UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

CARLOS CORRÊA SILVA ALVES

Sistema Web para Gerência de Formulários de Exames e Pesquisas Aplicados a Pacientes na área do Sono

Trabalho de Graduação.

Prof. Dr. Leandro Krug Wives (orientador)

Prof. Dr. Denis Martinez (co-orientação)

Porto Alegre, novembro de 2009.

AGRADECIMENTOS

Gostaria de agradecer, em primeiro lugar, a toda a minha família que me apoiou e incentivou nesta caminhada. Em especial à Flora, minha mãe, Gloria, minha avó e Flávia, minha esposa que estiveram sempre ao meu lado trazendo uma palavra amiga ou um gesto motivador em todos os momentos.

Deixo também registrado neste trabalho um profundo agradecimento ao meu co-orientador, Dr. Denis Martinez, o qual forneceu todo subsídio necessário além de seu entusiasmo e conhecimento para que fosse possível realizar este estudo.

Por fim, deixo o meu imenso apreço ao prof. Dr. Leandro Wives, orientador e amigo que, com toda sua experiência e conhecimento me guiou nesta atividade a qual conclui uma etapa de minha vida.

A todos que, de uma forma ou de outra contribuíram para a realização deste trabalho, o meu Muito Obrigado.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 6
LISTA DE FIGURAS 7
RESUMO 9
ABSTRACT 10
1 INTRODUÇÃO 10
1.1 Contextualização 10
1.2 A busca pela solução ideal 11
2 ANÁLISE DE REQUISITOS 12
2.1 Trabalhos relacionados 12
2.2 Funcionalidades necessárias 13
2.3 Linguagem e ambiente de desenvolvimento 13
2.4 Arquitetura de sistemas Web 14
2.5 Formulários estáticos e dinâmicos 15
3 DETALHAMENTO DA SOLUÇÃO 17
3.1 Visão geral da estrutura do sistema 17
3.2 Estrutura das classes 20
3.3 Modelo da base de dados 22
3.4 Detalhes da implementação 24
3.5 Estrutura de construção de formulários dinâmicos 29
3.5.1 Entrada de dados do tipo texto 31
3.5.2 Entrada de dados do tipo botão rádio 32
3.5.3 Entrada de dados do tipo caixa de seleção (lista suspensa) 33
3.5.4 Atributos comuns de um campo do formulário 33
4 INTERFACE COM USUÁRIO E OPERAÇÃO DO SISTEMA 35
4.1 Interface de cadastro e edição de pacientes 36
4.2 Interface de busca e listagem de pacientes 38

4.3 Interface de cadastro de formulário dinâmico 39
4.4 Interface de associação entre formulários e pacientes 41
4.4.1 Interface de associação de um paciente a um ou mais formulários 41
4.4.2 Interface de associação de um formulário a um ou mais pacientes 42
4.5 Interface de preenchimento de formulários dinâmicos 43
5 AVALIAÇÃO DA FERRAMENTA 46
6 CONCLUSÕES 47
REFERÊNCIAS 49
BIBLIOGRAFIA 50
APÊNDICE A – ARQUIVOS E PASTAS DO SISTEMA 51
APÊNDICE B – DICIONÁRIO DO BANCO DE DADO 54

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
GUI	Grafical User Interface
HTML	Hyper-Text Markup Language
PHP	Hypertext Processor
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structered Query Language
UFRGS	Universidade Federal do Rio Grande do Sul

LISTA DE FIGURAS

Figura 3.1: *Template* principal do sistema..... 18
Figura 3.2: Página com um formulário instanciado. 19
Figura 3.3: Página com um *grid* instanciado. 20
Figura 3.4: Configuração de entrada de dados (campo) de um formulário dinâmico. ... 22
Figura 3.5: Modelo de dados dos cadastros básicos. 23
Figura 3.6: Modelo de dados da estrutura de formulários dinâmicos. 23
Figura 3.7: Tela inicial do sistema..... 25
Figura 3.8: Mescla de grid com formulário, gerado a partir da classe GridControlPage. 28
Figura 3.9: Página inicial de configuração de campos de um formulário dinâmico. 31
Figura 3.10: Entrada de dados do tipo texto. 32
Figura 3.11: Controle de entrada de dados do tipo Botão rádio. 32
Figura 3.12: Entrada de dado do tipo radio. 32
Figura 3.13: Entrada de dado do tipo caixa de seleção. 33
Figura 4.1: Tela de login. 35
Figura 4.2: Usuário devidamente autenticado, na página inicial do sistema..... 36
Figura 4.3: Tela de cadastro de pacientes. 37
Figura 4.4: Após realizar o cadastro, o sistema informa o novo código. 37
Figura 4.5: Listagem de formulários disponíveis para o preenchimento. 38
Figura 4.6: Formulário de busca de pacientes. 39
Figura 4.7: Campo 3 somente irá ser exibido se a resposta do campo 1 for “Sim”. 40
Figura 4.8: Tela que relaciona formulários a um paciente específico. 42
Figura 4.9: Listagem de formulários disponíveis. 42
Figura 4.10: Listagem de pacientes que podem ser selecionados para associar ao formulário. 43
Figura 4.11: Na tela de edição de cadastro de pacientes, pode-se selecionar qual formulário se deseja preencher. 43

Figura 4.12: Exemplo de um formulário a ser preenchido. 44
Figura 4.13: Exemplo de um formulário a ser preenchido. 44

RESUMO

O presente trabalho descreve a modelagem e a implementação de um sistema Web para gerenciamento de formulários dinâmicos de exames e de pesquisas da área do sono. O sistema foi desenvolvido utilizando a linguagem de programação PHP e base de dados MySQL.

Com base no sistema utilizado na Clínica do Sono, foi elaborado um modelo de dados para armazenar perguntas e respostas em forma de formulários. Após a modelagem de dados, foi desenhada uma interface que fornecesse ao usuário a capacidade de elaborar questões e associar respostas de cada paciente.

Em todo o trabalho, há a descrição detalhada das tecnologias, conceitos e modelos envolvidos em cada módulo do sistema, bem como os problemas encontrados e a forma como foram solucionados.

Por fim, é disponibilizado um manual operacional do sistema com a descrição detalhada de cada tela.

Palavras-Chave: Aplicação Web, PHP, MySQL, informática médica, tratamento do Sono, formulários dinâmicos.

The development of a Web System to Manage Dynamic Medical Forms and Enquiries in the Sleep Context

ABSTRACT

This work describes the modeling and the implementation of a Web system for managing dynamic forms and enquires in sleep context. The system was developed using PHP programing language and MySQL database.

Based on the system used at the Clínica do Son, a model of data was composed to store questions and answers in the shape of forms. After placing the data in a pattern, an interface was drawn to provide the user with the capacity to elaborate questions and associate answers from each patient.

On every project, there is a detailed description of the technologies, concepts and models involved in each module of the system, as well as the problems found and the way they were solved.

Finally, an operational manual of the system is made available with the detailed description of each screen.

Key words: Web system, PHP, MySQL, medical computer science, Sleep treatment, dynamic forms.

1 INTRODUÇÃO

1.1 Contextualização

O presente trabalho tem como principal objetivo modelar e implementar um sistema de gerenciamento de formulários de entrada de dados de cadastros gerais e dados de pesquisas para a Clínica do Sono, sob co-orientação do Dr. Denis Martinez. A Clínica do Sono é uma clínica especializada em tratamentos médicos voltados para os distúrbios do sono. Além de realizar tratamentos especializados, a equipe realiza coleta de informações pertinentes à realização de estudos e pesquisas desta área.

A idéia de se construir uma ferramenta capaz de gerenciar formulários de pesquisa teve base num estudo sobre o atual sistema utilizado na Clínica. O sistema atual compreende uma base de dados Access[1] que é preenchida através da transcrição da coleta feita em formulários de papel.

Na primeira etapa do estudo realizado buscou-se entender o modelo de dados, bem como a dinâmica de cadastramento para que fosse possível modelar e sugerir uma técnica mais apropriada e eficiente. Assim, surgiu a possibilidade da construção de diversos formulários, alguns de cadastros básicos e outros traduzindo o que seriam os questionários de pesquisas e exames.

Contudo, durante a especificação e a construção dos primeiros protótipos de tela, percebeu-se o grande dinamismo com o qual estes questionários surgiam ou tornavam-se obsoletos. Em alguns casos, havia a necessidade de apenas incluir ou retirar alguma pergunta.

Assim, deu-se início à modelagem e a implementação de uma ferramenta capaz de elaborar formulários dinâmicos, com os quais é possível construir questionários de forma personalizada, sem a necessidade de codificação ou alteração do sistema.

[1] Access é um sistema de banco de dados da Microsoft distribuído no pacote MS Office. Trata-se de um aplicativo compacto que fornece funcionalidades para construção de um pequeno banco de dados disponibilizando também estruturas de formulários para a inserção de dados e elaboração de relatórios.

1.2 A busca pela solução ideal

Uma breve análise do sistema atualmente utilizado pela Clínica do Sono deixa claro que os recursos de coleta de dados são onerosos e complicados, sendo, em sua maioria, feitos através de formulários impressos com preenchimento de forma manuscrita.

Por isso buscou-se elaborar formulários que facilitassem o processo de cadastramento de dados, sejam eles oriundos de usuários da clínica, pacientes da clínica ou indivíduos relacionados a pesquisas elaboradas pela clínica. A idéia proposta contemplou a construção de formulários Web disponibilizados na Internet, de tal forma que o próprio fornecedor da informação realize a inserção dos dados ao invés do preenchimento de formulários de papéis, tirando a carga inicial dos funcionários da clínica.

Um fato que justifica o desenvolvimento voltado para Web é a facilidade de acesso, tendo em vista a possibilidade de se entrar no sistema através de qualquer computador que esteja conectado à rede mundial. Ou seja, a centralização da informação com a descentralização do acesso.

Com base em preceitos ergonômicos e dentro dos conceitos da Engenharia de Software, buscou-se a elaboração de formulários de fácil entendimento e com leiaute agradável, que não canse o usuário durante o preenchimento, bem como a automação necessária para realizar a entrada correta dos dados.

Entendida a natureza do problema, buscou-se viabilizar dois tipos de formulários: estáticos e dinâmicos. O primeiro tipo foi desenvolvido para realizar a inserção de dados consolidados, como informações de pacientes e cadastros gerais. O segundo, dito formulário dinâmico, viabilizou a confecção de formulários personalizados, no qual é possível determinar os campos e as formas de resposta esperada. Desta maneira, um usuário capaz de elaborar estes formulários, pode livremente determinar quais questões o formulário deve possuir bem como o estilo de resposta, como uma entrada de texto livre, ou uma opção dentre algumas respostas padrões, por exemplo.

2 ANÁLISE DE REQUISITOS

Pela análise inicial do problema proposto, ficou clara a necessidade de fornecer ao usuário do sistema uma grande capacidade de customização, tendo em vista a grande dinâmica associada à entrada de dados. Na prática, o que se percebeu foi a pequena gama de formulários estáticos, tais como cadastro de pacientes e cadastros de dados associados aos pacientes (profissão, convênios, etc.). Porém, havia uma grande quantidade de tabelas armazenando o que seriam os questionários de pesquisas e dados de consultas de pacientes. Estas tabelas armazenavam dados relativos a pesquisas e exames médicos realizados em pacientes que freqüentavam a clínica e, posteriormente, estes dados eram tabulados fornecendo subsídios para pesquisas e trabalhos realizados pelo Dr. Denis.

Percebeu-se então a necessidade de construir um sistema capaz de, inicialmente, possibilitar a entrada de dados dos cadastros básicos e, num segundo momento, viabilizar um mecanismo de especificação e construção de questionários e formulários personalizados.

Neste capítulo serão detalhados os aspectos relevantes para a modelagem e a construção do sistema de gerenciamento de formulários.

2.1 Trabalhos relacionados

Sabe-se que formulários Web são amplamente utilizados como uma forma eficaz de coleta de dados[2]. Na grande maioria dos sistemas, estes formulários refletem alguma estrutura pré-definida de um modelo de dado, isto é, servem de interface para um modelo consistente de banco de dados.

[2] Desde os primeiros sistemas de cadastramento, a forma mais utilizada de inserção de dados é a utilização de formulários. Com o advento da internet, passou a se utilizar formulários Web, que, assim como os tradicionais formulários de aplicações Standalone, viabiliza controles para inserir textos ou selecionar opções pré cadastradas. Os autores Freitas, Janissek (R.) e Moscarola em seu artigo “Dinâmica do processo de coleta e análise de dados via web” (2004), fundamentam a coleta de dados através de formulários Web.

A novidade que este trabalho propõe está fundamentada em alguns sistemas existentes. Um deles, conhecido como Google Enquete[3], fornece uma forma de seus usuários criarem enquetes com perguntas personalizadas. Cada pergunta possui um conjunto de respostas possíveis, também determinadas pelo próprio usuário.

Outro trabalho de interesse, seguindo a idéia de ferramenta para elaboração dinâmica de interface, é o Trabalho de Diplomação do Marcos P. Ferreira (FERREIRA, 2003), no qual ele descreve e implementa um *Framework*[4], chamado "PHP2Go", que fornece uma API capaz de construir *GUIs* de maneira rápida e consistente. Entre os componentes descritos em seu trabalho, há a elaboração de formulários, ainda que estáticos, mas totalmente parametrizáveis.

2.2 Funcionalidades necessárias

O sistema de Gerenciamento de Formulários deve ser capaz de fornecer a entrada de dados de cadastros básicos, a elaboração de formulários dinâmicos e uma forma de gerenciar as informações armazenadas.

Além destas funcionalidades, a ferramenta deve possuir um sistema de gerenciamento de usuários e controle de acesso.

Como forma primária de visualização, deve-se utilizar os próprios formulários para exibição dos dados cadastrados. Para o caso de haver listagem de dados, devem ser utilizados estruturas do tipo *Grid*[5].

2.3 Linguagem e ambiente de desenvolvimento

Para poder disponibilizar este processo de entrada de dados o sistema foi desenvolvido para o ambiente Web, isto é, possui uma arquitetura baseada na Web. Para tanto, do lado do servidor a tecnologia de programação utilizada é a linguagem PHP e do lado do cliente é o HTML/JavaScript[6].

---

[3] O sistema da Google disponibiliza uma interface rica e bem estruturada para montar enquetes. Estas enquetes apresentam perguntas e controles para inserir os dados solicitados. Um bom tutorial que demonstra este sistema pode ser encontrado no seguinte site: http://www.bgk.com.br/blog/tutoriais/Enquete/Enquete_GDocs.htm.

[4] Conceitualmente, Framework é um conjunto de abstrações próprias para resolver um problema de um domínio específico. Framework de um software compreende num conjunto de implementações, de uma linguagem específica, utilizada para auxiliar no desenvolvimento de um sistema (http://pt.wikipedia.org/wiki/Framework).

[5] Uma forma muito comum de se exibir dados tabulados é através de matrizes, isto é, tabelas onde cada linha corresponde a um registro. Um Grid, no contexto deste trabalho, é o nome dado a esta matriz.

[6] Javascript é uma linguagem de script que possibilita interagir com o conteúdo de uma página HTML. Mais detalhes sobre esta linguagem podem ser encontrados em Goodman (2004).

Para o lado do servidor, foi utilizado o sistema Apache[7], que implementa um servidor Web, juntamente com o Banco de Dados MySQL, uma vez que é um SGBD robusto e sintonizado para o uso com a linguagem PHP.

Sobre o PHP pode-se dizer que é "(...) uma linguagem de criação de scripts do lado do servidor, que pode ser incorporada ao HTML ou utilizada como um binário independente." (CONVERSE, 2003). Suas características principais são: código aberto e gratuito, otimizada para utilizar com o servidor Apache e o banco de dados MySQL, sintaxe similar a da linguagem C, possibilidade de programação imperativa e/ou orientada a objetos, estável e, por fim, muito popular.

A linguagem PHP, embora muito rica em seus recursos, é uma linguagem fracamente tipada. Para a implementação de grandes sistemas, isso muitas vezes se torna um problema pelo simples fato de se perder a garantia e a segurança da utilização das estruturas de dados[8]. Ainda, por ser uma linguagem interpretada, muitos dos erros sintáticos só são percebidos em tempo de execução (chamadas de funções não declaradas, por exemplo). Mesmo assim, a facilidade de se criar e trabalhar com estruturas de dados é muito útil quando se trata de sistemas Web.

2.4 Arquitetura de sistemas Web

O modelo elaborado está calcado no conceito de aplicação cliente/servidor (BENETT, 1997) construído em três camadas: apresentação / servidor de aplicação / servidor de banco de dados. Nesse modelo, tem-se o usuário utilizando um Navegador Web (*Web Browser*) como *front-end* da aplicação, conectado a uma rede internet (ou intranet) que propicia o acesso ao servidor de aplicação. O servidor de aplicação por sua vez armazena e executa as regras de negócio, acessando, se necessário, um terceiro servidor que exerce a função de persistência de dados.

Em alguns casos, pode-se encontrar uma parte das regras de negócio na camada do cliente, isto é, parte da aplicação roda no navegador do computador do usuário. Esse tipo de modelo é relativamente útil quando se deseja reduzir a carga no servidor de aplicação deixando-o apenas encarregado de realizar a interface e comunicação com o servidor de dados. E, em especial para o sistema de gerenciamento de formulários dinâmicos, uma extensiva parte da codificação foi escrita para propiciar uma maior interatividade no lado do cliente.

Segundo Benett (1997, p. 20), a tecnologia Web proporciona três grandes vantagens: uma plataforma universal, um modo de exibição unificado e uma língua franca. Ainda, em sua obra, retrata que "A tecnologia Web acrescenta dois elementos à computação

---

[7] Maiores detalhes em http://www.apache.org/.

[8] Na realidade, existem muitas vantagens em se desenvolver através de uma linguagem fracamente tipada como, por exemplo, uma maior flexibilidade no tratamento dos dados. Muitos desenvolvedores defendem esta abordagem, contudo, ressaltam a necessidade de se tomar um maior cuidado durante a codificação do sistema. Sebesta (2005) traz em sua obra uma entrevista (pags.: 216 e 217) com o desenvolvedor Rasmus Lerdorf, responsável pela infra-estrutura do site *Yahoo* o qual descreve as vantagens de se utilizar linguagens de *Script*, dentre elas o PHP, para solucionar problemas.

baseada em documentos. Em primeiro lugar, ela cultua os documentos HTML como padrão universal[...]. Em segundo lugar, a tecnologia Web independe de plataforma.".

Como exemplos de sistemas Web, pode-se citar o site de e-commerce Submarino[9], o site de conveniência do Banco do Brasil[10] e até mesmo o sistema de matriculas da UFRGS[11].

Assim, pode-se descrever, de forma sucinta e com base na obra de Benett (1997), que a arquitetura de sistemas Web é composta pelos seguintes itens:

- um navegador (*web browser*) que fornece a interface visual com o usuário;
- um servidor que contem a lógica da aplicação;
- fluxo de dados sempre inicia pelo usuário no qual realiza requisições ao servidor e aguarda a resposta do mesmo.

Além disso, a arquitetura possui as seguintes características:

- geralmente não há necessidade de se instalar qualquer aplicativo adicional na máquina do usuário;
- a sessão de trabalho (e persistência de objetos e estrutura de dados) geralmente só existe durante o processamento da requisição do usuário pelo servidor, embora existam recursos para se gravar por algum tempo os dados que venham a ser necessários existir em memória;
- o sistema é dito on-line, pois está sempre disponível dede que o usuário tenha acesso a um navegador e conectado ao servidor de aplicação.

2.5 Formulários estáticos e dinâmicos

Conforme Cury (2005), formulário é um documento com campos pré-impressos onde são preenchidos os dados e as informações, e que permite a formalização das comunicações, o registro e o controle das atividades das organizações.

Sistemas Web apresentam grande facilidade para o desenvolvimento de formulários, comumente chamados de *Web Forms* ou simplesmente Formulários HTML, segundo Crane (2007). Uma de suas grandes vantagens é o fato de dispensar papéis ou qualquer outro meio impresso de se exercer a coleta de informação. Além disto, os recursos computacionais disponíveis através de formulários HTML permitem criar uma experiência rica e altamente interativa com o usuário.

Outra grande vantagem, a qual embasa este trabalho, é a maneira pela qual podem ser elaborados, isto é, sua implementação pode ser feita de maneira estática, inicialmente especificada e após codificada, ou então de forma dinâmica, a qual possibilita ao usuário um grande poder de personalização.

---

[9] http://www.submarino.com.br

[10] http://www.bb.com.br

[11] https://www11.ufrgs.br/PortaisUfrgs/portal_do_aluno/aluno.htm

Especificamente, os campos de um formulário HTML apresentam, no mínimo, dois atributos: um identificador e um valor associado. Geralmente, este identificador está mapeado a um elemento de um repositório. A natureza, então, de um formulário estático, é manter constante este mapeamento. Logo, em formulários dinâmicos, o mapeamento deve ser feito durante a carga do mesmo.

Como exemplo de um formulário estático tem-se uma entrada de texto mapeada para um campo qualquer de uma tabela de um banco de dados. Desta forma, o principal trabalho da camada de aplicação é inserir o valor atribuído ao identificador de entrada de texto ao campo apropriado da tabela do banco de dados.

Por outro lado, formulários dinâmicos apresentam uma maior complexidade no sentido em que não há este mapeamento forte entre os campos do formulário e o modelo da camada de aplicação. Assim, é necessário um modelo que realize de forma dinâmica a associação entre os campos do formulário e os campos do banco de dados.

Outros detalhes, não menos importantes, estão relacionados aos leiautes de visualização dos componentes, à limitação dos controles disponíveis, ao tratamento das informações inseridas e ao modelo de dados que tornam a implementação de formulários dinâmicos uma tarefa onerosa.

Por fim, cabe mencionar a grande capacidade funcional em que um sistema com o gerenciamento de formulários dinâmicos pode proporcionar. Uma vez que o usuário possa determinar seus próprios formulários, ele passa a ter uma grande flexibilidade de personalização e independência de manutenção do sistema. Por outro lado, no que diz respeito ao sistema, este pode ser modularizado e disponibilizado como produto independente do negócio a ser implantado.

3 DETALHAMENTO DA SOLUÇÃO

Neste capítulo tem-se o detalhamento das etapas e das estruturas modeladas e codificadas para construção do Sistema de Gerenciamento de Formulários. Tem-se, como um dos objetivos deste capítulo, fornecer um ponto de partida para o entendimento da codificação, bem como a estrutura de classes e modelos de dados do sistema desenvolvido.

3.1 Visão geral da estrutura do sistema

O sistema foi dividido em três camadas: visualização, aplicação e dados. A camada de visualização apresenta o esquema de *templates*[12], que torna fácil a manutenção e ajuda no conceito de reutilização. A camada de aplicação apresenta toda a lógica do tratamento de dados e validações. Por fim, a camada de dados implementa as funcionalidades de acesso ao banco de dados. Trata-se de uma estrutura clássica e bem definida.

Ainda, na camada de visualização, tem-se arquivos no formato HTML que expressam as interfaces visuais na forma de páginas Web. Existe uma página principal que trás o *template* principal, ou seja, o formato padrão que todas as telas do sistema deve ter. Este formato padrão inclui um cabeçalho no topo da página, a localização dos menus e a área de texto (também chamada de área de conteúdo). Na Figura 3.1 pode-se ver a estrutura do *template* principal.

Na área de conteúdo são instanciadas as demais estruturas visuais, como formulários e *grids* de visualização, como mostram as figuras 3.2 e 3.3. Cada formulário ou *grid* associado ao *template* principal pode trazer consigo implementação de *scripts* Javascript para programar o comportamento do sistema no lado do cliente. Isto acontece, por exemplo, na interface de elaboração dos formulários dinâmicos.

A camada de aplicação apresenta, fundamentalmente, as regras de negócio associadas às funcionalidades do sistema. Em sua grande maioria, realiza validação dos dados enviados pelo usuário e estabelece a comunicação com a camada de acesso aos dados.

---

[12] Template (ou "modelo de documento") é um documento sem conteúdo, com apenas a apresentação visual. Por exemplo, conteúdos que podem aparecer no início e conteúdos que só podem aparecer no final. Web *templates* (ou "modelos de página") são instrumentos utilizados para separar a apresentação do conteúdo em web design, e para a produção massiva de documentos Web (site: http://pt.wikipedia.org/wiki/Templates).

E, finalmente, a camada de dados apenas fornece uma API de acesso ao banco de dados, sendo que, além do conhecimento das regras de negócio, é também função da camada de aplicação conhecer o domínio dos dados utilizando a camada de dados apenas para realizar acesso ao banco MySQL.

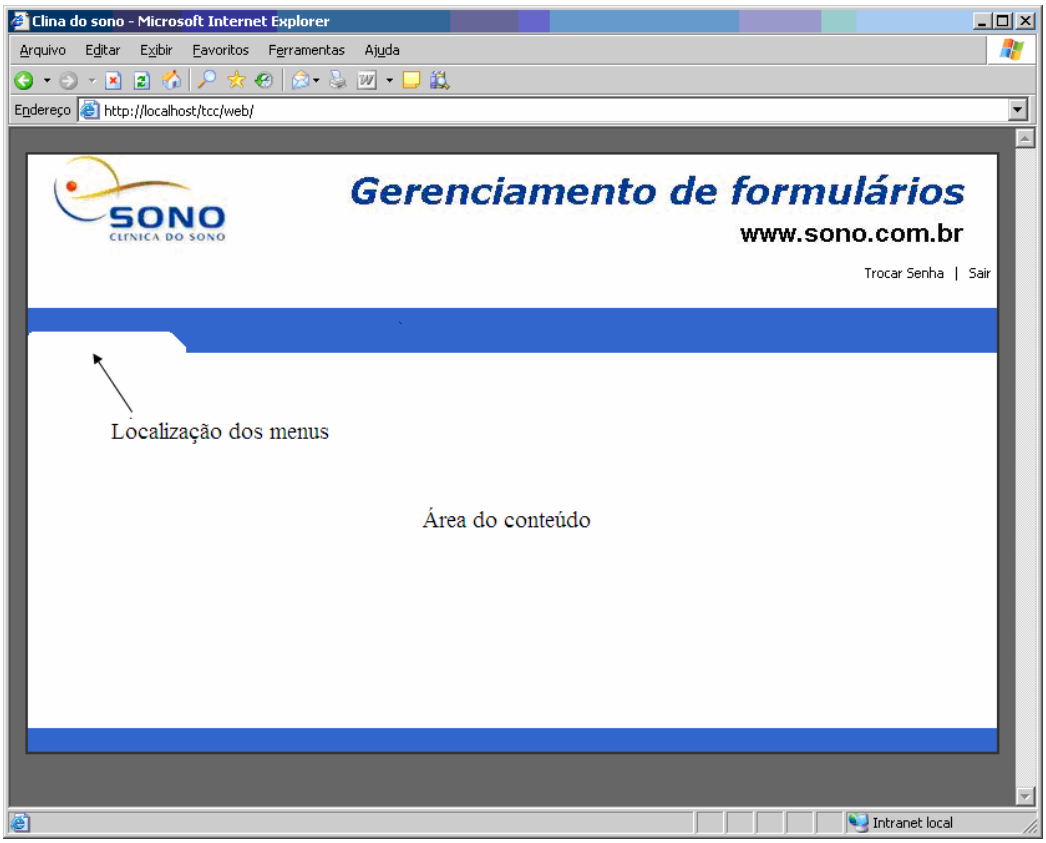

Figura 3.1: *Template* principal do sistema.

A Figura 3.2 ilustra uma página do sistema com um formulário instanciado. São definidos os rótulos de cada entrada de dado à esquerda, e na direita são disponibilizados os controles de inserção de dados.

Figura 3.2: Página com um formulário instanciado.

A Figura 3.3 ilustra uma página do sistema com um *grid* instanciado. Consiste em uma tabela onde a primeira linha descreve o cabeçalho, ou simplesmente o nome das colunas e as demais linhas as informações tabuladas.

Figura 3.3: Página com um *grid* instanciado.

3.2 Estrutura das classes

O Sistema de Gerenciamento de Formulários, como descrito anteriormente, foi implementado utilizando a linguagem PHP. Contudo, para tornar a experiência com o usuário mais atrativa, foi utilizada a linguagem de *script* Javascript. Assim, os arquivos da camada de aplicação foram implementados em PHP e os arquivos da camada de visualização foram construídos com HTML/Javascript.

As seguintes classes foram definidas na construção do sistema:

- ParsePage (PHP)
- BasePage (herdada de ParsePage)
- GridPage (herdada de BasePage)
- GridControlPage (herdada de GridPage)

- Connection (PHP)
- DataSet (PHP)
- TFieldSet (Javascript)

A classe ParsePage realiza a leitura de um *template* e, através das funções SetValue substituem as variáveis do *template* pelos valores correspondentes. Um exemplo pode ser visto na listagem a seguir, na qual substitui um campo do *template* por um valor qualquer:

```
//No template
<b>{campo}</b>

//instanciado o template na variável $tpl
$tpl->SetValue('{campo }', "Um valor qualquer");
```

A classe BasePage implementa os atributos comuns do sistema. Isto é, define as diretivas base com os componentes de visualização durante a sua construção além de realizar o *parsing*[13] dos elementos comuns: links, menus e título padrão da interface visual.

A classe GridPage implementa funcionalidades para a leitura e exibição de *grids* na área de conteúdo. Um *grid renderiza* uma tabulação de dados. A principal função desta classe é a AddRow, na qual insere um registro no *grid*.

A classe GridControlPage, que herda funcionalidades da classe GridPage implementa a construção de controles HTML *input text*, *radiobuttom* e *select*. Esta classe é responsável pela *renderização* dos controles de entrada de dados dos formulários dinâmicos.

A classe Connection fornece uma API de acesso ao banco de dados MySQL e a classe DataSet implementa funcionalidades para o armazenamento de dados oriundos de uma consulta SQL.

Por fim, tem-se a classe TFieldSet. Esta classe, construída em Javascript, é responsável pela lógica de criação e manipulação dos formulários dinâmicos. Ela implementa funcionalidades de criação dos controles (*input text, radiobuttom e select*), armazena informações associadas a cada controle criado (etiqueta de exibição, texto de ajuda, ordenamento do controle, *flags* e referencia entre campos).

Na prática, a classe TFieldSet manipula os atributos dos campos do formulário dinâmico. Os atributos podem ser visto na Figura 3.4.

---

[13] A ação de *parsing* é justamente localizar uma cadeia de caracteres num texto de entrada e realizar alguma ação sobre a cadeia no texto de entrada. Neste caso, há a substituição da cadeia localizada por outro valor.

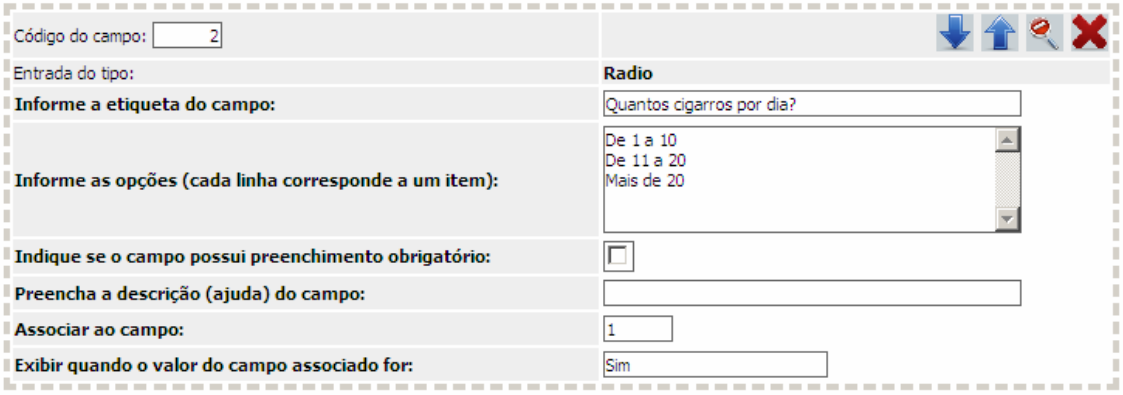

Figura 3.4: Configuração de entrada de dados (campo) de um formulário dinâmico.

3.3 Modelo da base de dados

O modelo necessário para o desenvolvimento do sistema consiste no conjunto mínimo de tabelas de cadastros básicos e uma estrutura genérica para comportar a construção de formulários dinâmicos.

Desta forma, para sustentar a base do sistema de formulários dinâmicos, foi modelada uma tabela para o cadastro de pacientes e uma tabela para o cadastro de usuários do sistema. Estas duas tabelas não têm relevância direta para o sistema, mas são necessárias uma vez que cada formulário dinâmico terá seu preenchimento associado a um paciente e para poder acessar o sistema é necessário ser um usuário do mesmo.

Por outro lado, o sucesso da implementação de um sistema de gerenciamento de formulários dinâmicos está intimamente ligado ao modelo da base de dados, pois um formulário dinâmico necessita de um pequeno dicionário de dados associado a ele. Assim, devem-se armazenar informações do formulário gerado e de cada campo que o irá compor.

Para o formulário, foi indicado apenas um nome e uma categoria. Esta categoria tem como objetivo auxiliar na hora de catalogar ou associar um determinado formulário aos pacientes. Por exemplo, um formulário que retrata um questionário específico para o sexo feminino poderia estar associado a uma categoria específica para este fim.

Já para os campos, foram observados uma etiqueta, um tipo e uma ordem. Assim como na definição de um campo de uma tabela, um campo do formulário dinâmico tem um nome, um tipo de dado associado e uma ordem de exibição. Outros atributos foram modelados e podem ser vistos nas figuras que seguem.

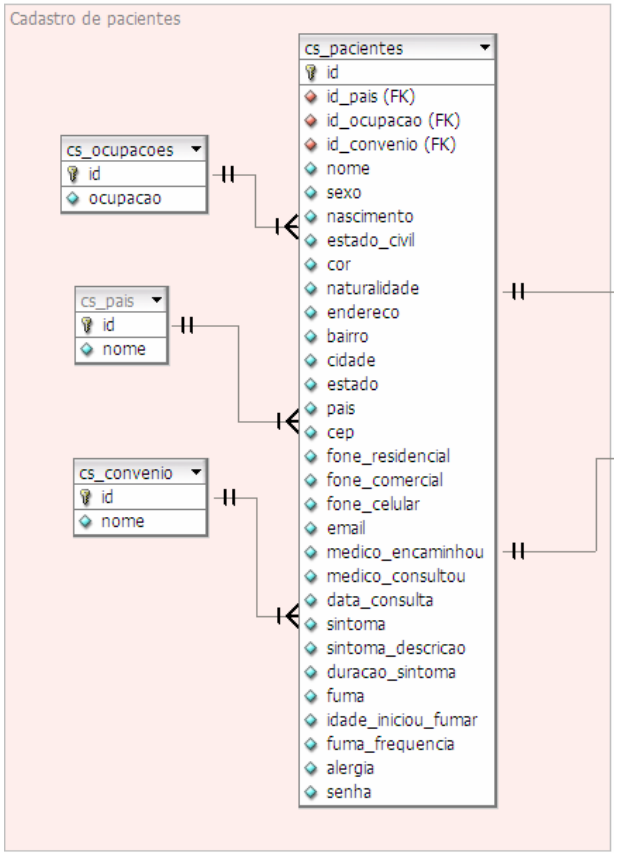

Figura 3.5: Modelo de dados dos cadastros básicos.

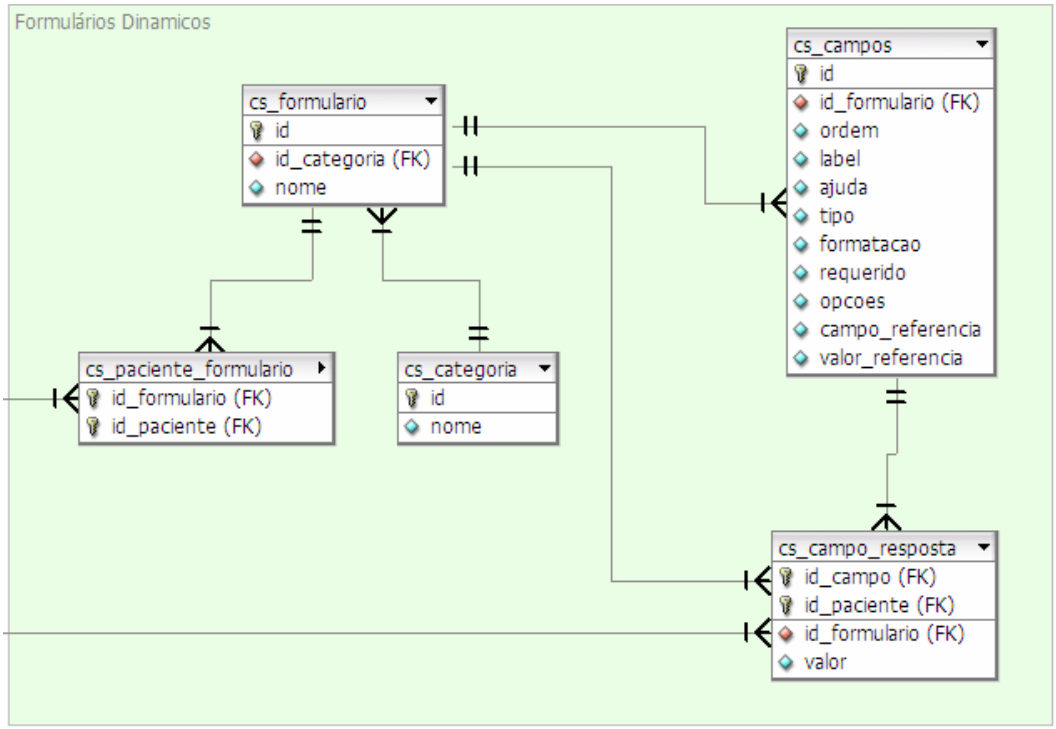

Figura 3.6: Modelo de dados da estrutura de formulários dinâmicos.

O Apêndice B detalha o dicionário de dados das tabelas relacionadas ao gerenciamento de formulários dinâmicos.

Cabe salientar que na tabela *cs_campo_resposta* o campo *id_formulario* aparece de forma redundante. Isto foi feito para garantir maior performance na busca pelos dados inseridos vinculados a um determinado formulário.

3.4 Detalhes da implementação

Como padrão de um sistema Web, característico de uma Intranet[14], o primeiro passo é acessar o sistema. A primeira página, portanto, apresenta uma tela de *login* que solicita os valores de usuário e senha. Esta página apresenta apenas as características da identidade visual, sem utilização das estruturas de *templates*, como mostra a Figura 3.7. Ao inserir as informações de usuário e senha, o sistema realiza a autenticação do usuário e cria uma sessão no qual fica registrado.

Uma vez autenticado, o sistema apresenta a seguinte estrutura de construção:

1. pagina.php
    a. carga do *template* associado
    b. consulta à base de dados (se necessário)
    c. substituição dos valores dos campos do *template*
2. Exibição do HTML
    a. carga das funções Javascript (se necessário)

[14] Conforme Bennet (2007), “Intranet consiste em uma rede privativa de computadores que se baseia nos padrões de comunicação de dados da Internet pública”. O autor afirma ainda que uma Intranet não é uma tecnologia, mas sim um conceito de organizar e estruturar um sistema baseado em uma rede.

Figura 3.7: Tela inicial do sistema.

Esta estrutura é padrão e toda página que compõe o sistema está baseada neste modelo. Assim, para cada página existe um arquivo PHP, controlando sua construção e exibição, e um arquivo HTML, definindo sua interface visual, com base no *template* padrão. Ainda, no arquivo HTML associado, é implementado, quando necessário, funções Javascript para tratar do comportamento visual, bem como lógica de aplicação do lado do cliente.

Na listagem que segue, há um exemplo de construção de uma página do sistema:

```
//arquivo inicial.html
<b>Olá, {usuario}! Bem vindo ao sistema</b>

//arquivo inicial.php
$tpl = new BasePage();
$tpl->SetTitle('Página inicial do sistema');
$tpl->SetContentFromFile('{page_content}', '_pages/inicial.html');
$tpl->SetValue('{usuario}', 'Carlos Alves');
$tpl->OutPut();
```

Conforme a listagem anterior, têm-se dois arquivos principais: inicial.html e inicial.php. O primeiro possui apenas um trecho de código HTML, incluindo uma variável de substituição `{usuario}`. Já, no arquivo inicial.php, tem-se inicialmente a criação de um objeto da classe BasePage. No construtor desta classe há a leitura do *template* principal, que ajusta os elementos padrão de todas as páginas. Em seguida, é carregado o conteúdo da página, através do método `SetContentFromFile`, que carrega o arquivo inicial.html. No *template* principal, existe uma variável de

substituição chamada `{page_content}`, que trata justamente da área de conteúdo da página.

Uma vez definidos os parâmetros básicos da página, basta ajustar os demais campos. No caso deste exemplo, foi ajustado o nome do usuário para o valor *Carlos Alves*.

A classe BasePage apresenta uma estrutura muito simples, sendo que, para exibição de *grids* com dados tabulados, como por exemplo, uma listagem de pacientes ou de usuários, foi implementada a classe GridPage. Esta classe, que herda as funcionalidades da classe BasePage, possui um método adicional chamado `AddRow`, que insere uma linha no *grid*. Além disto, a classe exige que o arquivo HTML associado possua uma estrutura diferenciada, como mostra a listagem a seguir:

```
//arquivo grid_pacientes.html
<table>
  <tr>
    <td>Código</td>
    <td>Nome</td>
    <td>E-Mail</td>
    <td>Telefone</td>
    <td> </td>
  </tr>
  {row}
  <tr>
    <td>{pcod} </td>
    <td>{nome} </td>
    <td>{email} </td>
    <td>{fone} </td>
    <td><a href="{actEditar}">Editar</a></td>
  </tr>
{endrow}
</table>
```

No exemplo da listagem anterior, foi definida uma tabela HTML, onde a primeira linha representa o cabeçalho do *grid*. Neste caso, tem-se as colunas *Código*, *Nome*, *E-mail* e *Telefone*. A próxima linha define a linha de dados e deve estar envolvida pelos delimitadores `{row}` e `{endrow}`. Ainda, deve apresentar o mesmo número de células do cabeçalho.

Cada célula presente na linha de dados possui uma variável de substituição que irá receber o valor indicado pelo arquivo PHP durante a construção da página. Deste modo, a próxima listagem demonstra a forma com a qual a página de listagem de pacientes é construída:

```
//arquivo listagem_pacientes.php
$tpl = new GridPage(‘grid_pacientes.html’);

$pac_lista = new DataSet();

$apac_lista->Open(“select * from cs_pacientes”);

while($x = $pac_lista->Next()){
      $dado = array(
        '{pcod}'  => $x['codigo'],
        '{nome}'  => $x['nome'],
        '{email}' => $x['email'],
        '{fone}'  => $x['fone'],
        '{actEditar}' => 'editar_paciente.php?idp='.$x['id'],
        '{actSenha}'  => 'execute.php?idp='.$x['id'].'&codigo=X',
        '{actForms}'  => 'associar_forms_paciente.php?idp='.$x['id']
      );
      $p->AddRowData($dado);
}
$tpl->OutPut();
```

Conforme a listagem anterior, o construtor da classe GridPage exige apenas que seja indicado o arquivo HTML associado que define a estrutura do *grid*. Após instanciar o objeto, é realizada uma busca na tabela de pacientes, no banco de dados e, para cada registro, é criado um vetor onde o índice é dado pela variável de substituição e o valor associado ao índice é o próprio valor que será exibido.

Finalmente, a classe GridControlPage apresenta as mesmas funcionalidades da classe GridPage, contudo, pode-se criar uma linha do *grid* associando um controle de entrada de dados. Isto é feito chamando o método `AddControl`, que, conforme os parâmetros indicados, cria diretamente toda a estrutura da linha. Esta classe é utilizada para construir a interface visual dos formulários dinâmicos, uma vez que cada linha do *grid* exibe uma etiqueta e um controle que pode ser do tipo entrada de texto, botão rádio ou caixa de seleção. Um exemplo deste *grid* pode ser visto na Figura 3.8.

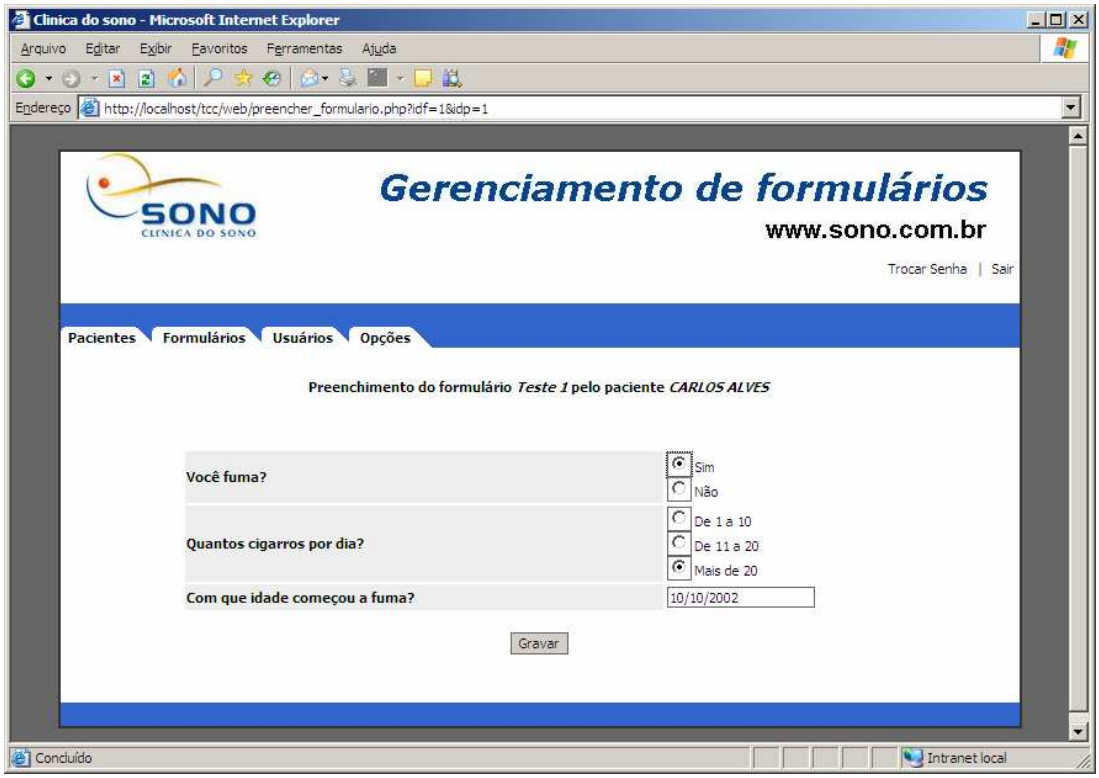

Figura 3.8: Mescla de grid com formulário, gerado a partir da classe GridControlPage.

Já do lado do cliente, existem algumas funções Javascript que enriquecem a experiência do usuário na utilização do sistema. Estas funções vão desde pequenas validações nos formulários de cadastros até a implementação da classe TFieldSet que gerencia toda configuração dos campos dos formulários dinâmicos. Pela importância desta classe, ela será descrita no Capítulo 3.5.

A necessidade de se trazer para o lado do cliente uma parte da lógica da aplicação é de fundamental importância dado que diminui a carga de comunicação cliente/servidor, isto é, gera menor carga no tráfego de informações entre o *browser* do usuário e o servidor do sistema, conforme Crane (2007) descreve em sua obra. Além disto, por se tratar de uma página HTML, não há um ambiente consistente e bem resolvido para realizar a interação com usuário, deixando assim a cargo do programador todo o trabalho de moldar o comportamento da interface. E isto só é possível graças à linguagem Javascript[15].

Todas telas que compõem a interface gráfica possuem um conjunto de funções Javascript que é carregada automaticamente através do *template* principal do sistema. Além disto, cada arquivo HTML que define uma tela do sistema pode ter um trecho de código Javascript que implementa as funcionalidades específicas do mesmo.

[15] Em verdade, existem outras linguagens de script que possibilitam enriquecer as páginas HTML. A mais comum é a Javascript e sua especificação pode ser encontrada no site http://www.ecma-international.org/publications/standards/Ecma-262.htm, mas pode-se facilmente realizar as mesmas construções com VBScript (http://msdn.microsoft.com/en-us/library/t0aew7h6(VS.85).aspx) ou JScript (http://msdn.microsoft.com/en-us/library/yek4tbz0(VS.85).aspx), por exemplo.

3.5 Estrutura de construção de formulários dinâmicos

Até agora, foram descritos os aspectos mais gerais da implementação do sistema. Contudo, a funcionalidade que realmente justifica este estudo e traz uma colaboração mais efetiva é a capacidade de prover uma interface gráfica simples e prática para a elaboração de formulários que possibilitam ao usuário criar seus próprios questionários de pesquisa.

Para compor esta funcionalidade, foi preciso modelar uma base de dados capaz de armazenar um pequeno dicionário de dados, que traduzisse os tipos de entrada de dados informados pelo usuário e o formato das possíveis respostas. Assim, surgiu o modelo de dado descrito no Capítulo 3.3.

Do lado do servidor, os arquivos PHP apenas carregam a interface visual correspondente à tela de construção do formulário e realizam o acesso ao banco de dados para gravar ou atualizar as informações inseridas pelo usuário.

Quem de fato implementa as funcionalidades de configuração de um formulário dinâmico é a lógica implementada na classe **TFieldSet**, codificada em Javascript. Esta classe possui funcionalidades que atendem aos seguintes requisitos:

- Criar um campo do formulário (entrada de dado);
- Configurar o aspecto visual do campo;
- Estabelecer relacionamentos entre os campos:
  - Um campo só é exibido conforme a resposta de um outro campo;
- Posicionar um campo:
  - Mover um campo para cima;
  - Mover um campo para baixo;
- Exibir/ocultar informações (atributos) de um campo na tela de gerenciamento do formulário;
- Excluir um campo;
- Submeter para o servidor os dados relativos às informações configuradas.

Estas funcionalidades rodam no *browser* do usuário. Portanto, até que sejam enviados os dados ao servidor, tem-se apenas as configurações locais realizadas pelo usuário. Este comportamento, ao mesmo tempo em que agiliza o processo de configuração, traz um inconveniente: caso ocorra algum erro durante a construção do formulário dinâmico, como perda de conexão com a internet ou fechamento do *browser*, as alterações feitas até então podem ser perdidas. Assim, recomenda-se que os campos sejam primeiramente pensados e desenhados para depois serem de fato construídos no

sistema. Ainda, é possível, após construir o formulário, editá-lo, mesmo que este já possua algum preenchimento posterior.

A classe TFieldSet gerencia tanto a exibição das áreas para a configuração dos controles quanto os tipos de controles criados bem como seus atributos. Seus principais métodos são:

- **Construtor**: recebe como parâmetro um *container*[16] onde os campos serão exibidos;
- **LoadTemplate**: recebe a identificação de um *template* de um campo e retorna o elemento visual correspondente ao controle desejado (neste caso, o *template* corresponde a construção HTML que exibe um pequeno formulário onde é possível configurar os atributos de cada campo);
- **CreateControl**: cria um controle de entrada de dados, isto é, um elemento com os atributos gerais e os atributos pertinentes a cada tipo de controle. Podem ser entrada de texto, radio ou caixa de seleção;
- **AddControl**: recebe o tipo de controle a ser instanciado, cria o controle, define o leiaute de configuração deste controle e o instancia no *container* definido no momento da criação do calsse;
- **RemoveControl**: remove o controle da lista de campos do formulário e retira o leiaute referente à configuração deste controle do *container*;
- **MoveUp**: move um controle para cima na lista de controles, tratando o re-ordenamento dos demais controles e mantendo as referencias já configuradas pelo usuário;
- **MoveDown**: move um controle para baixo, tratando o re-ordenamento dos demais controles e mantendo as referências já configuradas pelo usuário.

Os atributos de cada campo de um formulário dinâmico são os seguintes:

- **Etiqueta**: o nome do campo; pode ser entendida como sendo a pergunta de um questionário;
- **Tipo de entrada de dado**: texto, botão rádio ou caixa de seleção;
- **Obrigatoriedade de preenchimento**: indica se um determinado campo deve ser preenchido ou pode ser deixado sem valor algum;
- **Descrição (ajuda)**: texto que auxilia no preenchimento, dando dicas ou simplesmente explicando o dado que deve ser inserido;
- **Campo associado**: indica o código de um segundo campo ao qual se deseja associar um primeiro campo;
- **Valor do campo associado**: indica o valor do campo associado que quando for escolhido faz com que o campo em questão seja exibido.

---

[16] No contexto do sistema, o *container* é uma área que delimita e agrega demais controles HTML. Geralmente são *tags* " ", identificadas por um atributo *id*.

De acordo com o tipo de entrada de dado de um determinado campo, outros atributos podem surgir e são descritos nos Capítulos 3.5.1, 3.5.2 e 3.5.3.

Conforme a Figura 3.9, pode-se criar um novo campo através dos links ao final da página de configuração de um formulário. Cada um dos três links instancia um campo com um tipo de entrada de dado. O processo de como configurar e gerenciar um formulário dinâmico é descrito no Capítulo 4.

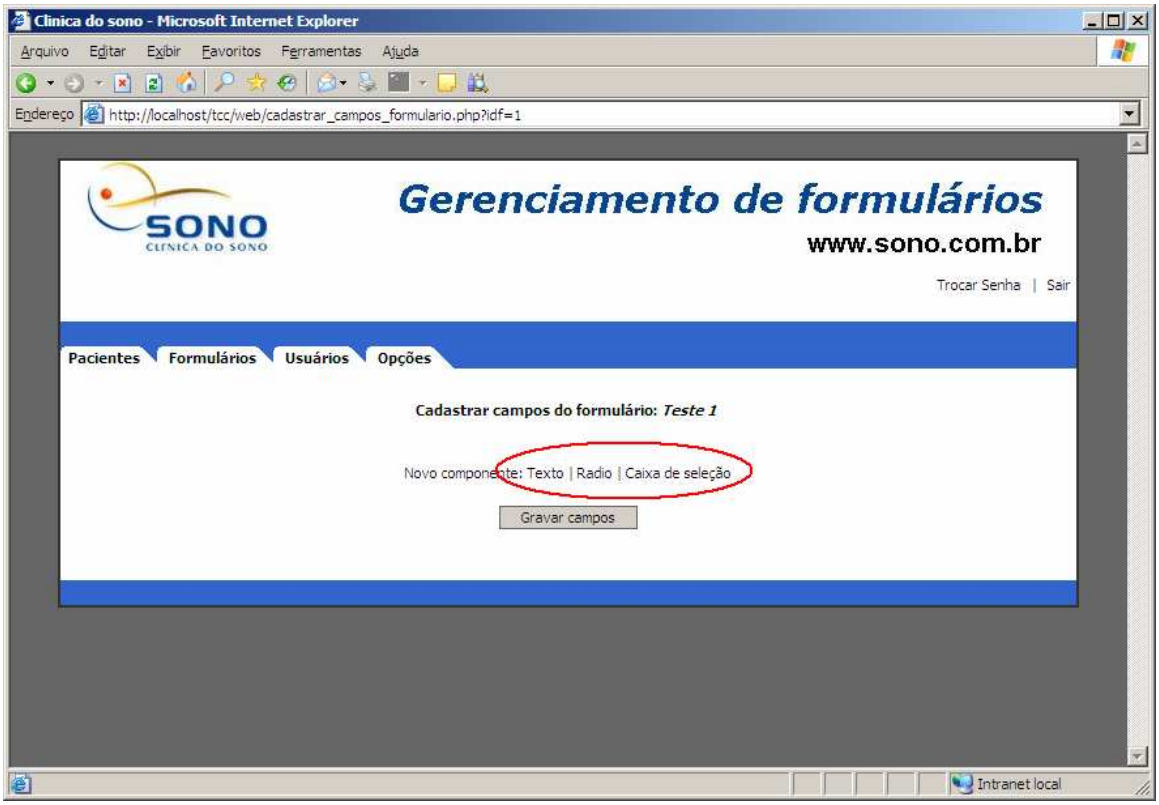

Figura 3.9: Página inicial de configuração de campos de um formulário dinâmico.

Existem 3 tipos de controles que podem ser associados a cada campo de um formulário dinâmico: Texto, Botão Rádio e Caixa de seleção no estilo “Lista Suspensa”.

3.5.1 Entrada de dados do tipo texto

Os campos de um formulário com controle do tipo texto permitem ao usuário digitar a informação de forma livre. Porém, pode-se designar três tipos de validação no momento do preenchimento: aceitar todo e qualquer conteúdo escrito pelo usuário (Alfanumérico), permitir que seja digitado apenas valores numéricos (Apenas números) ou permitir somente entradas do tipo data (Data). No caso de validar somente números ou somente datas, o usuário se depara com impossibilidade de digitar letras ou qualquer outro símbolo que não seja numérico. E, no caso de campos que só aceitem datas, existe uma máscara para deixar a informação no formado “DD/MM/AAAA”, onde DD representa o dia com dois dígitos, MM representa o mês com dois dígitos e AAAA representa o ano com quatro dígitos.

A Figura 3.10 ilustra o leiaute de configuração de um campo com entrada do tipo texto.

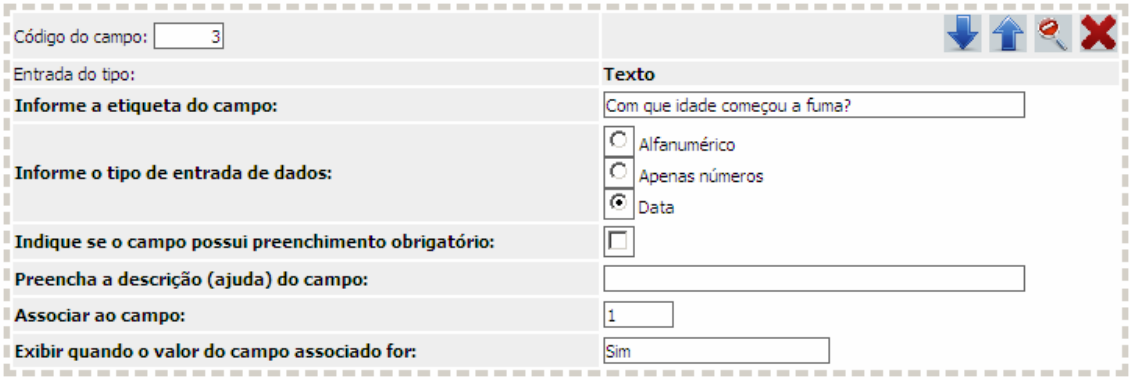

Figura 3.10: Entrada de dados do tipo texto.

3.5.2 Entrada de dados do tipo botão rádio

Este controle permite ao usuário escolher um dentre dois ou mais valores possíveis. Os valores possíveis, descritos como opções na interface visual, são preenchidos em forma de uma lista, separados por quebras de linha. A característica fundamental de controles do tipo rádio é permitir ao usuário que escolha somente um único valor de resposta. A Figura 3.11 ilustra a forma de exibição deste controle.

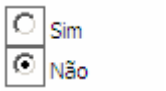

Figura 3.11: Controle de entrada de dados do tipo Botão rádio.

Já a Figura 3.12 ilustra o leiaute de configuração de um campo com entrada do tipo botão rádio.

Figura 3.12: Entrada de dado do tipo radio.

3.5.3 Entrada de dados do tipo caixa de seleção (lista suspensa)

Assim como os controles do tipo botão rádio, este controle permite escolher uma dentre duas ou mais opções. Porém, diferentemente de botões rádio, todos os valores de opções ficam encapsulados em uma lista oculta, sendo exibida apenas no momento da seleção e de forma suspensa, como ilustra a Figura 3.13.

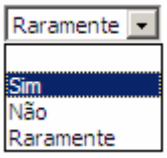

Figura 3.13: Controle de entrada de dados do tipo Caixa de seleção.

O leiaute de configuração de campos do tipo Caixa de seleção, assim como o leiaute de configuração de controles do tipo Botão rádio, demanda o preenchimento das opções a serem listadas. A Figura 3.14 ilustra o leiaute de configuração de um campo que contem o controle do tipo Caixa de seleção.

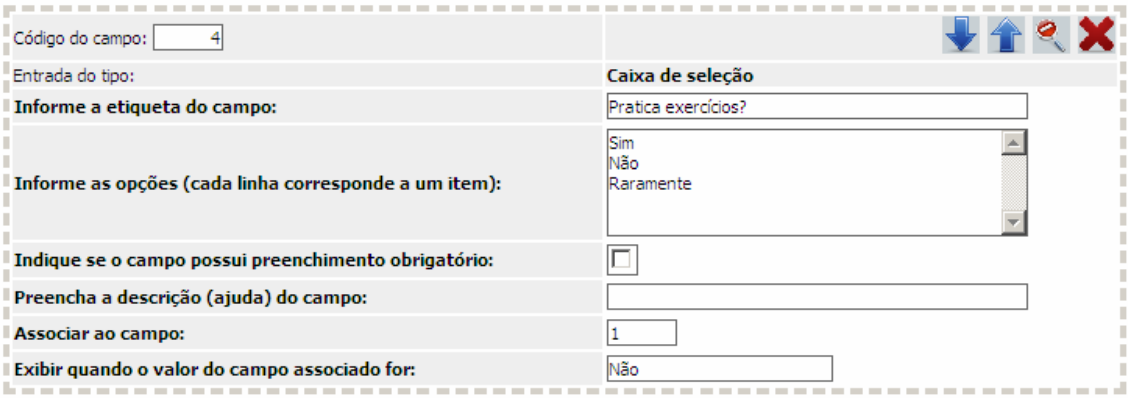

Figura 3.13: Entrada de dado do tipo caixa de seleção.

3.5.4 Atributos comuns de um campo do formulário

Para cada campo criado, é necessário definir, ao menos, uma etiqueta que, num primeiro momento, pode ser entendida como a pergunta de um questionário. Além disto, é possível indicar se o campo é obrigatório ou não. Se for obrigatório, durante o preenchimento do questionário, caso o usuário não informe nenhum valor, o sistema irá alertá-lo de que há a necessidade de informar um valor e não irá permitir que os dados sejam gravados.

Por fim, é possível associar um campo a um outro campo. Isto permite que, conforme a resposta do segundo, o primeiro seja exibido ou não. Para ilustrar esta funcionalidade, imagine que se a “pergunta 1” solicita que seja informado o Sexo do

paciente e, se a resposta for “Feminino”, então aparece logo abaixo outra pergunta que solicita a quantidade de partos que realizou – esta pergunta é conveniente apenas para pacientes do sexo feminino. Evita, portanto, que o usuário receba perguntas que não lhe dizem respeito e organiza melhor o fluxo de preenchimento.

Uma vez configurados os campos de um formulário dinâmico, este pode ser associado aos pacientes, o que é indicado pela tabela “cs_paciente_formulario”. Quando um formulário é associado a um paciente, este pode entrar na tela de preenchimento de formulários, realizando assim a inserção de dados. Estes procedimentos são explicados de forma detalhada no Capítulo 4.

4 INTERFACE COM USUÁRIO E OPERAÇÃO DO SISTEMA

Este capítulo descreve as telas do sistema, também chamadas de interface com o usuário. Além disto, é detalhado o fluxo de execução operacional do sistema. De uma forma geral, são demonstrados os passos e ações a serem feitas pelo usuário para que possa realizar as tarefas de gerenciar o cadastro de pacientes, bem como criar e gerenciar os formulários dinâmicos.

A primeira tela apresentada no sistema é a área de *login*. O acesso a esta tela é dado pelo endereço do sistema[17]. Nela, são informados os dados de *login* do usuário e a senha de acesso, como mostra a Figura 4.1.

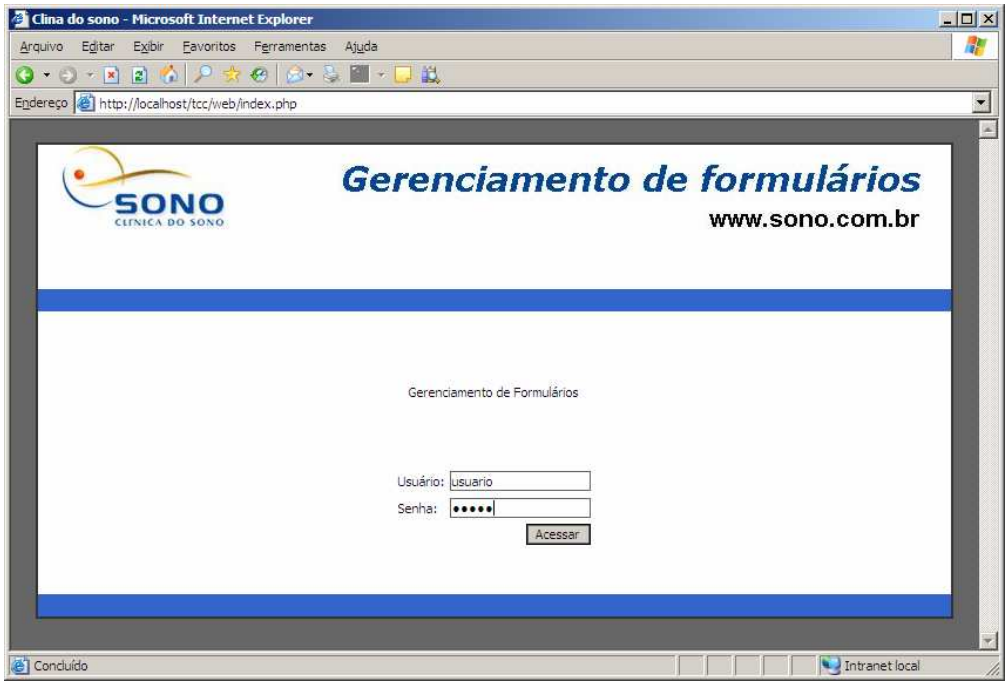

Figura 4.1: Tela de login.

Sendo informados valores de *login* de usuário e senha válidos, o sistema redireciona para a página inicial, conforme a Figura 4.2.

[17] Este trabalho está atualmente hospedado no endereço http://www.sono.net.br/sistema.

Figura 4.2: Usuário devidamente autenticado, na página inicial do sistema.

4.1 Interface de cadastro e edição de pacientes

Para cadastrar um paciente, basta acessar o menu **Pacientes** e clicar no sub-menu **Cadastrar**. A página exibida contém o formulário no qual é possível informar todos os valores relacionados ao cadastro de um paciente. O único campo de preenchimento obrigatório é o campo **Nome**. O campo **Código do paciente**, se deixado em branco, faz com que o sistema gere um código automaticamente e atribua ao novo cadastro[18]. Se for preenchido, força o sistema a cadastrar com o valor indicado, sendo gerado um erro caso o valor já exista na base. As figuras 4.3 e 4.4 ilustram este processo.

[18] A numeração dos códigos atribuídos pelo sistema é monotônica crescente. Para cada novo cadastro, o sistema verifica o maior número e incrementa em uma unidade.

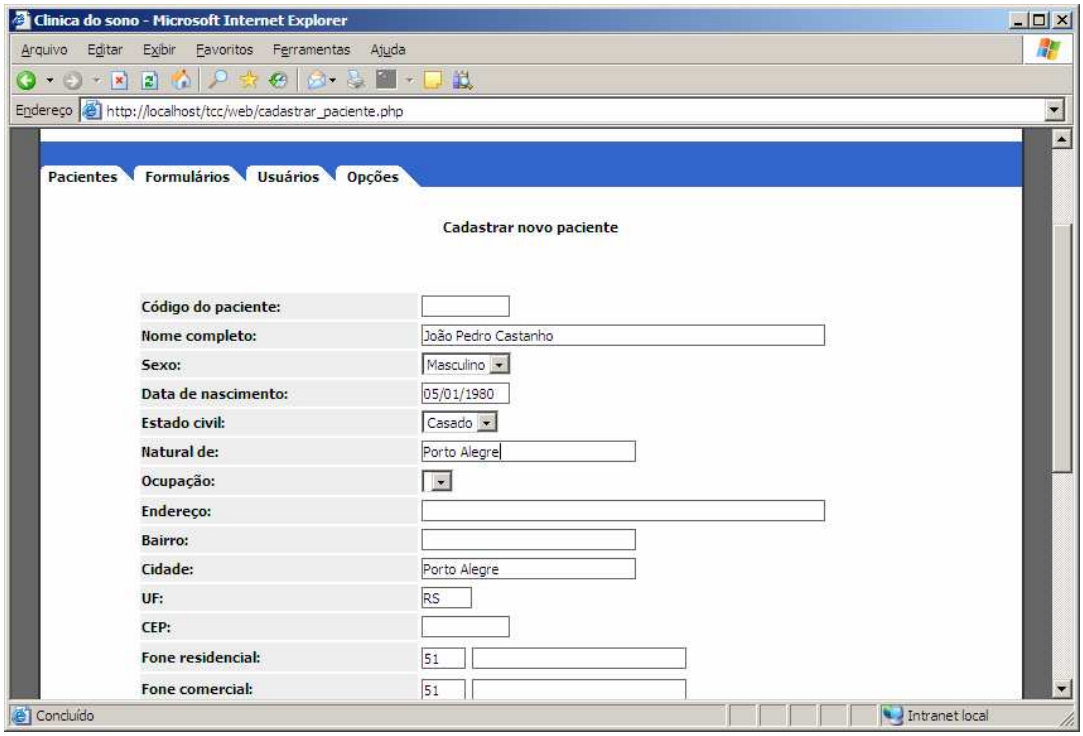

Figura 4.3: Tela de cadastro de pacientes.

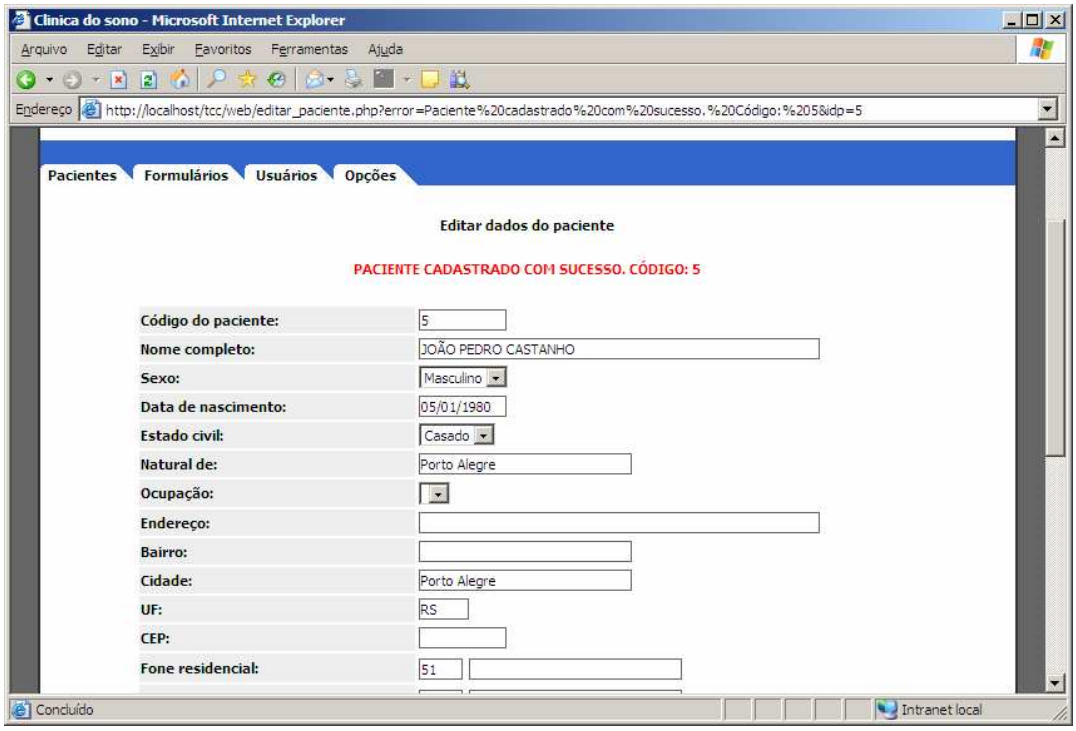

Figura 4.4: Após realizar o cadastro, o sistema informa o novo código.

Ao clicar no botão **Gravar Dados**, o sistema insere as informações no banco de dados informando que a operação ocorreu com sucesso (em caso de erro, o sistema exibe a mensagem de erro gerada) e indica o código do atribuído ao cadastro.

A edição do cadastro de pacientes é apresentada da mesma forma que a tela de cadastro após a inserção de um novo cadastro, ou seja, é o mesmo formulário, porém com os valores do cadastro carregados nos campos apropriados.

É possível mudar qualquer dado que se deseje relativo ao cadastro em edição. Entretanto, a mudança de código de um cadastro de paciente está condicionada à valores únicos, isto é, não é possível alterar o código de maneira que dois pacientes acabem possuindo o mesmo valor.

Outro detalhe importante, visto na Figura 4.5, é que na edição do cadastro de pacientes há a possibilidade de preencher questionários que tenham sido previamente construídos, através da funcionalidade de elaboração de formulários dinâmicos e que esteja disponibilizado para este paciente.

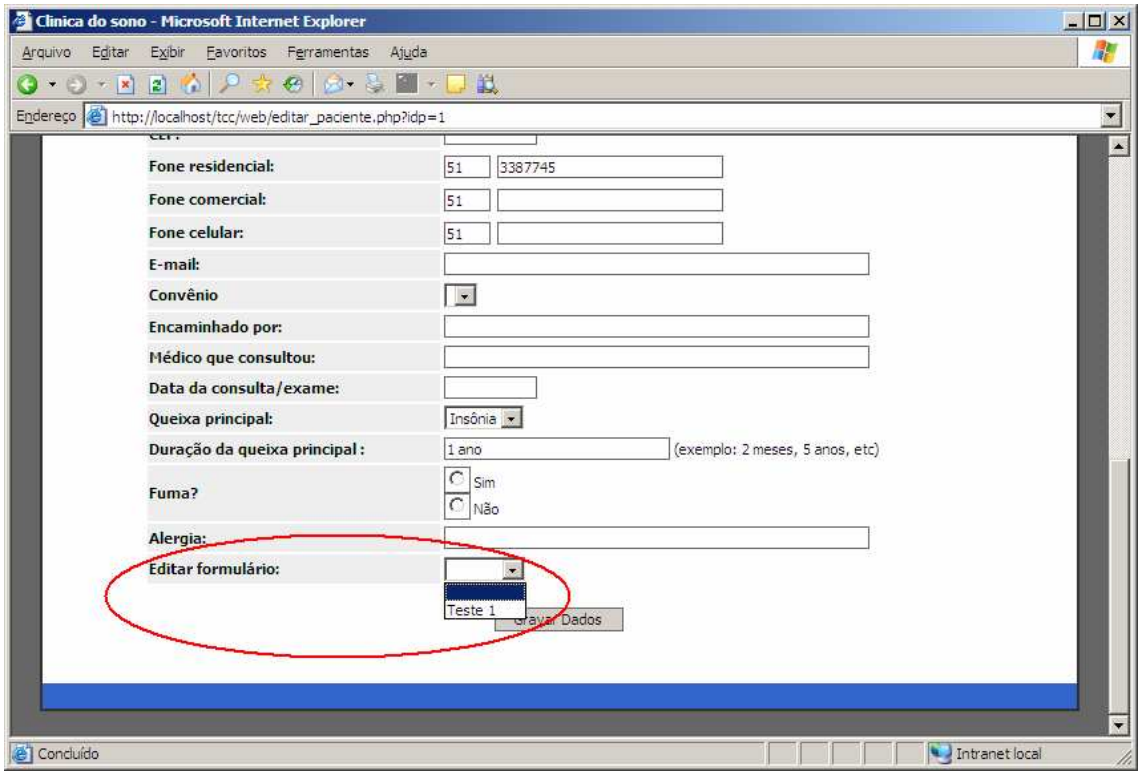

Figura 4.5: Listagem de formulários disponíveis para o preenchimento.

4.2 Interface de busca e listagem de pacientes

O sistema disponibiliza uma forma simples de listar e buscar pacientes cadastrados na base de dados. Através do menu **Pacientes** e sub-menu **Buscar**, é exibido um pequeno formulário no qual é possível informar alguns dados relativos ao filtro de busca de pacientes. Ao deixar todos os campos em branco no formulário de busca, todos os registros são listados. A figura 4.6 ilustra o formulário de busca.

Figura 4.6: Formulário de busca de pacientes.

Uma vez listado os pacientes em um *grid* (cada linha do *grid* corresponde a um registro do cadastro) tem-se duas ações principais associadas a cada registro disponíveis nos links **Editar** e **Associar Formulários**. O primeiro direciona para a página de edição de dados do cadastro de pacientes. Já o segundo link, leva à interface de associação de formulários a um paciente. Este último é descrito no Capítulo 3.9.

4.3 Interface de cadastro de formulário dinâmico

Esta funcionalidade é a mais complexa, porém, a mais importante do sistema. Para iniciar o processo de cadastro e configuração de um formulário dinâmico, basta acessar o menu **Formulários** e clicar no sub-menu **Cadastrar**. A primeira tela do processo solicita que seja preenchido o nome do formulário. Na prática, o nome do formulário implica no nome do questionário de pesquisa ou formulário de exame que será aplicado aos pacientes. Ainda, nesta mesma tela, é necessário informar a categoria a qual o formulário irá pertencer. Preenchidos os campos solicitados, deve-se clicar no botão **Cadastrar**.

A próxima tela que se abre, consiste na área de configuração dos campos que o formulário irá conter. Bem abaixo, ao final da página, existem três links que possibilitam criar os três tipos de entrada de dados: **Texto**, **Botão rádio** e **Caixa de seleção**.

Cada entrada de dado criada irá gerar um controle associado que será disponibilizado na tela de preenchimento. A entrada de texto configura uma *HTML Input*; a entrada do tipo botão de rádio configura um *HTML radionbuttom*; a entrada do tipo caixa de seleção configura um *HTML select*.

Para qualquer um dos três tipos é necessário informar uma etiqueta (descrição do campo). Para a entrada do tipo texto, deve-se informar que tipo de valores podem ser digitados: texto livre, somente números ou data. Para as entradas do tipo botão rádio e caixa de seleção, deve-se preencher as opções de resposta.

Opcionalmente, para cada um dos três tipos, existe uma *flag* que indica se o preenchimento é obrigatório, uma entrada para o texto de ajuda que auxilie ou explique o campo do formulário e uma referência para outro campo. Esta referência para outro campo, faz com que o campo atual seja exibido somente quando o campo referenciado possuir o valor informado.

Para compreender o processo de campos referenciados, tem-se o seguinte exemplo: o campo de código 1 é do tipo rádio e tem as opções “Sim” e “Não”. Para associar um outro campo de, por exemplo, código 2, ao campo de código 1, basta informar no campo de código 2 o valor 1 onde diz “Associar ao campo”. Se quiser que o campo de código 2 seja exibido quando o campo de código 1 contiver o valor “Sim” selecionado, basta informar “Sim” onde diz “Exibir quando o valor do campo associado for:” no campo de código 2. Para exibir conforme dois ou mais valores, basta separá-los por ponto-e-vírgula. A configuração do exemplo descrito é ilustrada na fig. 4.7.

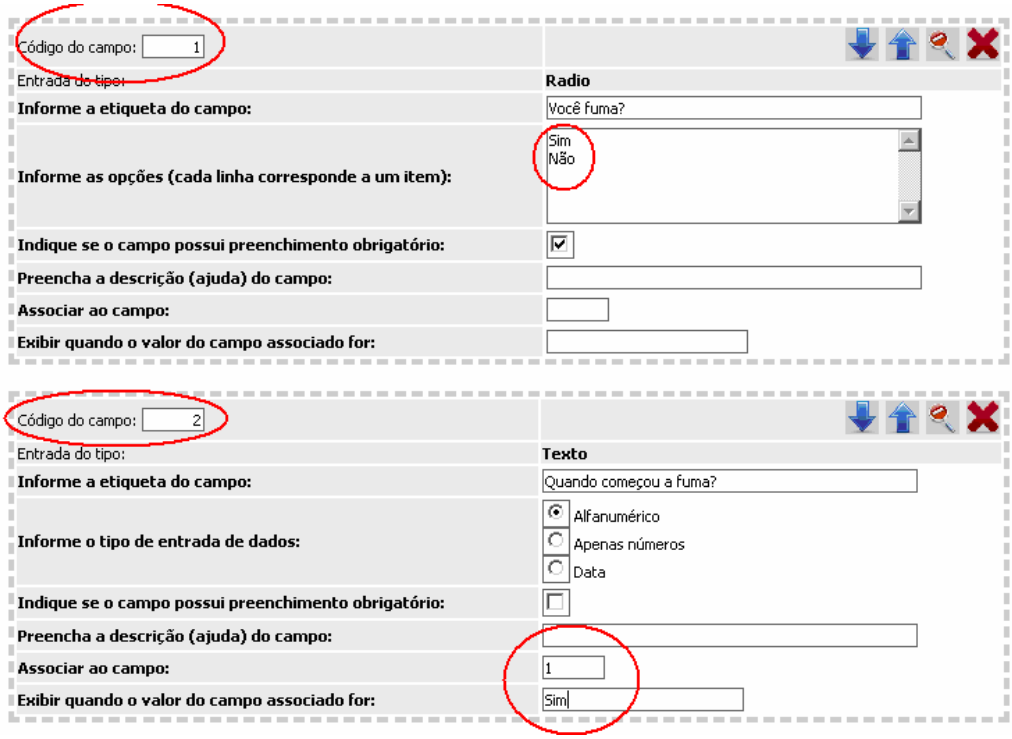

Figura 4.7: Campo 3 somente irá ser exibido se a resposta do campo 1 for “Sim”.

A área que guarda os atributos de cada campo do formulário possui 4 botões, em forma de imagens, que trazem outras funcionalidades de configuração. Os dois primeiros botões, da esquerda para direita, movimentam o campo para baixo e para cima, respectivamente. Isto implica na ordem em que os campos são exibidos na interface de preenchimento.

O terceiro botão, apenas colapsa a área de atributos para reduzir a quantidade de informações exibidas na tela. Por fim, o último botão, exclui o campo do formulário.

Uma vez definidos os campos do formulário, basta clicar no botão **Gravar campos**.

4.4 Interface de associação entre formulários e pacientes

Para que um formulário dinâmico possa ser preenchido, deve-se associá-lo aos pacientes. Este método visa organizar e disponibilizar para cada paciente somente aqueles formulários de interesse. Um caso de estudo levantado durante a discussão das funcionalidades é a existência de questionários que são aplicados somente a mulheres.

Existem duas formas de associar um formulário ao cadastro de um paciente:

- Para cada paciente do sistema, individualmente, podem ser associados um ou mais formulários;
- Para cada formulário, individualmente, podem ser associados um ou mais pacientes

Como bem pode ser observado, as duas maneiras são complementares, facilitando assim a forma com a qual se deseja relacionar pacientes com os formulários dinâmicos.

4.4.1 Interface de associação de um paciente a um ou mais formulários

Para relacionar um paciente específico aos formulários cadastrados, basta realizar a busca do paciente em questão e clicar no link Associar formulários. Com isto, a próxima tela exibe todos os formulários dinâmicos disponíveis e o usuário apenas marca aqueles que desejar associar, como mostra a Figura 4.8.

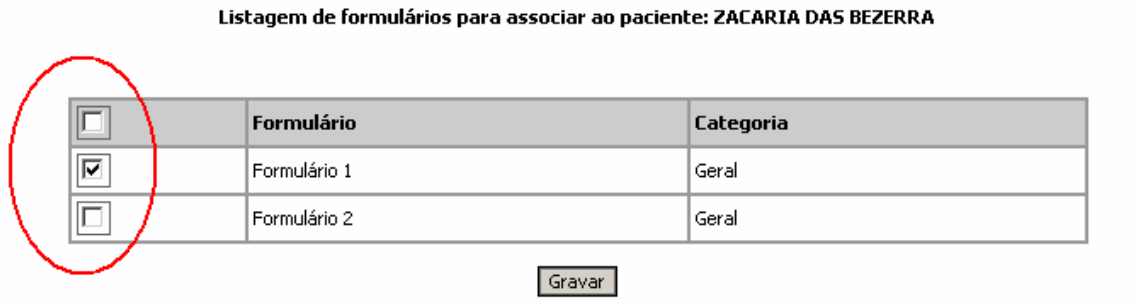

Figura 4.8: Tela que relaciona formulários a um paciente específico.

Após selecionar os formulários desejados, basta clicar no botão **Gravar**.

4.4.2 Interface de associação de um formulário a um ou mais pacientes

Para relacionar um formulário aos pacientes cadastrados no sistema, basta selecionar o menu **Formulários** e o sub-menu **Listar todos**. Desta forma, todos os formulários cadastrados serão exibidos em um *grid*. O próximo passo consiste em clicar no link Associar no registro do formulário desejado, como mostra a Figura 4.9.

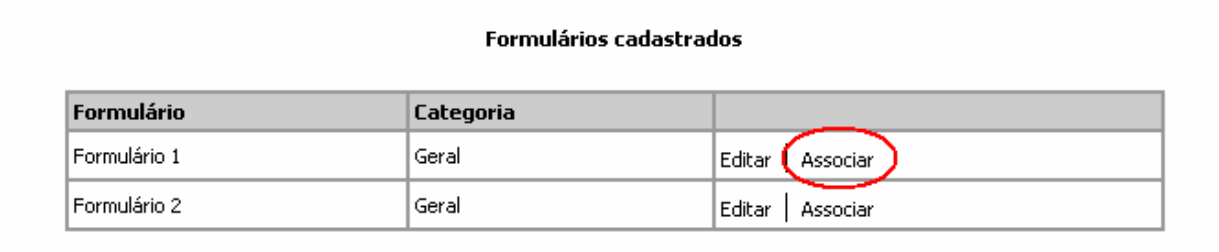

Figura 4.9: Listagem de formulários disponíveis.

Ao clicar no link, é disponibilizada a mesma tela de busca de pacientes cadastrados. Pode-se preencher qualquer campo para retornar um conjunto de cadastros de pacientes específico ou simplesmente deixar em branco todos os campos para que sejam listados todos os pacientes.

Ao clicar no botão **Buscar**, é exibida a listagem de pacientes com a qual pode-se selecionar quais registros desejar, como mostra a figura 4.10. Por fim, basta clicar no botão Gravar para que o processo seja concluído.

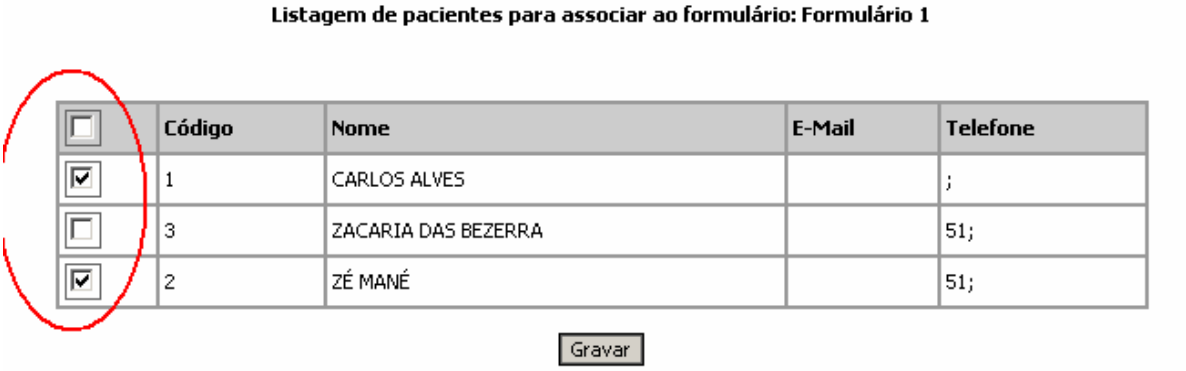

Figura 4.10: Listagem de pacientes que podem ser selecionados para associar ao formulário.

4.5 Interface de preenchimento de formulários dinâmicos

A última etapa consiste no preenchimento dos campos de cada formulário dinâmico com os dados relativos a cada paciente. Por ser relativo aos dados de um paciente, a forma de se chegar até o formulário é através do cadastro de pacientes.

Assim, estando na tela de edição dos dados de cadastro do paciente, o último controle do formulário de dados, denominado **Editar formulário**, dispõe todos os formulários atribuídos ao paciente em questão. Selecionando qualquer um que esteja disponível na caixa de seleção (Figura 4.11), o sistema redireciona para a interface de preenchimento.

Figura 4.11: Na tela de edição de cadastro de pacientes, pode-se selecionar qual formulário se deseja preencher.

A Figura 4.12 demonstra o leiaute da tela de preenchimento de um questionário. No exemplo da figura, o campo “Você fuma?” está indicado como sendo “Não”. Em decorrência desta resposta, logo abaixo é exibido outro campo com a pergunta “Pratica exercícios?”. Porém, se a resposta da pergunta “Você fuma?” fosse “Sim”, seriam exibidos outros campos, como pode ser observado na Figura 4.13.

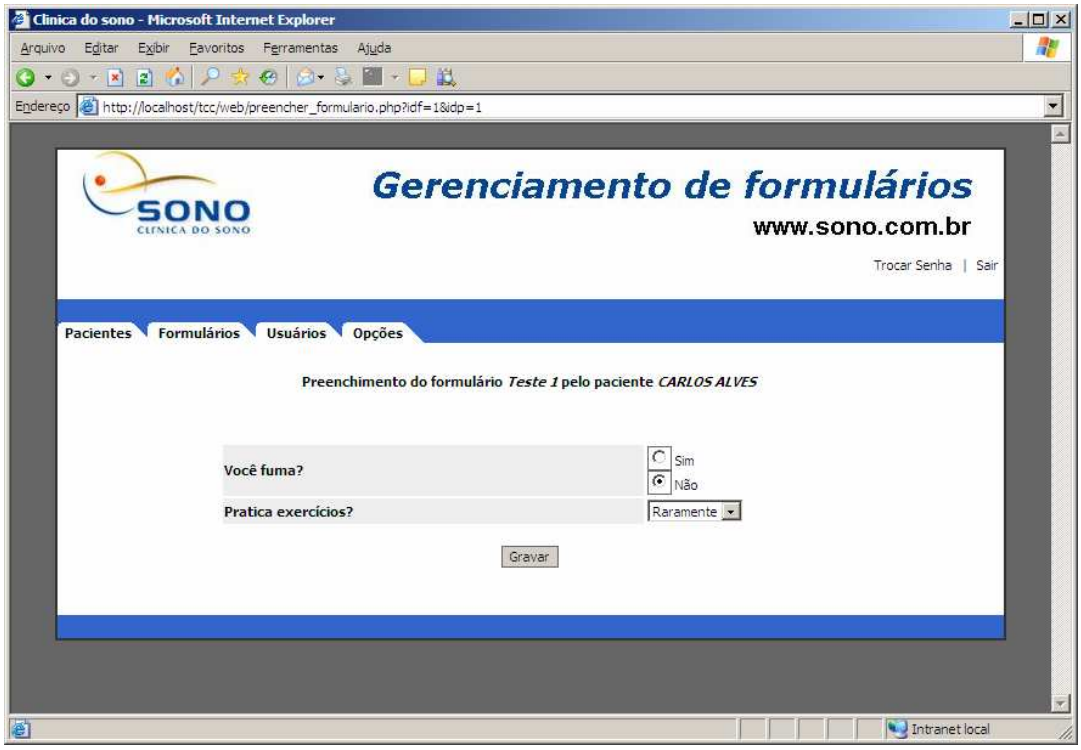

Figura 4.12: Exemplo de um formulário a ser preenchido.

A Figura 4.13 mostra outro exemplo de leiaute de preenchimento de um formulário dinâmico.

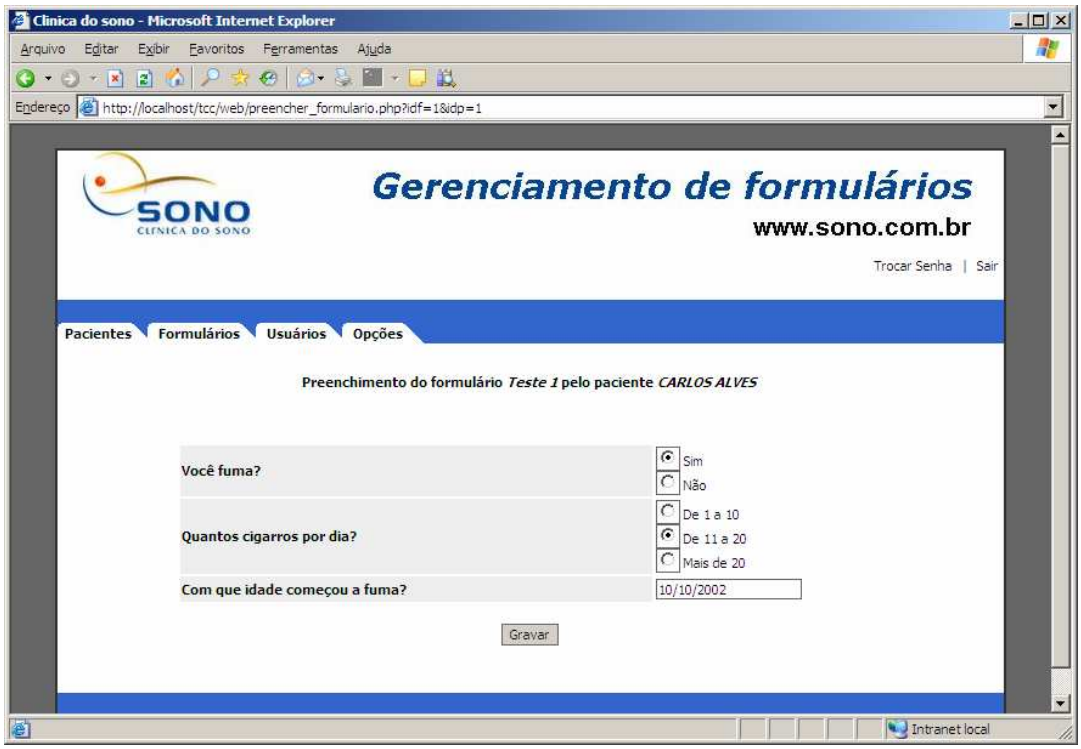

Figura 4.13: Exemplo de um formulário a ser preenchido.

O preenchimento consiste em inserir os dados de cada campo, de acordo com o solicitado. O processo finaliza ao clicar no botão **Gravar**.

5 AVALIAÇÃO DA FERRAMENTA

A proposta deste Capítulo é descrever o resultado obtido em função do esforço dedicado à elaboração deste Trabalho de Conclusão. Contudo, por se tratar de um projeto prático que nasceu com o objetivo de atender uma demanda real e necessária, nada mais justo do que trazer neste momento a avaliação e a satisfação do Dr. Denis.

A dinâmica associada ao estudo e elaboração do Sistema de Gerenciamento de Formulários seguiu alguns aspectos clássicos da Engenharia de Software, como reuniões com o Dr. Denis para a análise de requisitos, a confecção de alguns diagramas e modelos para que o problema pudesse ser bem entendido e, por fim, a construção de alguns protótipos para poder validar a solução apresentada.

Assim, dos requisitos levantados ao longo desse processo e da ferramenta final apresentada, o resultado obtido foi assumido como sendo muito bom e dentro do esperado, segundo as próprias palavras conferidas pelo Dr. Denis. Ainda, em E-mail enviado por ele, tem-se o seguinte:

> "Gostaria de deixar registrada a minha admiração pelo profissionalismo, dedicação e seriedade do Carlos [...] " (por correio eletrônico enviado a Carlos Alves e Leandro Wives na data de 30/10/2009)

Dos requisitos que não puderam ser atendidos, segundo a necessidade imposta pelo Dr. Denis, ficaram detalhes de leiaute, fluxo mais inteligente de associação entre formulários e pacientes, além da utilização de alguns recursos visuais mais elaborados, tais como calendários e caixas de seleção editáveis (Listas suspensas com possibilidade de entrada textual).

Outro requisito importante, também não atingido, foi a disponibilização do cadastro de pacientes para o grande público, ou seja, a implementação de um controle de acesso mais sofisticado conferido aos pacientes da Clínica a possibilidade de acessar o sistema para conferência de seus dados bem como o preenchimento dos questionários.

Finalmente, na data de 23/10/2009, o Dr. Denis validou e homologou o sistema, dentro daquilo que havia sido ora especificado, observando apenas a necessidade de se realizar os demais ajustes para que o sistema pudesse ganhar o campo da Rede Mundial, e não só apenas um ambiente de Intranet.

6 CONCLUSÕES

O trabalho realizado objetivou a construção de um sistema para o gerenciamento de formulários de entrada de dados para pacientes e dados de pesquisa para a área do sono. Em grande parte, o trabalho foi calcado na experiência e conhecimento do co-orientador Dr. Denis Martinez, o qual forneceu as informações necessárias para a construção do sistema.

Inicialmente, foi realizado um estudo com base no sistema atualmente utilizado – banco de dados Access – e uma breve análise nos métodos e processos adotados pelos funcionários e pacientes da clínica para que fosse possível desenhar e modelar um sistema de gerenciamento de formulários. A primeira etapa do trabalho constituiu na construção de formulários de cadastros de dados gerais, tais como cadastro de pacientes e cadastro de usuários. Isto foi útil, pois ajudou a compreender o modelo atual do domínio de dados.

Ao contrário do que se tinha pensado no começo, onde se queria apenas elaborar uma base para construção de diversos formulários e então gerenciá-los, surgiu a idéia, em parte sugerida pelo próprio Dr. Denis, de construir uma ferramenta capaz de dar ao próprio usuário o cerne da construção e elaboração dos formulários.

Estes formulários são estruturas que idealizam e implementam questionários com o qual o Dr. Denis aplica em seus pacientes. Com os dados coletados, são realizadas pesquisas diversas. Assim, viu-se o grande dinamismo associado a cada questionário, pois é muito comum o surgimento de novos questionários de pesquisa assim como ajustes posteriores ao seu preenchimento.

A idéia da construção de um sistema Web ganhou força quando se pensou em atribuir aos próprios pacientes a tarefa de preenchimento destes formulários. Uma vez que o sistema é disponibilizado na internet, qualquer usuário pode ter acesso de qualquer computador que tenha acesso à rede mundial.

Dentro de um fluxo de trabalho constante, gastando-se cerca de 6 a 8 horas por semana, para o entendimento do problema, através de entrevistas e reuniões dirigidas pelo Dr. Denis, e com o auxílio do orientador Leandro, foi possível modelar uma primeira versão que serviu como base para a construção do sistema final.

Ao final do processo de implementação e homologação, viu-se que a meta de chegar a disponibilizar o acesso do sistema ao grande público (funcionários e pacientes) seria uma tarefa praticamente inalcançável. De fato, o sistema como se imaginou não haveria como ser concebido por apenas um desenvolvedor, mas sim, com a integração de uma equipe em tempo integral trabalhando no sistema.

Dentre os requisitos levantados, não foi possível concluir a implementação do controle de acesso com níveis de usuários. Isto permitiria um maior controle sobre o uso do sistema. Outro requisito que não chegou a ser codificado foi a utilização de controles mais sofisticados de manipulação de dados, como por exemplo, calendários nas entradas de datas.

Ficando assim a sugestão para trabalhos futuros a reestruturação do leiaute com a inclusão de controles mais inteligentes e a integração com outras áreas da Clínica, como o controle de consultas e controle financeiro, como sugerido pelo Dr. Denis durante as reuniões.

Ainda, uma área de muito interesse relatada pelo Dr. Denis foi uma forma de poder gerar relatórios e realizar mineração de dados sobre os dados coletados, pois, o sistema desenvolvido neste trabalho teve como escopo a elaboração de formulários para facilitar somente a coleta de dados.

Por fim, no escopo de trabalhos futuros, sugere-se a construção de uma ferramenta capaz de exportar ou fornecer uma forma mais “amigável” no tratamento das respostas dos formulários dinâmicos. Isto pois, da forma como foi organizado o modelo de dados, os dados ficam condensados em apenas uma tabela num formato genérico.

Assim, entrega-se, através deste trabalho, o embrião de um sistema capaz de automatizar e ampliar a capacidade de gerenciamento de dados da Clínica do Sono. Como saldo, tem-se a experiência extremamente positiva adquirida com a vivência da busca de uma solução para um problema real.

REFERÊNCIAS

BENETT, Gordon. Intranets : como implantar com sucesso na sua empresa. Rio de Janeiro: Campus, 1997.

CONVERSE, T., PARK, J. PHP: a Bíblia. Rio de Janeiro: Campus, 2003. Tradução da 3º edição.

CRANE, D. Ajax em Ação. 1º Ed. São Paulo: Pearson Prentice Hall, 2007.

CURY, Antonio. Organização e métodos: uma visão holística. 8. ed. São Paulo: Atlas, 2005.

FREITAS (H.), JANISSEK (R.) e MOSCAROLA (J.). Dinâmica do processo de coleta e análise de dados via web. CIBRAPEQ - Congresso Internacional de Pesquisa Qualitativa, 24 a 27 de março, Taubaté/SP, 2004. 12 p.

SEBESTA, R. W. Concepts of programming languages. 7º ed. Boston: Pearson/Addison, 2005.

BIBLIOGRAFIA

FERREIRA, M. P. Implementação de um framework para desenvolvimento web utilizando PHP : Projeto de Diplomação. 2003.

GOODMAN, D. JavaScript – Bible. 5th ed. Indiana: Wiley Publishing, 2004.

LAFORE, R. Estruturas de Dados & Algoritmos. 1º ed. Rio de Janeiro: Ciências Modernas, 2004.

Manuais da linguagem de programação PHP através do endereço eletrônico: **http://www.php.net/docs.php**.

Manuais do banco de dados MySQL através do endereço eletrônico: **http://dev.mysql.com/doc/refman/5.1/en/**.

SOARES, W. Programando em PHP : Conceitos e Aplicações. 2 º ed. São Paulo : Érica, 2000.

Welling, L. PHP e MySQL : desenvolvimento web. 3º ed. Rio de Janeiro: Elsevier, 2005.

APÊNDICE A – ARQUIVOS E PASTAS DO SISTEMA

Neste apêndice, encontra-se a descrição da estrutura de pastas e a descrição dos arquivos do Sistema de Gerenciamento de Formulários Web, desenvolvidos neste trabalho.

A estrutura de pastas é a seguinte:

- Sistema
  - _img
  - _menus
  - _pages
  - _system

A tabela a seguir descreve cada arquivo presente no sistema:

Arquivo	Pasta	Descrição
associar_forms_paciente.php	Sistema	Exibe um grid com a listagem de todos os formulários que podem ser associados a um determinado paciente.
associar_formularios.php	Sistema	Exibe um grid com a listagem de pacientes para serem associados a um determinado formulário dinâmico.
autentica.php	Sistema	Ativada pela página de login, realiza a autenticação do usuário.
buscar_pacientes.php	Sistema	Exibe formulário de filtro para a busca de pacientes.
cadastrar_formulario.php	Sistema	Exibe formulário para cadastrar um formulário dinâmico.
cadastrar_paciente.php	Sistema	Exibe formulário para o preenchimento com os dados de cadastro de um paciente.
editar_campos_formulario.php	Sistema	Exibe a interface de configuração dos campos de um formulário dinâmico.
editar_pacientes.php	Sistema	Exibe formulário para a edição dos dados de cadastro de um paciente.
execute.php	Sistema	Este arquivo implementa todas as funções relativas ao controle do sistema.

		Todos os formulários do sistema tem suas ações controladas por este arquivo.
grava_campos.php	Sistema	Ativado pela interface de configuração de formulários dinâmicos. Realiza a gravação dos dados relativos à configuração de cada campo de um determinado formulário dinâmico.
index.php	Sistema	Página de login.
inicial.php	Sistema	Paginal inicial do sistema, após usuário realizar o login
listar_formularios.php	Sistema	Exibe um grid com a listagem de todos os formulários dinâmicos cadastrados.
listar_pacientes.php	Sistema	Exibe um grid com a listagem de pacientes.
logado.php	Sistema	Este arquivo está presente em todas as páginas do sistema que necessitam de autorização de acesso. Faz o controle de usuário logado.
preencher_formulario.php	Sistema	Exibe um formulário dinâmico para que seja preenchido.
menus.html	_menus	Template com os menus padrão do sistema. É instanciado na construção do template principal.
form_buscar_paciente.html	_pages	Codificação do formulário de filtro na tela de busca de pacientes.
form_cadastrar_campos.html	_pages	Codificação da interface de configuração de formulários dinâmicos.
form_formulario.html	_pages	Codificação do formulário de cadastro de um formulário dinâmico.
form_pacientes.html	_pages	Codificação do formulário de cadastro de pacientes.
grid_associar_forms_paciente.html	_pages	Codificação do grid de pacientes para associar a um determinado formulário.
grid_associar_fornulario.html	_pages	Codificação do grid de formulários para associar a um determinado paciente.
grid_campos_formulario.html	_pages	Codificação de um grid com os controles de inserção de dados de um formulário dinâmico.
grid_formularios.html	_pages	Codificação do grid de listagem de formulários dinâmicos cadastrados.
grid_pacientes.html	_pages	Codificação do grid de listagem de pacientes.
template.html	_pages	Template principal do sistema.
all.js	_system	Funções Javascript que implementam o comportamento da interface do lado do cliente.
FormFieldAPI.js	_system	Codificação da classe TFieldSet.
consts.php	_system	Constantes utilizadas no sistema.
db.php	_system	Codificação das classes Connection e DataSet.
funcoes.php	_systema	Funções diversas utilizadas no sistema do lado do servidor.

global.php	_system	Define a conexão com o banco de dados, e ajusta os parâmetros globais da aplicação.
parse_page.php	_system	Implementa as classes ParsePage, BasePage, GridPage e ControlGridPage.

APÊNDICE B – DICIONÁRIO DO BANCO DE DADO

A seguir, é descrita estrutura das tabelas do banco de dados que foram modeladas para serem utilizadas no sistema de gerenciamento de formulários.

Tabela **cs_formulario**		
Descrição	Esta tabela armazena os formulários criados por um usuário	
Campo	**Tipo**	**Descrição**
id	inteiro	Chave primária
id_categoria	inteiro	Referencia para cs_categoria
nome	string	Nome do formulário

Tabela **cs_categoria**		
Descrição	Esta tabela armazena as categorias disponíveis de formulários	
Campo	**Tipo**	**Descrição**
id	inteiro	Chave primária
nome	string	Nome da categoria

Tabela **cs_paciente_formulario**			
Descrição	Esta tabela armazena a relação de pacientes associados aos formulários		
Campo	**Tipo**	**Descrição**	
id_paciente	inteiro	Referencia para cs_paciente	Chave primária
id_formulario	inteiro	Referência para cs_formulario	

Tabela **cs_campo_resposta**			
Descrição	Esta tabela armazena a resposta dada por um paciente para cada campo de um formulário.		
Campo	**Tipo**	**Descrição**	
id_campo	inteiro	Ref. Para cs_campo	Chave primária
id_paciente	inteiro	Ref. Para cs_pacientes	
id_formulario	inteiro	Ref. Para cs_formulario	

valor	string	Valor da resposta.

Tabela **cs_campos**		
Descrição	Esta tabela armazena os campos de um formulário.	
Campo	**Tipo**	**Descrição**
id	inteiro	Chave primária
id_formulario	inteiro	Ref. Para cs_formulario
ordem	inteiro	Ordem do campo
label	string	Nome do campo
ajuda	string	Texto que explica o preenchimento do campo
tipo	inteiro	1 = texto, 2 = radio, 3 = caixa de seleção
formatação	inteiro	1 = livre, 2 = numero, 3 = data
requerido	string	“T” = sim, “F” = não
opções	string	Lista de respostas para o caso de radio e caixa de seleção
campo_referencia	inteiro	indica a ordem do campo a qual este campo se referencia
campo_valor	string	indica o valor do campo referencia a qual este campo deve ser exibido