderá selecionar o estilo *CSS* a ser utilizado em seus formulários, como no exemplo apresentado na Figura 11.

Figura 11 – *Screenshot* da página de estilo

Neste exemplo o estilo aplicado ao formulário gerado conterá texto centralizado, *background-color* azul, *tags* H1 em cor laranja e *inputs* sem formatação.

Após a postagem do script *SQL* e a seleção de estilo para as páginas, o usuário irá clicar no botão "*Gerar Formulários".* A partir desse ponto será chamada uma estrutura "*IF-ELSE",* que garantirá que a caixa de texto em que foi enviado o *script* SQL conterá o texto "CREATE TABLE 'nomedealgumatabela' ('". O Quadro 6 demonstra a estrutura de repetição, para verificação.

```
//pegar nome da tabela
if ( eregi('CREATE TABLE `(.)+` \(', $data, $matches) ) {
        $table['name'] = find_text($matches[0]);
        $max = count($data_lines);
        for ($i = 1; $i < $max; $i++ ) {
                if ( strpos( trim($data_lines[$i]), '`') === 0) { // esta linha tem uma coluna
                        $col = find_text(trim($data_lines[$i]));
                        $blob = ( stripos($data_lines[$i], 'TEXT') || stripos($data_lines[$i], 'BLOB') ) ? 1 :
                        $datetime = ( stripos($data_lines[$i], 'DATETIME') ) ? 1 : 0;
                        eval( "\$table['$col'] = array('blob' => $blob, 'datetime' => $datetime );");
                }
        }
        $show_form = 1;

}
else {
        $message .= "O código SQL > 'CREATE TABLE `nome_tabela` ( ' < não foi encontrado";
}
```

Quadro 6 – Verificação de caracteres.

Se contiver algum texto com inicial *“CREATE TABLE ‘nomedealgumatabela’ (”*, o código será executado e os formulários serão gerados. A Figura 12 apresenta a geração de formulários dentro das caixas de textos, todos contendo código CSS referente a seleção de estilo realizado na página anterior .

Figura 12 – *Screenshot* da página de geração de formulários

Se a caixa de texto não possuir o texto requerido para validação, uma mensagem de erro aparecerá no topo da página: "*O código SQL > 'CREATE TABLE ‘nome_tabela’ ' < não foi encontrado*". A Figura 13 demonstra a mensagem de validação.

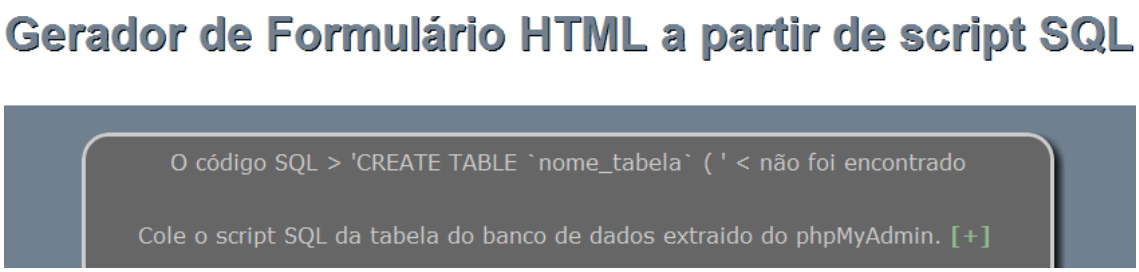

Figura 13 - Mensagem de Validação

3.9.2 Funções Criadas

Nesta seção serão mostradas as funções desenvolvidas para o funcionamento do sistema.

3.9.2.1 Função Find_Text

Nesta função é feita a procura do script SQL enviado dentro dos parênteses, como *CREATE TABLE `cliente` (`id` int(10) NOT NULL auto_increment, `nome` varchar(100) NOT NULL).* O Quadro 7 apresenta o código função find_text.

```
function find_text($text, $delimit_start = '`', $delimit_end = '`') {
        $start = strpos($text, $delimit_start);
        if ($start === false) return false;

        $end = strpos( substr($text, $start + 1), $delimit_end);
        if ($end === false) return false;

        return substr ( $text, $start + 1, $end);
}
```

Quadro 7 – Código função find_text.

3.9.2.2 Função Listtable

Está é a função que irá gerar o código para listagem dos registros gravados no banco de dados. O Quadro 8 apresenta a função *listtable*:

```
function listtable(){

    $return_string = '';
    $column_array = array();
    $return_string .= "<? \n";

            $return_string .= "echo \"<!DOCTYPE html> \"; \n";
            $return_string .= "echo \"<meta charset=UTF-08 /> \"; \n";
            if ($this->table['radio'] != '') $return_string .= "echo \"<style>{$this->table['radio']}</style>\"; \n";
    if ($this->table['include'] != '')  $return_string .= "include('{$this->table['include']}'); \n";
$return_string .= "echo \" <center>\"; ";
    $return_string .= "echo \"<h1>Listar</h1> \"; \n";
    $return_string .= "echo \"<table border=1 >\"; \n";
    $return_string .= "echo \"<tr>\"; \n";
    foreach($this->table AS $key => $value) {
        if (is_array($value)) {
            $column = $key ;
            $column_array[] = $key;
            $return_string .= "echo \"<td><b>". $this->title($column) ."</b></td>\"; \n";
        }
    }
```

Quadro 8 – Função *listtable*

3.9.2.3 Função NewRow

Esta é a função para gerar o código PHP do cadastro de novos registros. O Quadro 9 demonstra a função *newrow*, para inclusão de um novo registro no banco de dados.

```
function newrow(){

    $return_string = '';
    $return_string .= "<? \n";
            $return_string .= "echo \"<!DOCTYPE html> \"; \n";
            $return_string .= "echo \"<meta charset=UTF-08 /> \"; \n";
            if ($this->table['radio'] != '') $return_string .= "echo \"<style>{$this->table['radio']}</style>\"; \n";
    if ($this->table['include'] != '')  $return_string .= "include('{$this->table['include']}'); \n";
            $return_string .= "echo \" <center>\"; ";
             $return_string .= "echo \"<h1>Novo Registro</h1> \"; \n";
    $column_array = array();
    $text = '';

    foreach($this->table AS $key => $value) {
        if (is_array($value)) {
            $column = $key ;
            if($column != $this->table['id_key'] ){
                $column_array[] = $key;
                if($value['blob'] == 1){
                    $text .= $this->html_chars("<p><b>" . $this->title($column) . ":
                    </b><br /><textarea name='$column'></textarea> \n");
                }
                else {
                    $text .= "<p><b>" . $this->title($column) . ":</b><br /><input type='text' name='$column'/> \n";
                }
            }
        }
    }
```

Quadro 9 – Função *newrow*.

3.9.2.4 Função EditRow

Esta é a função criada para gerar o código de edição de registros. O Quadro 10 apresenta a função *editrrow*.

```
function editrow(){
    $return_string = '';
    $return_string .= "<? \n";
            $return_string .= "echo \"<!DOCTYPE html> \"; \n";
            $return_string .= "echo \"<meta charset=UTF-08 /> \"; \n";
            if ($this->table['radio'] != '') $return_string .= "echo \"<style>{$this->table['radio']}</style>\"; \n";
    if ($this->table['include'] != '')  $return_string .= "include('{$this->table['include']}'); \n";

            $return_string .= "echo \" <center>\"; ";
            $return_string .= "echo \"<h1>Novo Registro</h1> \"; \n";
    $column_array = array();
    $text = '';

    $return_string .= "if (isset(\$_GET['{$this->table['id_key']}']) ) { \n";

        $return_string .= "\${$this->table['id_key']} = (int) \$_GET['{$this->table['id_key']}']; \n";
```

Quadro 10 – Função *editrow*

3.9.2.5 Função DeleteRow

E por ultimo a função para excluir registros do banco de dados. O Quadro 11 demonstra a função *deleterow*.

```
function deleterow(){
    $return_string = '';
    $return_string .= "<? \n";
            $return_string .= "echo \"<!DOCTYPE html> \"; \n";
            $return_string .= "echo \"<meta charset=UTF-08 /> \"; \n";
            if ($this->table['radio'] != '') $return_string .= "echo \"<style>{$this->table['radio']}</style>\"; \n";
    if ($this->table['include'] != '')  $return_string .= "include('{$this->table['include']}'); \n";

            $return_string .= "echo \" <center>\"; ";
            $return_string .= "echo \"<h1>Excluir</h1> \"; \n";
```

Quadro 11 – Função *deleterow*

4 CONSIDERAÇÕES FINAIS

Através deste estudo é possível concluir que um *framework* para desenvolvimento ágil tem grande importância no uso cotidiano de desenvolvedores WEB que sempre buscam novas ferramentas para aumento de produtividade em seus projetos.

Com o uso do gerador de formulários HTML, os desenvolvedores poderão ficar focados na lógica do negócio de seus aplicativos em desenvolvimento, e dessa forma economizarão tempo por evitarem escrever linhas de código desnecessárias, que seriam usadas somente para testes, que em muitos casos são apagadas por completo até o final do ciclo do projeto.

4.1 TRABALHOS FUTUROS

Este trabalho sugere o desenvolvimento de um módulo administrativo para o software de geração de formulários HTML a partir de *scripts* SQL. No módulo administrativo poderão ser cadastrados novos temas estilizados com CSS, diariamente, aumentando a variedade de recursos disponíveis para o usuário do *framework*.

Com o uso de recursos avançados de RIA através de tecnologias como *Ajax*, HTML 5 e CSS 3, poderá também ser criado um gerador de estilo visual interativo, o qual poderá ser usado para mudar tamanho, posição, cor, efeitos e outros vários itens disponíveis nos formulários gerados.

REFERÊNCIAS

Blog Cliente Web. Disponível em**:** <clienteweb.blogspot.com/ >. Acessado em 05/04/2011.

BUDD, Andy. **Criando páginas web com css: soluções avançadas para padrões web**. São Paulo, SP: Pearson Prentice Hall, 2006.

CAMARGOS, Luiz Fernando Macedo; MENEZES, Marco Antonio Figueiredo. **Introdução à HTML e PHP**. Rio de Janeiro, RJ: Ciência Moderna, 2008.

COLISSON, Simon. **Desenvolvendo CSS na Web**. Rio de Janeiro: Alta Books, 2008.

DALL'OGLIO, Pablo. PHP **Programando com Orientação a Objetos.** Novatec. 2009.

GILMORE, W. Jason. **Dominando PHP e MySQL: do iniciante ao profissional**. Rio de Janeiro: Alta Books, 2008.

GONÇALVES, Edson**. Ajax na prática: todo o poder do AJAX com frameworks JavaScript independentes do servidor, aliados ao desenvolvimento Web 2.0.** Rio de Janeiro, RJ: Ciência Moderna, 2007.

MELONI, Julie C. **Fundamentos de PHP**. Rio de Janeiro: Ciência Moderna, 2000.

MILANI, André. **Construindo aplicações web com PHP e MySQL**. São Paulo: Novatec, c2010.

NIEDERAUER, Juliano. **Desenvolvendo websites com PHP**. 2. ed. São Paulo: Novatec, 2011.

NIEDERAUER, Juliano. **Web interativa com Ajax e PHP**. São Paulo: Novatec, 2007.

PHP Scaffold. Diponível em: <www.phpscaffold.com>. Acessado em 02/04/2011.

SILVA, Maurício Samy. **JavaScript: guia do programador**. São Paulo: Novatec, 2010.

SILVA, Osmar J. **JavaScript - Guia Prático do Webmaster.** 2a edição, 2005.

Site Tableless. Disponível em: <http://www.tableless.com.br/css3-o-que-vem-por-ai>. Acessado em 05/04/2011.

Site HTML Staff. Disponível em: <http://www.htmlstaff.org>. Acessado em 10/04/2011.

Site Tecmundo. Disponível em: <http://www.tecmundo.com.br/2254-o-que-e-html-5-.htm>. Acessado em 07/04/2011.

SOARES, Walace. **AJAX (Asynchronous JavaScript And XML): guia prático.** 3. ed. São Paulo: Érica, 2007.

W3 Schools – CSS. Disponível em: <www.w3schools.com/css/> Acessado em 08/04/2011.

W3C. Disponível em: <www.w3c.org>. Acessado em 10/04/2011.

WELLING, Luke; THOMSON, Laura; FURMANKIEWICZ, Edson. **PHP e MySQL: desenvolvimento Web**. Rio de Janeiro: Campus, 2005