Gerador de Formulários

Jonas Henrique M. Korndorfer[1], Marcelo Augusto R. Schmitt[2]

[1]Campus Porto Alegre – Instituto Federal do Rio Grande do Sul (IFRS)
CEP 90.030-041 – Porto Alegre – RS – Brasil

jonas.korndorfer@gmail.com, marcelo.schmitt@poa.ifrs.edu.br

Resumo. *Tendo em vista deficiências nos geradores de formulários disponíveis no mercado, este artigo propõe um novo modelo para este tipo de sistema. O modelo busca solucionar questões relativas à tipagem de dados e ao controle de versionamento de formulários. São abordados aspectos sobre as potencialidades e o funcionamento dos módulos "Construção do Formulário", "Inserção de Dados" e "Geração de relatórios". A composição e forma de apresentação dos formulários ou novas versões destes podem ser definidas dinamicamente através de uma interface gráfica. Novos tipos de campos podem ser definidos e utilizados em diferentes formulários. Além disso, é proposta uma estrutura para inserção de dados nos formulários, visando à facilitação de recuperação destes, para pesquisas e relatórios.*

1. Introdução

Ao se acessar a Internet, percebe-se o uso frequente de formulários, utilizados para cadastros, enquetes, pesquisas e outras coletas de dados. Um exemplo percebido pela maior parte dos usuários da rede é o próprio cadastro ou entrada em uma rede social. Embora tais formulários possam parecer simples, o tempo demandado com tarefas rotineiras como a codificação de cadastros básicos e códigos de infraestrutura em geral violam duas das principais constantes da engenharia de software: "programmers time is valuable" e "programmers don't like repetitive, boring tasks" (Herrington, 2003, p.17). A simplicidade da tarefa não elimina os custos gerados e os problemas no desenvolvimento do software.

Nesse contexto, surgiram os geradores de código e, mais especificamente, os geradores de formulários, que visam mitigar estes problemas, fornecendo uma interface amigável (Nielsen e Loranger, 2000) para a construção do formulário, de forma fácil, prática e rápida. Existem vários geradores de formulários no mercado. Dentre os mais conhecidos, podem ser citados JotForm[1], Wufoo[2], Pandaform[3], Formstack [4] entre outros. Embora as interfaces com o usuário sejam distintas, todos estes *softwares* apresentam as seguintes funcionalidades:

- Geração do *layout* do formulário;
- Geração do código HTML (*HyperText Markup Language*);
- Geração do código CSS (*Cascading Style Sheets*);
- Geração do código JavaScript;

[1] http://www.jotformz.com/
[2] http://www.wufoo.com/
[3] http://pandaform.com/
[4] https://www.formstack.com/

- Banco de dados de *back-end* para armazenamento dos dados inseridos nos formulários criados;
- *Scripts* SQL (*Structured Query Language*) pré-definidos, para o acesso aos dados;
- Diversos modos de exibição dos dados.

Esses geradores resolvem a maioria dos problemas na criação de entrada de dados, padronizando o *layout*, e diminuindo o tempo de trabalho e os custos. Entretanto, poderiam explorar algumas outras funcionalidades, como a geração de tipos de campo e o controle de versões para os formulários.

Os tipos de campos podem ser entendidos como objetos, pois podem representar uma unidade de conhecimento (Campos, 2004), possuindo uma semântica implícita ao seu próprio nome além de regras de negócio. Esses campos poderiam ser definidos em tempo de execução pelo desenvolvedor e salvos em uma base de dados para sua reutilização. Ao alimentar um formulário, cada campo digitado transforma-se em um dado armazenado no banco de dados e que possui significado previamente definido. Por exemplo, poderíamos utilizar essa técnica para implementar um sistema de laudos de exames laboratoriais, no qual os campos podem ser reutilizados em diversos formulários ou, neste caso, exames. Uma vez armazenados, os dados gerados, possuem grande valor para elaboração de relatórios gerenciais, estatísticas, como fontes para pesquisas médicas e em saúde pública, ou mesmo como fonte de dados para Data mining (Maletzke et al., 2007).

O controle de versões para formulários permitiria mudanças no seu conteúdo sem perder a possibilidade de recomposição do seu formato inicial ou formatos anteriores. Isso pode ser implementado a partir da criação de máscaras de formulários com datas de validade. Dessa forma, torna-se possível criar o controle de versões para os formulários. Esta questão é importante porque, em alguns casos, um formulário pode ser alterado, mas as suas outras versões precisam ser mantidas. Um exemplo disto é a impressão de formulários antigos que, por questões legais, precisam ser visualizados exatamente iguais à sua primeira versão.

O presente artigo apresenta um gerador de formulário que inclui as funcionalidades descritas anteriormente: tipagem de campos e versionamento de formulários. O texto apresenta os fundamentos teóricos que justificam o modelo definido, as funcionalidades do sistema e a descrição do software desenvolvido. Este último funciona dividido em três etapas ou módulos. O primeiro módulo permite a criação do formulário, da máscara e dos tipos de campos. O segundo, a inserção dos dados nos formulários criados. E, por último, o terceiro módulo, a geração de relatórios com os dados coletados.

O sistema respeita o padrão de projeto "*Model View Controller"* (MVC) (Krasner e Pope, 1988). Este padrão é muito utilizado em aplicações WEB e visa a separação entre a interface (*View*) com o usuário, as ações (*Controller*) do sistema e seus dados (*Model*). Além disso, o desenvolvimento foi realizado na linguagem de programação PHP, auxiliada pelo *framework* YII[5]. O *framework* foi escolhido, pois já implementa o padrão MVC e funciona com orientação a objetos. Estas características facilitaram o desenvolvimento do sistema, sua manutenção e documentação.

[5] Yii, http://www.yiiframework.com/about

2. Desenvolvimento do gerador de formulários

O sistema é dividido em três módulos:

- Módulo de construção do formulário;
- Módulo de inserção dos dados;
- Módulo de geração de relatórios.

Os três módulos funcionam de forma independente. Entretanto, para que se possa fazer inserção de dados, o formulário e respectivo *layout* precisam estar previamente criados. Da mesma forma, para geração de relatórios é necessária a inserção prévia dos dados em instâncias dos formulários.

2.1. Módulo um, Construção do Formulário

O primeiro módulo é responsável por iniciar o processo de utilização do sistema, em geral, provendo estruturas para a construção dos formulários. A figura 1 detalha as atividades previstas para o primeiro módulo do sistema, separando as possíveis ações do usuário gerente.

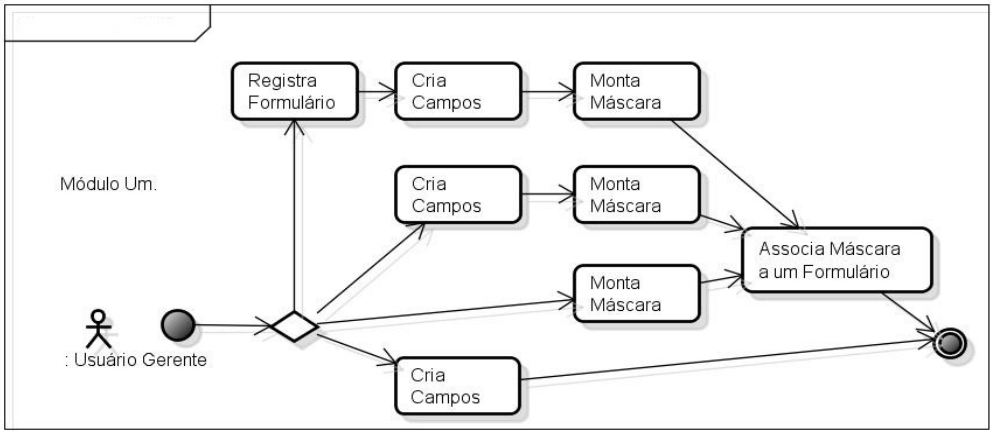

Figura 1. Diagrama de Atividades do Módulo Um

As ações apresentadas na figura 1 fazem referência à criação de um formulário como um todo, provendo diversos caminhos para isto. Como demonstrado nesta figura (figura 1), as atividades são independentes permitindo que o usuário possa escolher a sequencia que preferir, para criar seu formulário. Isto demonstra a versatilidade da estrutura para a construção de um formulário.

No modelo proposto (figura 2), um formulário é composto por três itens básicos:

- Registro do formulário no banco de dados, contendo seu nome e descrição;
- Construção e associação, de uma máscara, ao registro do formulário;
- Construção e associação, de um *layout*, para a máscara.

Os tipos de campo ficam vinculados aos elementos do *layout*, entretanto não possuem nenhum vinculo direto com o formulário, visando garantir que possam ser reutilizados. Isto pode ser visto no modelo de entidades e relacionamentos apresentado na figura 2.

Para a completa construção de um formulário utilizável, é necessária a criação de sua máscara. Esta máscara é montada compondo o *layout* do formulário, que por sua

vez, contem a composição dos campos desta máscara. Além disso, a máscara armazena as informações de sua data de validade e nome, a fim de possibilitar o versionamento dos formulários. Portanto, o módulo um do sistema também é dividido em três etapas: a criação dos tipos de campo, o cadastramento do formulário e a construção da máscara.

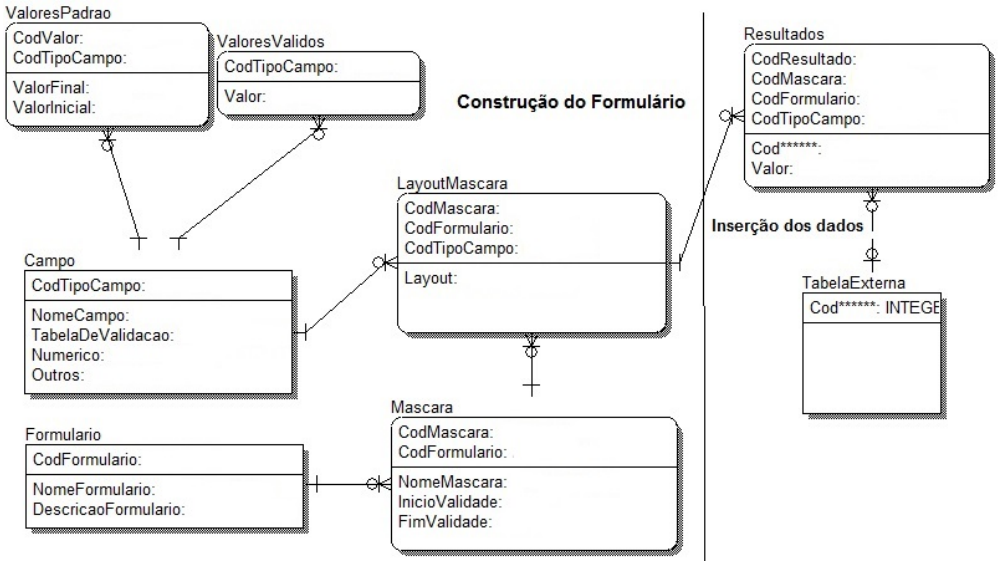

Figura 2. Modelo Entidade-Relacionamento do Sistema

Esta estrutura, além de facilitar a manutenção do sistema, permite que o usuário defina todos os campos necessários antes de criar um formulário, ou máscara, e ainda que os campos sejam reutilizados, tendo em vista que são unidades independentes dentro do sistema, podendo fazer parte de qualquer máscara (figura 2). Um exemplo prático é o da criação de um modelo de exame laboratorial, no qual o laboratorista já possui as definições dos campos, como seu nome, descrição e suas regras de negócio. Com isso, pode então criar os campos antecipadamente, para serem posteriormente utilizados na elaboração de diversas máscaras de diversos formulários. Podem ser criados, por exemplo, tipos de campo para posteriormente comporem um hemograma, como hemácias, hemoglobina e hematócrito.

2.1.1. Criação dos Tipos de Campo

Um tipo de campo é criado a partir de uma interface na qual é possível cadastrar suas características e regras de negócio (figura 3). Este será salvo no banco de dados como um novo tipo de campo (figura 2, entidade: Campo), permitindo, dessa maneira, a sua utilização na construção de uma máscara. Manter os tipos armazenados é importante, tendo em vista que podem ser utilizados a qualquer momento em qualquer formulário. Além disso, como os tipos campo devem possuir uma semântica associada, precisam estar armazenados em algum local, para que tenham um código no contexto do sistema, e assim possam ser recuperados.

Figura 3. Interface para Cadastro de Tipos de Campos

No sistema desenvolvido, o cadastro dos tipos de campo permite a definição de características como seu nome e descrição (figura 3). Além disso, provê a atribuição de regras de negócio ao tipo criado. Atualmente o protótipo permite que as regras definam se o tipo de campo é somente numérico, obrigatório, ou com tamanho máximo. Futuramente outras opções de regras de negócio como a definição de valores válidos para determinado campo e valores padrão, serão adicionadas ao sistema.

2.1.2. Cadastramento dos Formulários

O cadastro de um formulário refere-se ao seu registro no sistema. Para isto é disponibilizada uma interface simples (figura 4).

Figura 4. Interface para Cadastro de Formulários

A interface criada (figura 4) permite o cadastro de um nome e uma breve descrição do formulário. Estes dados são armazenados na entidade formulário do banco de dados (figura 2, entidade Formulario). Este registro de um formulário permitirá que futuramente máscaras componham seu *layout*. Para isto existe o relacionamento entre um formulário e suas respectivas máscaras, explicado na sessão 2.1.3 deste artigo.

2.1.3. Criação das Máscaras

As máscaras definem o *layout* do formulário, e são montadas a partir de campos básicos do HTML. Por sua vez, esses campos são disponibilizados na tela de forma que possam

ser arrastados por um usuário e posicionados em um local que permite a definição de um *layout* para o formulário (figura 5). Outro ponto importante desta interface é que nela podem ser criadas versões dos formulários. Ao definir-se um *layout* para um formulário, será determinada uma data inicial de validade deste. Caso outro *layout* seja criado para este mesmo formulário, o *layout* anterior assume uma data de validade final e passa a servir apenas pera consulta de dados antigos, por outro lado o novo *layout* torna-se vigente para inserção de novos dados.

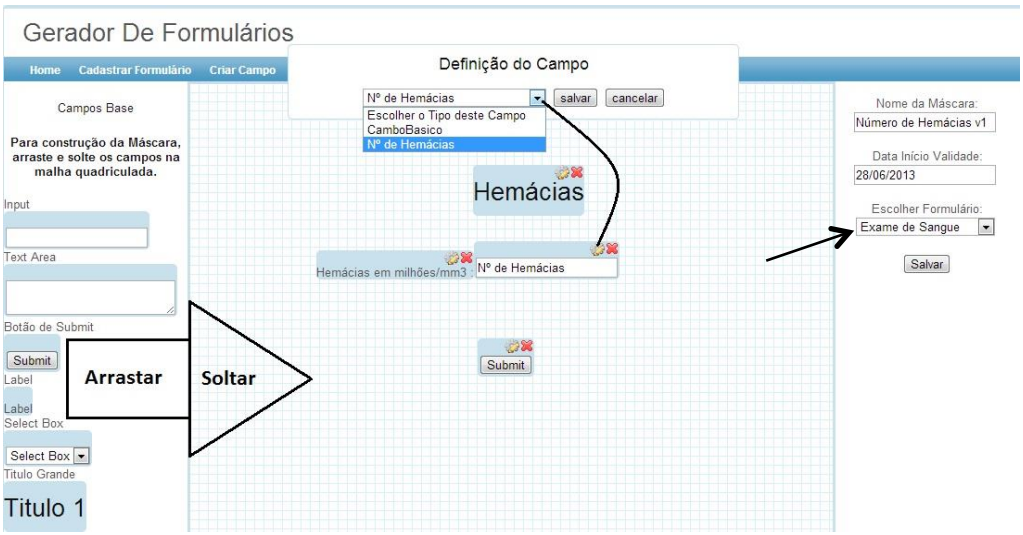

Figura 5. Interface para Construção das Máscaras

Na tela de construção de uma máscara (figura 5), três itens chamam a atenção:

- Os campos básicos posicionados a esquerda;
- A malha quadriculada central;
- Um formulário à direita.

Os campos do lado esquerdo da interface representam campos básicos do HTML e são utilizados para a confecção de um *layout*. Estes campos podem ser arrastados pelo usuário para a malha quadriculada central, na qual são posicionados para construção da máscara.

Ao posicionar um campo na malha, dois botões são exibidos ao usuário, para edição e exclusão dos campos, representados respectivamente por uma engrenagem e um x. Cada campo base recebe um tratamento distinto quando seu botão de edição é acionado. Campos que não representam uma possível entrada de dados, como títulos e *labels*, permitem que o usuário edite apenas o seu conteúdo textual. Por outro lado, os elementos que podem representar inserção de dados, como *inputs* e *TextArea*, fornecem uma edição diferente. Estes proveem, por meio de uma janela central, a possibilidade de associar-se um campo básico a um tipo de campo definido previamente. Com isso, este campo básico passa a ter um significado e conter um nome, descrição e regras de negócio associadas.

O formulário à direita define o nome da máscara a ser criada, a data na qual ela entra em vigência e a qual formulário ela está relacionada. Definir o nome e a data de vigência de uma máscara é uma ação obrigatória dentro do sistema. Isto ocorre, pois um formulário pode conter varias máscaras com *layout*s antigos, entretanto apenas uma

válida para inserção de dados. Outro ponto é a definição do formulário ao qual esta máscara está associada, o que posteriormente permitirá sua recuperação.

Acionando o botão salvar do formulário à direita (figura 5), o sistema irá persistir os dados de nome da máscara, data inicial de validade e formulário associado, na tabela máscara do banco de dados (figura 2, entidade Mascara). O *layout* construído é armazenado na tabela LayoutMascara do banco (figura 2, entidade LayoutMascara). Esta, por sua vez, armazena o posicionamento dos campos na máscara e seus respectivos tipos.

2.2. Módulo dois, Inserção de Dados

Este módulo do sistema permite o carregamento do formulário a partir de sua máscara, o preenchimento dos campos por parte do usuário e a persistência dos dados. Além disso, antes do armazenamento, os campos serão validados pelo sistema de acordo com suas regras de negócio previamente definidas, visando manter a integridade dos dados.

A figura 6 apresenta o diagrama de atividades referente ao processo de inserção dos dados em um formulário.

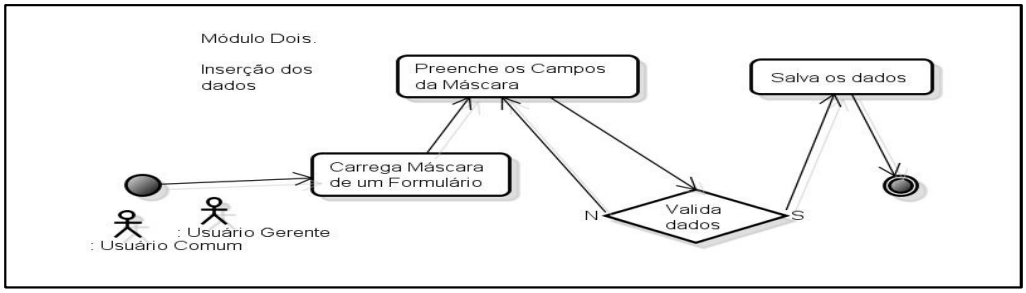

Figura 6. Diagrama de Atividades do Módulo Dois

Este diagrama apresenta as etapas que um usuário percorre para realizar uma inserção de dados (figura 6). Começando com a seleção de um formulário, passando ao seu preenchimento, e terminando com a persistência dos dados.

Para a inserção de dados foi elaborada uma interface que permite ao usuário, selecionar o formulário, carregar sua máscara, preencher os campos e salvar os dados (figura 7). Além disso, foi adicionada a possibilidade de caso o sistema esteja sendo implantado em um estabelecimento com um banco de dados já definido, associarem-se os registros de um formulário a uma entidade deste. Um exemplo disso é se o sistema for instalado em um laboratório de análises clínicas, e for construído um formulário para hemograma, será possível associar-se os valores deste, a um paciente do laboratório.

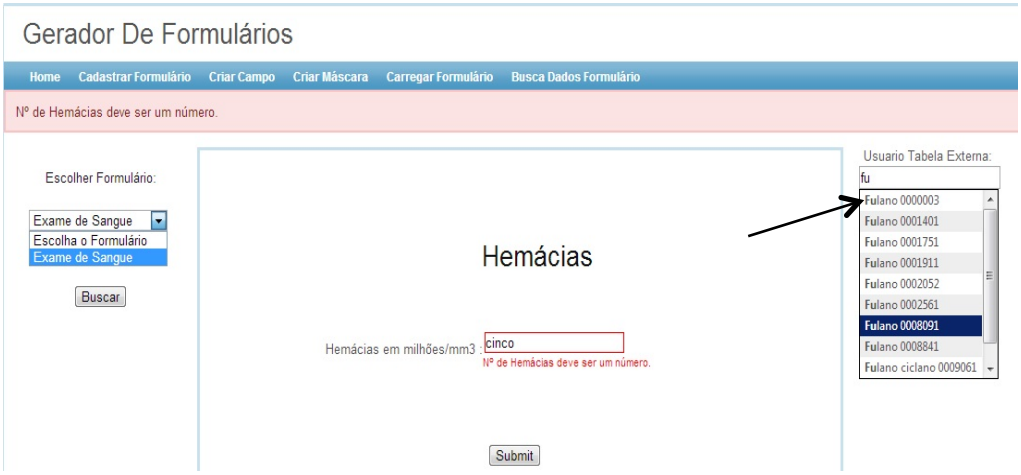

Figura 7. Interface para Carregar e Preencher os Formulários

A interface atual permite que o usuário carregue um formulário selecionando-o no *selectbox* a esquerda da tela (figura 7) e acionando o botão buscar. Com o formulário carregado na área central da tela, dados podem ser inseridos nos campos e armazenados utilizando o botão *Submit*. Além disso, antes de armazenar os dados, o sistema valida-os de acordo com as regras de negócio definidas para cada campo e, caso exista alguma regra violada, são exibidas mensagens de erro no topo da tela e no respectivo campo.

Outro ponto importante é o campo usuário tabela externa a direita, na figura 7. Este campo serve para associarem-se os registros do formulário a uma entidade existente no local de instalação do sistema. Para esta associação foi criada uma tabela no modelo, com o nome de TabelaExterna (figura 2, entidade TabelaExterna). Na fase atual do sistema, esta tabela (figura 2, entidade TabelaExterna), deve ser preenchida em uma etapa de implantação da aplicação. Esta etapa consiste em carregar os dados da entidade existente no local de instalação, para banco de dados (figura 2, entidade TabelaExterna) do sistema, com isso permitindo o cruzamento dos dados.

Os registros dos dados de todos os formulários serão feitos em uma tabela única, que armazena o valor do campo e relaciona os códigos de formulário, campo, máscara e possivelmente da tabela externa (figura 2, entidade Resultados). Isto significa que cada campo representa um dado armazenado no formato de um registro na tabela.

Manter os registros em uma única tabela facilitará a pesquisa, tendo em vista que todas as informações necessárias para uma filtragem na recuperação dos dados, podem ser agrupadas por tipos de campos que por sua vez possuem uma semântica definida pelo usuário. Além disso, como o sistema permite a utilização de um mesmo campo em diversos formulários, a recuperação de todos os resultados de um único campo ou o numero de ocorrências deste, desconsiderando a qual formulário ou máscara ele está relacionado, é interessante. Um caso em que pode ser útil a recuperação dos dados desta forma é, por exemplo, em laboratórios de análises em geral. Nestes laboratórios, cada campo pode ser um resultado de uma análise e cada análise tem um custo. Portanto, para fins gerenciais, fazer um levantamento de quantas análises de um determinado tipo foram feitas, tem grande importância.

2.3. Módulo três, Geração de relatórios

A geração de relatórios permitirá a recuperação dos dados armazenados de varias formas diferentes. Essas formas permitem o relacionamento dos códigos de máscara, formulário, e campo (figura 8), para a recuperação dos dados. Além disso, o relatório poderá ser emitido levando em consideração o relacionamento entre o resultado e um registro da tabela externa.

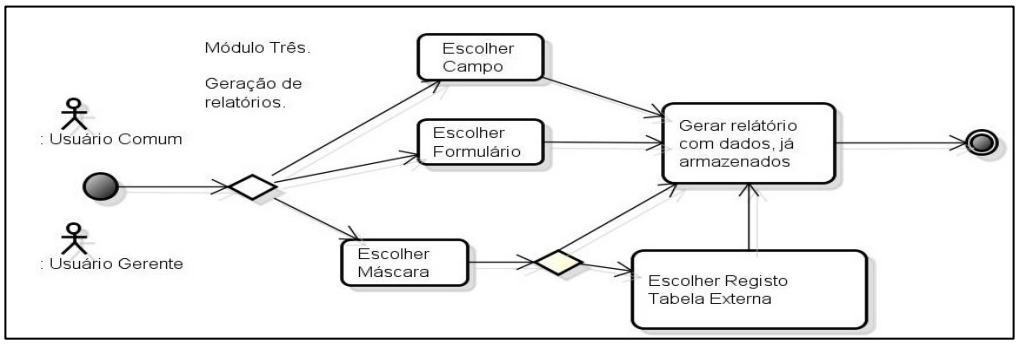

Figura 8. Diagrama de Atividades do Módulo Três

A figura 8 demonstra as atividades executadas pelos usuários, dos tipos gerente e comum, para a geração de um relatório. Além disso, mostra que uma das ações necessárias é a escolha dos parâmetros de entrada para a definição do tipo de relatório a ser gerado.

O protótipo desenvolvido possibilita a seleção de uma máscara e um registro da tabela externa. Com isso, o sistema recupera o *layout* do formulário e os valores relacionados a esse registro (figura 9).

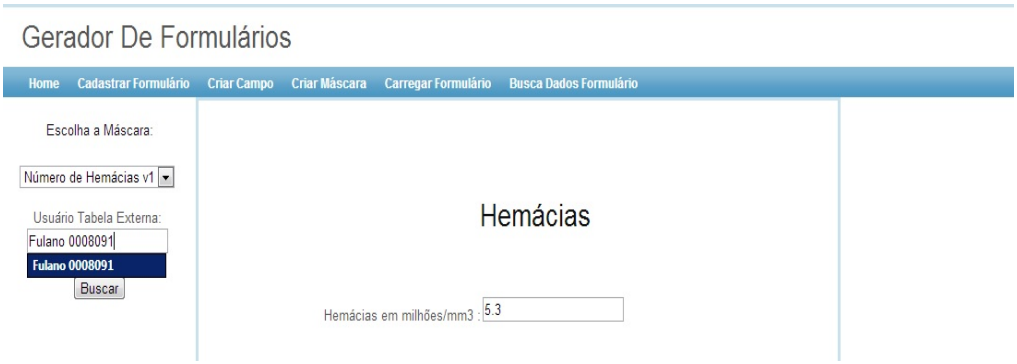

Figura 9. Interface para Buscar Dados dos Formulários

É possível a emissão de três tipos de relatórios:

- Relatórios por Máscara;
- Relatórios por Formulário;
- Relatórios por Campo.

Os relatórios baseados em uma máscara foram implementados com a interface da figura 9. Esta permite que uma máscara seja escolhida no *selectbox*, e um registro da tabela externa seja selecionado, com o campo usuário tabela externa, a esquerda da tela. Com isso, os dados referentes ao código desta máscara, relacionados, ao código da tabela externa, são exibidos de acordo com o *layout* da máscara. Este relatório pode ser usado, por exemplo, para a emissão de um laudo de exame.

O relatório baseado em formulários será gerado através dos resultados de todas as máscaras atreladas ao seu código. Esta opção serve para fins administrativos, visando fornecer informações como, quantas versões o formulário possui, qual a versão válida no momento e quantos registros já existem para este formulário.

Com o relatório baseado em um determinado tipo de campo, será possível recuperar todos os dados armazenados referentes a seu código. Esta forma serve para fins administrativos ou de pesquisas. Recuperando os dados desta maneira, os códigos de máscara e formulário, são desconsiderados. Então, a partir destes resultados, é possível, por exemplo, a descoberta de quantos campos de determinado tipo já foram armazenados. Outra utilidade pode levar a característica semântica do campo em consideração, por exemplo, a descoberta do número de resultados positivos para leptospirose obtidos por determinado laboratório. Além disso, como o sistema permite a associação destes resultados a usuários já existentes no laboratório, realizar um cruzamento de dados entre os resultados dos exames e os pacientes fica extremamente simples. Isso devido à possibilidade de acessar-se diretamente o tipo de campo desejado, recuperando-se seu valor e paciente associado. Por exemplo, em pesquisas na área de saúde pública, seria possível agrupar os resultados pelas regiões dos pacientes, desta forma, provendo a descoberta de locais de alta incidência da doença.

3. Casos de Uso

Tendo em vista que o sistema poderá ser utilizado por um número grande de usuários, para melhor organização, serão determinados níveis diferentes de permissões, com dois tipos de usuário: o gerente e o comum (figura 10). Desta forma, será possível a administração do sistema de forma mais organizada, permitindo uma divisão de tarefas e um controle de acesso a determinadas funcionalidades.

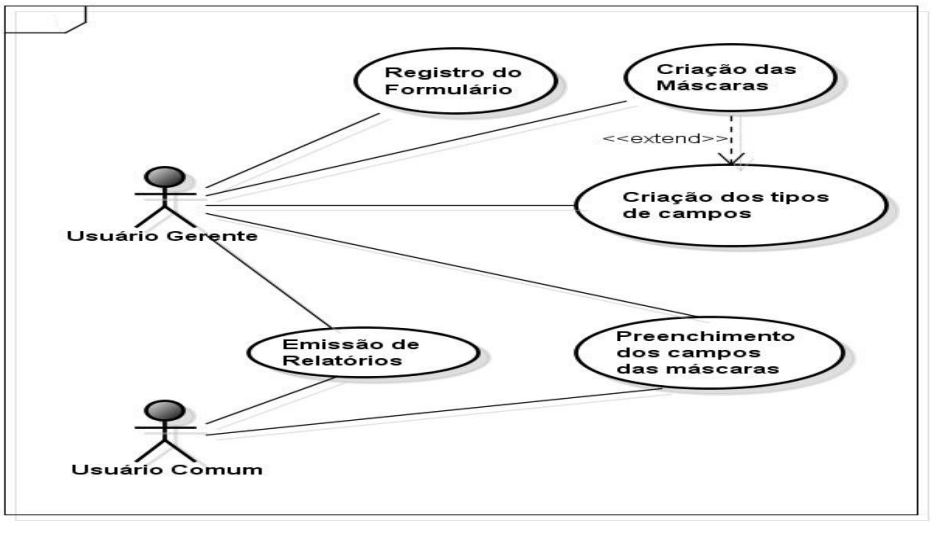

Figura 10. Diagrama de Casos de Uso do Sistema

Existirão dois tipos de usuário:

- Usuário Gerente;
- Usuário Comum.

O usuário gerente será responsável por administrar o sistema, tendo permissão total para acessar qualquer funcionalidade, (figura 10, Usuário Gerente). Dentre as suas

tarefas, a principal é a criação dos formulários como um todo, desde a definição dos campos até a construção e associação de uma máscara para o formulário.

O usuário comum ficará com permissões limitadas dentro do sistema, podendo apenas gerar relatórios e inserir dados em formulários já criados por um gerente, (figura 10, Usuário Comum). Com isso o sistema controlará o acesso à funcionalidade de criação do formulário, não permitindo que o usuário comum altere as estruturas de campos, máscaras e formulários, criadas por um gerente. Desta forma, este usuário fica com as tarefas mais práticas, que podem ser solicitadas por um gerente.

4. Protótipo desenvolvido e trabalhos futuros

O protótipo desenvolvido está descrito no capítulo 2 deste artigo, e é um primeiro passo para a implementação completa do modelo. No estágio atual é possível cadastrar formulários, criar tipos de campo, construir máscaras, inserir e recuperar os dados. Estes itens compõem todo o modelo, entretanto podem ser aprimorados para contemplarem mais funcionalidades e opções aos usuários. Pretende-se dar continuidade ao trabalho, tendo como principais objetivos o estudo de uma interface mais amigável, a implementação de mais regras de negócio associáveis aos tipos de campo e a programação de mais elementos e opções para a construção das máscaras.

Por se tratar de um protótipo e ter como principal objetivo provar a importância da criação dos tipos de campo e versões dos formulários, a implementação de níveis de usuário não foi realizada, tendo em vista que não influi nestas questões. Por outro lado, foi desenvolvido o relatório por máscaras, com o objetivo de demonstrar um meio de recuperação dos dados inseridos. Além disso, foi acrescentada a possibilidade de uma tabela externa ser inserida no contexto do sistema. Esta tabela ainda não está totalmente definida, fazendo parte de uma fase de instalação do sistema. Porém, pretende-se desenvolver um método para que o próprio usuário da aplicação possa importar sua tabela.

5. Conclusão

O desenvolvimento de aplicações baseadas em formulários web é tarefa corriqueira que demanda considerável tempo dos programadores e que pode ser facilitada por ferramentas automatizadas. Embora existentes, tais ferramentas podem receber aperfeiçoamentos importantes. Com as estruturas de armazenamento e o padrão de desenvolvimento proposto, foi possível a construção de um gerador de formulários que permite, além da criação de um simples formulário, a criação de tipos de campos e o controle de versões. Estas estruturas possibilitarão uma nova gama de uso dos geradores. Um exemplo desta nova abrangência é a aplicabilidade deste gerador para a construção de laudos de exames laboratoriais em geral. Isto ocorre devido à importância da criação de tipos de campos, com semântica associada, e o controle de versões, para estes laboratórios. Outro exemplo demonstra a importância da forma como os dados são armazenados. Este ocorre no caso de os laboratórios de um hospital, utilizarem o sistema. Hospitais muitas vezes possuem algum tipo de vínculo com universidades e unidades governamentais, por este motivo, manter os dados armazenados todos em uma única tabela, com fácil acesso, diversas formas de filtragem e separados por tipos, tem

grande importância, tendo em vista que estes estabelecimentos e até mesmo o próprio hospital, podem ter grande interesse em realizar pesquisas com os dados armazenados.

6. Referências

CAMPOS, M. L. A. Modelização de domínios de conhecimento: uma investigação de princípios fundamentais. **Ciência da Informação**, Brasília. v. 33, n. 1, p. 22-32, abr. 2004.

HERRINGTON, Jack. **Code generation in action**. Greenwich: Manning Publications, 2003, p. 17.

KRASNER, G. E.; POPE S. T. **A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System**, tech. report, ParcPlace Systems, Mountain View, Calif., 1988.

MALETZKE et al. Mapeamento de Informações Médicas descritas em Formulários para Bases de Dados Estruturadas. **In: VII Workshop de Informática Médica (WIM 2007), Porto de Galinhas**, 2007, p. 49–58.

NIELSEN, Jacob; LORANGER, Hoa. **Usabilidade na Web: Projetando Websites com Qualidade**. Rio de Janeiro: Campus, 2007.