UNIVERSIDADE FEEVALE

CRISTIAN CARDOSO

NÚMENOR - GERADOR DE CÓDIGO PARA APLICAÇÕES WEB

Novo Hamburgo, novembro de 2010.

CRISTIAN CARDOSO

NÚMENOR - GERADOR DE CÓDIGO PARA APLICAÇÕES WEB

Universidade Feevale
Instituto de Ciências Exatas e Tecnológicas
Curso de Sistemas de Informação
Trabalho de Conclusão de Curso

Professor Orientador: Juliano Varella de Carvalho

Novo Hamburgo, novembro de 2010.

AGRADECIMENTOS

Um agradecimento especial aos meus familiares que sempre estiveram ao meu lado, apoiando, incentivando, compreendendo e acreditando nos meus estudos.

Ao meu orientador Juliano Varella de Carvalho, pela atenção, dedicação, incentivo e fundamental apoio nas sugestões e revisões deste trabalho.

RESUMO

Com a popularização do uso da Internet e o grande crescimento da utilização de sistemas *on-line*, juntamente com os constantes avanços na área de Tecnologia da Informação, as tecnologias inovadoras viabilizaram o desenvolvimento de aplicações e ferramentas complexas sob o ambiente web. Uma destas tecnologias que tem ganhado grande ênfase são os geradores de códigos em linguagens de programação orientadas a objetos, voltadas à programação web, que tem como principal objetivo, realizar a construção de funcionalidades das aplicações de maneira mais eficiente, segura e de forma padronizada, garantindo a diminuição do tempo de desenvolvimento, como a redução dos custos do projeto. Neste trabalho será realizado um estudo e comparação entre sistemas geradores de códigos existentes para a linguagem de programação PHP, que é uma das principais linguagens existentes para Internet, esta linguagem é gratuita e muito conhecida por sua simplicidade, praticidade e portabilidade. Será proposto também o desenvolvimento de um novo gerador de código que reúna os conceitos mais atuais em programação para a Internet, tendo como base os *frameworks* Zend Framework e Jquery, existentes no mercado atual, a fim de agilizar o desenvolvimento de sistemas web, utilizando a linguagem de programação PHP.

Palavras-Chave: Gerador de código, CRUD, *Framework*, Zend Framework, PHP.

ABSTRACT

The internet popularization and growth of online systems coupled with advances on the Information Technology (IT), the innovative technologies made possible the development of complex web tools and applications. One technology who gained large focus are the oriented object code generator for web environment, which has as its main objective, construct features of the applications more efficiently, safely and standardized, ensuring also beyond the reduction of development time, the reduction of project costs. This work is a detailed study and a comparison of existing code generation systems for the PHP programming language, one of the most important existing computer languages for the Internet. this language is free and known for the simplicity, convenience and portability. This work will also propose developing a new code generator that meets the newest concepts in computer programming for the Internet, based on the frameworks Zend and Jquery, existing in the current market in order to improve the development of web systems, using PHP.

Key words: Database Code generator, CRUD, Framework, Zend Framework, PHP.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous Javascript and XML
CSS	Cacading Style Sheet
CMS	Content Management System
CRUD	Create, Retrieve, Update e Delete
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
JSON	Javascript Object Notation
MVC	Model-View-Controller
OOP	Object-Oriented Programming
PDO	PHP Data Objects
PHP	Hypertext Preprocessor
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
URL	Universal Resource Locator
XML	Extensible Markup Language

SUMÁRIO

INTRODUÇÃO

No início da *World Wide Web* (entre 1990 e 1995), os sites web eram formados por pouco mais do que um conjunto de arquivos de hipertextos ligados, que apresentavam informações usando texto e alguns gráficos. Com o passar do tempo, a popularização do uso da Internet, não somente para páginas web, mas também para a utilização de sistemas on-line, fez com que as indústrias de software tenham como uma de suas principais metas a construção de sistemas e sites de alta qualidade e em um curto espaço de tempo. Na grande maioria das vezes estas metas não são alcançadas devido ao aumento da complexidade dos sistemas, fazendo com que muitos projetos ultrapassem seus prazos, custos ou que o software apresente baixa qualidade.

Na construção de sistemas, normalmente se observa a reutilização de códigos desenvolvidos anteriormente. Na maioria das vezes esta reutilização é feita através de "copiar/colar/modificar". A qualidade do código produzido pode variar conforme a quantidade destas operações de reuso. Quanto maior a quantidade de adaptações, pior a qualidade, devido a estas operações frequentemente levarem a erros, conforme (PRESSMAN, 1995) toda vez que uma mudança é introduzida num procedimento lógico complexo, o potencial de erros cresce.

Uma forma de reduzir este tipo de reutilização seria através da utilização de programas de geração de código. Um gerador de código é uma ferramenta de auxílio ao processo de desenvolvimento de sistemas que atua na fase de implementação do projeto, gerando o código-fonte que seria criado pelo programador. De forma geral, esses geradores criam o código-fonte com base nas informações existentes no modelo de dados. Isso significa ganho de produtividade, redução de tempo e qualidade no código gerado.

A qualidade do código varia durante o ciclo de vida de um projeto, podendo começar no alto e decair ou vice-versa. De acordo com (PAULA FILHO 2003) uma fração

significativa das modificações de manutenção introduz, por sua vez, novos defeitos. O código gerado aumenta sua qualidade com o tempo, pois os erros encontrados podem ser uniformemente reparados na base do código.

Outra maneira de agilizar o processo de desenvolvimento, a fim de evitar erros, seria com a utilização de *framework*. Conforme (LISBOA 2009), utilizar um *framework* de desenvolvimento significa reaproveitar o trabalho e o conhecimento de outra(s) pessoa(s) e seguir suas orientações. A palavra "reaproveitar" nos remete à orientação a objetos. O reaproveitamento de código não é privilégio nem invenção da orientação a objetos, mas esse paradigma possui técnicas eficazes para implementá-lo.

A qualidade distintiva de um *framework* é que ele oferece uma implementação para as funções básicas e invariantes e inclui um mecanismo para permitir que o desenvolvedor se conecte às diversas funções ou as estenda (LARMAN, 2007).

Conforme (SILVA e VIDEIRA, 2001), as atividades de desenvolvimento devem ser apoiadas por ferramentas, que a partir da especificação da modelagem, torna-se possível produzir, de forma automática, diversos componentes do sistema, permitindo a execução das tarefas com menores esforços e com maior agilidade. Desta forma, através da utilização de geradores de aplicações, integrados à *frameworks*, é possível desenvolver software com maior agilidade e consistência, seguindo boas práticas de programação.

Dentro do diversificado leque de categorias de ferramentas que prestam apoio às atividades da Engenharia de Software (CASE), uma específica vem ganhando cada vez mais destaque e, sobre ela, tem-se aplicado muitos investimentos nos últimos tempos: as Ferramentas de Geração de Código, ou simplesmente Geradores de Código.

Sendo assim, o objetivo deste trabalho será o desenvolvimento de uma ferramenta de auxílio a programadores capaz de gerar código para o *framework* Zend de forma simples e rápida, tendo por principal finalidade agilizar o processo de criação de aplicativos, aumentando a qualidade do código, reduzindo assim custos das empresas.

Este trabalho está dividido em três capítulos.

O primeiro capítulo irá apresentar uma introdução sobre o assunto e o objetivo do trabalho, a fim de dar ao leitor as informações necessárias ao entendimento do assunto abordado.

No segundo capítulo será abordada a utilização de *frameworks* no desenvolvimento de software, descrevendo as suas características, benefícios, aspectos de qualidade, assim como, o *framework* Zend, utilizado no desenvolvimento da ferramenta.

No ultimo capítulo será apresenta a especificação e implementação da ferramenta proposta.

1 GERADORES DE CÓDIGO

A técnica de geração de código refere-se a utilizar um programa gerador de código, para gerar outro programa, ou parte dele. Um gerador de código é como qualquer outro programa que recebe alguma entrada de dados e cria alguma coisa nova como saída (ODISI, 2008).

Os geradores de código podem transformar atividades rotineiras em atividades automáticas, nas quais os geradores de código recebem como entrada uma especificação de alto nível e convertem tal especificação em programas, ou parte de programas. Esta especificação pode ser definida através de um diagrama gráfico, através de um formulário interativo onde o usuário seleciona as opções desejadas, ou até mesmo serem escritas em alguma linguagem, como se fossem uma linguagem de programação. Após a definição das especificações, o gerador de código atua de maneira semelhante a um compilador, transformando as informações de alto nível, as especificações realizadas, em uma implementação de baixo nível, neste caso o código-fonte (ODISI, 2008).

Os geradores de código também são úteis para fazer protótipos de códigos de produção. Usando um gerador de código, você pode montar em poucas horas um protótipo que demonstre os principais aspectos de uma interface com o usuário ou pode experimentar várias estratégias de projeto. Talvez demore várias semanas para codificar a mesma funcionalidade manualmente (MCCONNELL, 2005).

> "Um artefato é qualquer item criado como parte da definição, manutenção ou utilização de um processo de software." (FRANCA, 2000)

Um gerador de artefatos é uma ferramenta de software que produz um artefato a partir da sua especificação de alto nível conforme figura 1.1. O artefato gerado pode ser composto por outros artefatos.

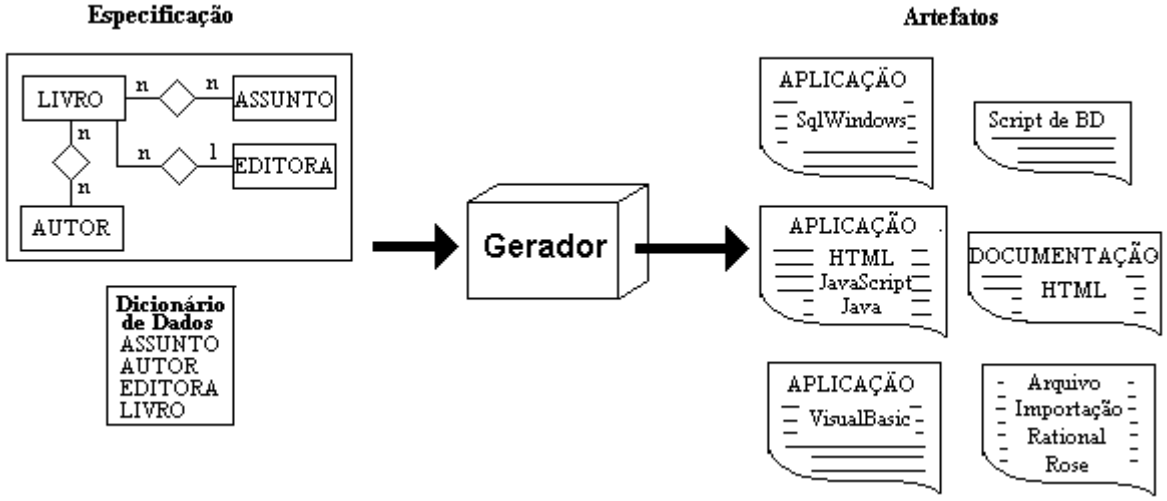

Figura 1.1: Visão esquemática de um gerador de artefatos.
Fonte: (FRANCA, 2000)

Desde o início, geradores comerciais são caracterizados por produzir aplicativos específicos ou por automatizar tarefas dentro do processo de desenvolvimento, como por exemplo, geração de: relatórios, telas, consultas, estruturas de banco de dados, programas de conversão de arquivos, etc. A sua utilização pode proporcionar elevados ganhos de produtividade. A arquitetura de um gerador de artefatos é formada por componentes que realizam as transformações da especificação até o produto final (FRANCA, 2000).

A figura 1.2 apresenta uma visão esquemática da utilização de geradores.

A construção de um artefato através de um gerador envolve as seguintes etapas:

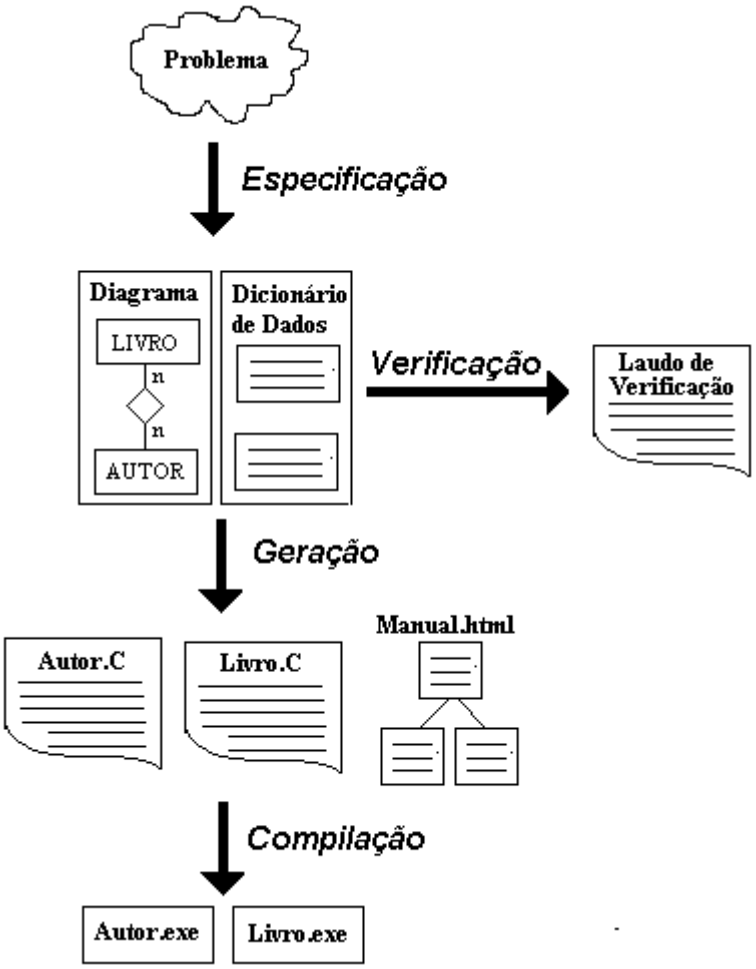

Figura 1.2: Visão esquemática da utilização do gerador de artefatos
Fonte: (FRANCA, 2000)

- **Especificação**: A especificação deve prover todas as informações necessárias para a geração do artefato. Além disso, antes de disparar o processo de geração, o programa deve verificar se a especificação está completa através do processo de verificação da especificação.
- **Geração**: Após fornecer a especificação e, caso esta não contenha problemas, o processo de geração é executado. A geração utiliza informações da especificação e da descrição do artefato que está sendo gerado. O artefato gerado pode ser composto por um ou mais arquivos, podendo estar codificado em diferentes linguagens.
- **Compilação do Artefato**: Caso o artefato gerado seja um programa, é necessário compilar os fontes gerados após a sua geração para obter uma versão executável do artefato.

Benefícios da geração de código para o programador segundo (HERRINGTON, 2003) são:

- **Qualidade**: evita grande quantidade de código escrito a mão, onde poderia ter códigos inconsistentes ou sem necessidade;
- **Consistência**: o código gerado pelo gerador é consistente, sem surpresas nos resultados, interface fácil de usar e entender;
- **Único ponto de conhecimento**: uma mudança do nome da tabela envolveria mudanças em manuais, no programa, na documentação, e na camada de teste. Com um gerador de código permite que você mude um nome da tabela em uma única posição e regenere então o programa, a documentação, e a camada de teste;
- **Projetos mais detalhados**: a geração de código comprime o tempo de programação em determinados sistemas, desta forma sobra mais tempo para ser gasto fazendo o projeto evitando o retrabalho.

Muitas das vantagens para o programador podem ser importantes para o administrador, para o crescimento da produtividade e da qualidade. Há certos aspectos, portanto que são importantes unicamente em nível de negócios. Segundo (HERRINGTON, 2003) as vantagens são:

- **Consistência arquitetural**: código gerado é bem documentado, mesmo que os membros saiam do projeto, o programa será de fácil entendimento;
- **Maior produtividade**: a geração de código reduz a quantidade de horas gastas no desenvolvimento do projeto e mantém os programadores focados em um único trabalho ao invés de grandes volumes de código.
- **Desenvolvimento ágil**: o software será fácil de modificar, caso ocorra alguma alteração basta gerar o código do projeto novamente.

Após o software ser gerado, testado e implantado, o sistema pode precisar de algum tipo de alteração. A alteração dos artefatos gerados por uma ferramenta pode representar um problema maior do que a alteração de aplicações desenvolvidas manualmente.

De acordo com (ODISI, 2008), a geração de código é ideal para problemas bem conhecidos, como por exemplo, camadas de acesso a banco de dados, ou qualquer outro problemas cuja arquitetura seja bem definida. Quando a estabilidade do código a ser gerado ou da arquitetura utilizada é duvidosa, não é aconselhável a utilização de geradores de código. Para que se torne possível a utilização de geradores em tais situações faz-se necessário que alguns protótipos sejam codificados manualmente antes da utilização do gerador de código, tendo como objetivo, adquirir conhecimento sobre a arquitetura e código a serem gerados

1.1 RAD (*Rapid Application Development*)

O RAD é um modelo de processo de software incremental que enfatiza um ciclo de desenvolvimento curto. O modelo RAD é uma adaptação “de alta velocidade” do modelo em cascata, no qual o desenvolvimento rápido é conseguido com o uso de uma abordagem de construção baseada em componentes. Se os requisitos forem bem compreendidos e o objetivo do projeto for restrito, o processo RAD permite a uma equipe de desenvolvimento criar um “sistema plenamente funcional“, dentro de um período de tempo muito curto como por exemplo, de 60 a 90 dias (PRESSMAN, 2010).

Os objetivos da utilização do RAD incluem desenvolvimento mais rápido de software e sistemas com alto nível de manutenção. Em acréscimo, dado que cada vez mais as ferramentas RAD incorporam o paradigma de orientação a objetos, os benefícios proporcionados pela reusabilidade ficam cada vez mais claros.

Grande parte das metodologias RAD requer o uso de novas ferramentas e tecnologias que têm uma curva de aprendizado significativa. Frequentemente, essas ferramentas e técnicas aumentam a complexidade do projeto e exigem um tempo adicional de aprendizado. Entretanto, depois que elas forem adotadas e a equipe se tornar experiente, elas podem aumentar significativamente a velocidade do ciclo de vida do desenvolvimento do sistema (DENNIS e WIXOM, 2003)..

1.2 Geradores de Código Existentes

A grande maioria dos geradores de códigos existentes trabalham de forma similar, quanto a construção das aplicações. Necessitam que um banco de dados já construído e de uma conexão com esse banco de dados e a partir das informações existentes, geram o código para a aplicação. Podendo classificar os geradores de código em dois diferentes modelos: ativo e passivo.

- Ativo: o código se mantém enquanto o gerador estiver ativo, ou seja, para que o aplicativo funcione, é necessária a presença do gerador.
- Passivo: o modelo passivo, o gerador cria o código inicial e sua manutenção e funcionalidade passa a depender completamente do desenvolvedor.

1.2.1 ScriptCase

É um ambiente completo de desenvolvimento de aplicações Web em PHP com uso da tecnologia AJAX. O desenvolvimento é feito diretamente no browser, permitindo integração da equipe de desenvolvimento, além de possibilitar o desenvolvimento colaborativo.

A ferramenta cria formulários para manipulação de dados em tabelas SQL para os bancos de dados mais usados no mercado como MySQL, PostgreSQL, Oracle, MS SQLServer, DB2, Sybase, MS Access, etc. As aplicações geradas são totalmente independente da ferramenta. O interessante é que os formulários criados já veem com funcionalidades que custariam boas horas de codificação. Por exemplo, utilizam diversos recursos AJAX como: navegação, validação de campos automática o que é bastante útil para campos do tipo data, moeda, número, telefone, etc.

O ScriptCase permite criar novos sistemas ou agregar aplicações WEB a sistemas já existentes. Para o usuário profissional, o ScriptCase permite a programação avançada (regras de negócio), permitindo até a criação de sistemas complexos. Para o usuário final, devido a facilidade de manuseio, o ScriptCase requer apenas conhecimento básico de SQL, torna-se forte gerador de relatórios e consultas em diversas saídas, destacando-se os formatos PDF, XLS e RTF.

1.2.1.1 Vantagens

- Suporte aos bancos de dados: MySQL, PostgreSQL, Oracle, MS SQLServer, DB2, Sybase, MS Access;
- Multi-plataforma;

1.2.1.2 Desvantagens

- Custo de R$800,00 para o pacote *Professional Edition*, sendo para desenvolvedores únicos e estudantes;
- Geração de código no anti-padrão mais conhecido e difundido entre desenvolvedores, o famoso código espaguete.

1.2.2 PHPRunner

Esse aplicativo agiliza a programação de sistemas para *Web* na linguagem PHP. Sua função é criar códigos e algoritmos para páginas da *Web* que acessem banco de dados.

PHPRunner é um gerador de código PHP, que ajuda a criar sites através de banco de dados, sem necessidade de conhecimento aprofundado de programação. Com PHPRunner, poderá criar paginas web facilmente utilizando a integração com as bases de dados como: MySQL, PostrgreSQL, Oracle, MS SQLServer, MS Access. O software pode ser facilmente aprendido, podendo criar um website em apenas 15 minutos.

PHPRunner oferece um grande número de modelos de aplicações e sites, com interface gráfica e estrutura de banco de dados. Todos os modelos são fáceis de trabalhar e totalmente personalizáveis. O modelo pode ser usado para um site, ou pode ser integrado com outras aplicações web. Alguns dos modelos disponíveis com o PHPRunner são: automóveis, anúncios classificados, base de conhecimento, imóveis, ofertas de empregos e notícias.

1.2.2.1 Vantagens

- Suporte aos bancos de dados: MySQL, PostrgreSQL, Oracle, MS SQLServer, MS Access;

1.2.2.2 Desvantagens

- Custo de $399,00 para o pacote Professional, e $599,00 o Enterprise Edition;
- Suportado somente Windows;
- Pouca documentação.

1.2.3 PHPForm

O site <www.phpform.org> é uma ferramenta on-line gratuita para criação de formulários. Não é necessário conhecimento para criação do formulário HTML. Com alguns cliques pode ser criado um formulário podendo selecionar a cor de fundo, podendo também incluir e aplicar configuração aos campos mais padrões de um formulário.

Existe a possibilidade de visualizar o formulário, podendo fazer o download dos arquivos gerados como: HTML, CSS e imagens. O código gerado segue uma estrutura padrão, facilitando bastante o processo de manutenção caso seja necessário, devido a não existir o "lixo" de código.

1.2.3.1 Vantagens

- Ferramenta *on-line* e gratuita;
- Facilidade na criação de formulários;

1.2.3.2 Desvantagens

- Gera apenas formulários HTML;
- Limitação em recursos e propriedades a serem definidas na configuração do formulário;
- Não existe documentação;

1.3 e-Gen Developer

O e-Gen Developer é uma IDE que proporciona o desenvolvimento rápido de aplicações para Web. É totalmente escrito em Java, baseado no *framework* Struts. O e-Gen foi criado para superar três desafios fundamentais enfrentados por gerentes de TI: aumentar a produtividade, reduzir os custos de manutenção e reduzir o tempo de treinamento e adaptação dos colaboradores (OLIVEIRA, 2005).

As aplicações desenvolvidas com o e-Gen não ficam sujeitas ao seu uso, pois o código gerado pode ser mantido por outras ferramentas disponíveis no mercado. Assim não existe dependência do uso do e-Gen, pois não é necessária sua instalação no servidor de produção (EGEN, 2010).

O e-Gen gera aplicações na tecnologia JAVA para ambiente Web. A comunicação com o banco de dados utiliza a API JDBC de forma otimizada através de uma camada de comunicação isolada (JDBCUtil). Essa camada permite o uso dos principais bancos de dados do mercado de forma independente e transparente, sem a necessidade de configurações especiais para a troca do fornecedor (EGEN, 2010).

De acordo com (BECKER, 200-) podemos citar as seguintes vantagens e desvantagens:

Dentre os pontos positivos podemos destacar o relacionamento entre tabelas. Podemos criar referências cruzadas e validar esses dados antes de inserirmos os dados na base, evitando assim inconsistência. E essas referências são feitas de maneira gráfica e intuitiva, facilitando ao programador a criação de um relacionamento e consistência forte nos dados utilizados.

Contudo essa abstração limita um pouco a funcionalidade do sistema, tornando difíceis as adaptações às funções disponibilizadas pelo e-Gen Developer. A atualização de campos de um formulário, por exemplo, não é possível através das funções existentes.

Por outro lado, a validação de campos, como de CPF, por exemplo, é bastante avançada e funcional.

Assim, o e-Gen é bom para criarmos aplicações de médio porte e que tenham acesso à base de dados. Para aplicações muito pequenas traz muito código agregado dificultando a

criação e execução. Para sistemas muito grandes, traz um nível de abstração muito grande, impedindo que sejam adaptadas funções de controle, o que pode muitas vezes ser desejável nesse tipo de sistema.

1.4 Considerações Finais

Atualmente existem diversas ferramentas de geração de código disponíveis no mercado, dentre as citadas por exemplo, Scriptcase e e-Gen prometem que ate mesmo pessoas com pouca experiência em programação sejam capazes de desenvolver um sistemas, podendo ser um problema pois estes geradores geram o código no anti-padrão spagueti sem divisões de camadas, o que muitas vezes até para um programador experiente, pode ser um problema na hora de dar manutenção.

É interessante notar que entre os desenvolvedores médios e avançados, existe certa resistência ou menosprezo pelo uso de geradores código, devido a muitos destes geradores gerarem uma grande quantidade de código desnecessário, seguindo muitas vezes do anti-padrão spagueti. Mas ao iniciar um projeto devem ser avaliado quais ferramentas podem ser utilizadas, e em quais etapas poderá utilizar, verificando também se a ferramenta funciona corretamente e se cumpre sua finalidade. Ao utilizar qualquer ferramenta que melhore a produtividade, poderá trazer alguns benefícios como por exemplo, poupar tempo de trabalhos repetitivos e pouco criativo, podendo passar a dedicar este tempo livre para outras atividades como melhorias do projeto.

2 FRAMEWORKS

Desde as primeiras aplicações o desenvolvedor se preocupa com a melhoria das ferramentas por ele utilizadas. Seguindo este conceito, cada vez mais surgem novas formas de se construir um software. Uma dessas formas é certamente o reaproveitamento de códigos que veio com o conceito de orientação a objeto. A partir disto, muitos outros conceitos foram agregados como o repositório de classes e objetos e em seguida o conceito de *frameworks*.

2.1 Conceito

Utilizar um *framework* de desenvolvimento significa reaproveitar o trabalho e o conhecimento de outra(s) pessoa(s) e seguir suas orientações. A palavra "reaproveitar" nos remete à orientação a objetos. O reaproveitamento de código não é privilégio nem invenção da orientação a objetos, mas esse paradigma possui técnicas eficazes para implementá-lo (LISBOA, 2009).

A ideia de qualquer *framework* é que os programadores ocupem-se mais em implementar as regras de negócio da aplicação do que tratar detalhes de baixo nível, que em geral são sempre os mesmos. O reaproveitamento de código reduz o tempo de desenvolvimento a cada projeto, conforme características em comum que vão sendo identificadas (LISBOA, 2008).

Em outras palavras o *framework* permite o desenvolvimento rápido de aplicações (RAD), o que faz economizar tempo, ajuda a criar aplicações mais sólidas e seguras além de reduzir a quantidade de código repetido. Os *frameworks* também permitem que os iniciantes criem aplicações mais estáveis garantindo uma boa relação entre o banco de dados e a camada externa de exibição (BELEM, 2010).

2.2 Utilização de *Frameworks*

A decisão por usar um *framework* de desenvolvimento reside na necessidade de estruturar os projetos de software, devido à grande complexidade que os mesmos alcançaram. Um projeto estruturado reduz custos, aumenta a qualidade da aplicação e reduz o tempo de desenvolvimento (LISBOA, 2009).

Ao utilizar um *framework*, muitas vezes nos deparamos com uma nova forma de programar ou até mesmo de se pensar sobre um sistema. Conforme (MINETTO, 2007) muitas vezes surge a sensação de estar "engessado", pois é preciso fazer as coisas da forma que o *framework* trabalha, de modo que qualquer coisa diferente requer um empenho melhor, devido à necessidade de aprender uma nova sintaxe, utilizar convenções de nomes de arquivos, variáveis e tabelas de banco de dados. Contudo, as vantagens a médio e longo prazo fazem valer esse pequeno esforço inicial.

Como todos os desenvolvedores que usam determinado *framework* programam usando as mesmas convenções, classes e bibliotecas, a manutenção de um programa torna-se muito mais fácil, mesmo que determinado script tenha sido escrito por outra equipe há vários meses. Isto contribui para que novos desenvolvedores ingressantes em uma equipe possam rapidamente se inteirar da forma como o *framework* é trabalhado, reduzindo custos e tempo para treinamento (MINETTO, 2007).

Outra vantagem dos *frameworks* de acordo com (SCHMITZ, 2010) é que eles facilitam a relação com bancos de dados e a camada externa de exibição. Isso acontece devido ao fato de os *frameworks* implementarem padrões de projeto que otimizam a comunicação com o banco de dados, e também deixarem a aplicação mais segura.

Conforme (SAUVÉ, 2010), observa como desvantagem o fato de que desenvolvê-los é algo complexo que exige planejamento. Além disso, implementar um software juntamente com um *framework*, leva mais tempo que simplesmente desenvolver uma aplicação. Os benefícios de desenvolver aplicações utilizando *frameworks* e reuso não surgem plenamente na sua primeira utilização, mas em longo prazo. Os ganhos de produtividade dependem do processo de maturidade que se adquire com o tempo de uso e correções de erros da ferramenta.

O número de *frameworks* de PHP e de outras linguagens cresce muito atualmente e, por isso, os usuários muitas vezes sentem dificuldades em escolher um *framework* para desenvolver seus projetos. Visto que cada *framework* tem suas particularidades, devemos estudá-lo antes de iniciar um projeto, já que uma escolha errada pode resultar em atraso (FREIRE, 2009).

2.3 PHP

O PHP sucede de um produto mais antigo, chamado PHP/FI. PHP/FI foi criado por Rasmus Lerdorf em 1995, inicialmente como simples *scripts* Perl como estatísticas de acesso para seu currículo online. Ele nomeou esta série de script de Personal Home Page Tools. Como mais funcionalidades foram requeridas, Rasmus escreveu uma implementação C muito maior, que era capaz de comunicar-se com base de dados, e possibilitava à usuários desenvolver simples aplicativos dinâmicos para Web. Rasmus resolveu disponibilizar o código fonte do PHP/FI para que todos pudessem ver, e também usá-lo, bem como fixar *bugs* e melhorar o código (PHP, 2010).

Os dois estudantes Andi Gutmans e Zeev Suraski que utilizavam PHP em um projeto acadêmico de comércio eletrônico resolveram, resolveram cooperar com Rasmus para aprimorar o PHP. Para tanto, reescreveram todo o código-fonte, tendo em vista melhorar sua performance e modularidade em aplicações complexas, para tanto resolveram batizado este núcleo de Zend Engine (Zeev + Andi).

Uma das grandes vantagens do PHP é sua facilidade de aprendizado. Ao ler poucas páginas de tutoriais ou de algum livro, um programador já é capaz de montar um formulário HTML e de criar um *script* PHP que processe os dados fornecidos pelo usuário. Isso favoreceu o rápido aumento do número de programadores e o surgimento de grandes softwares (MINETTO, 2007).

Contudo, essa facilidade de aprendizado permitiu o surgimento de programas, digamos, pouco eficientes, programas vulneráveis a ataques, com péssima manutenibilidade, os denominados "*spaggeti code*", com PHP e HTML inseridos no mesmo script, e também a dificuldade de se trabalhar em equipes, entre outros. Um dos argumentos usados por programadores de outras linguagens era que "PHP facilita a geração de programas ruins" (MINETTO, 2007).

2.4 *Design Patterns*

O termo *design pattern* ou simplesmente padrões de projetos, tem sido largamente utilizado no mundo da orientação a objetos para descrever formas de comunicação e relacionamentos entre objetos e classes de maneira a solucionar determinados problemas de projetos. A utilização de um *design pattern* não é a mais simples nem a mais rápida maneira de solucionar um determinado problema, mas é, sem duvida, a forma que traz maior flexibilidade e capacidade de reuso da solução, trazendo benefícios a médio e longo prazo na manutenibilidade do código-fonte (DALL'OGLIO, 2009).

> "Um *pattern* descreve um problema que ocorre com frequência em nosso ambiente, e então explica a essência da solução para este problema, de forma que tal solução possa ser utilizada milhões de outras vezes, sem ao menos repeti-la uma única vez" (DALL'OGLIO, 2009).

Segundo (GAMMA, 2000), os padrões são formados por quatro elementos básicos:

- **Nome do padrão**: é uma referência que pode ser usada para descrever um problema de projeto, suas soluções e conseqüências em uma ou duas palavras. O nome torna mais fácil pensar sobre os padrões de projetos, comunicá-los e avaliar custos e benefícios envolvidos;
- **Problema**: descreve quando o padrão deve ser aplicado. Neste elemento é explicado qual o problema e seu contexto. Pode descrever problemas de projeto específicos ou descrever problemas de estruturas de classe. Em alguns casos, o problema inclui uma lista de condições que devem ser satisfeitas para que faça sentido aplicar o padrão;
- **Solução**: descreve os elementos que compõe o projeto, seus relacionamentos, suas responsabilidades e colaborações. A solução não descreve um projeto concreto ou uma implementação em particular porque o padrão é como um gabarito que pode ser aplicado em diversas situações. O padrão fornece uma descrição abstrata de um problema de projeto e de como um arranjo geral de elementos resolve o mesmo;
- **Conseqüências**: são os resultados e análises das vantagens e desvantagens da aplicação do padrão. Embora as conseqüências sejam raramente mencionadas

> quando se descrevem decisões de projetos, elas são críticas para a avaliação de alternativas de projetos e para a compreensão dos custos e benefícios da aplicação do padrão. As conseqüências de um padrão incluem o seu impacto sobre a flexibilidade, a extensibilidade ou a portabilidade de um sistema.

A maioria dos padrões de projeto também torna o software mais passível de modificações. A razão para tal é que eles são soluções comprovadas pelo tempo; portanto, evoluíram em estruturas que podem tratar mudanças mais prontamente do que as que, muitas vezes, vêm primeiro a mente como solução (SHALLOWAY, 2004).

A maioria dos *frameworks* de desenvolvimento (pelo menos os atuais), como parte de sua proposta, adotam a programação orientada a objetos e alguns padrões de projeto. Um dos padrões que merece maior destaque é o padrão MVC (*Model, View, Controller*) que sugere a divisão do projeto de software em camadas lógicas, de modo a separar as funcionalidades. Isso torna fácil gerenciar mudanças na aplicação e reaproveitar funcionalidades em outros projetos (LISBOA, 2009).

2.5 O Padrão MVC (*Model View Controller*)

A utilização de MVC para aplicações web faz com que se desenvolva uma aplicação bem estruturada e com módulos bem distintos. Sendo que os *web designers*, que são os profissionais responsáveis pela parte visual da aplicação, não precisem entender rotinas de programação que são desenvolvidas por programadores propriamente ditos.

O MVC, conforme figura 2.1, é a forma de estruturar a aplicação de forma que os aspectos de programação fiquem divididos em três camadas, conforme definido por (MACORATTI, 2010) seguindo a seguinte arquitetura:

- ***Model*** **(Modelo)**: Representa os dados da aplicação e as regras do negócio que governam o acesso e a modificação dos dados. O modelo mantém o estado persistente do negócio e fornece ao controlador a capacidade de acessar as funcionalidades da aplicação encapsuladas pelo próprio modelo.

- ***View* (Visualização)**: Renderiza o conteúdo de uma parte particular do modelo e encaminha para o controlador as ações do usuário; acessa também os dados do modelo via controlador e define como esses dados devem ser apresentados.
- ***Controller* (Controle)**: controlador define o comportamento da aplicação, é ele que interpreta as ações do usuário e as mapeia para chamadas do modelo. Em um cliente de aplicações *Web* essas ações do usuário poderiam ser cliques de botões ou seleções de menus. As ações realizadas pelo modelo incluem ativar processos de negócio ou alterar o estado do modelo. Com base na ação do usuário e no resultado do processamento do modelo, o controlador seleciona uma visualização a ser exibida como parte da resposta a solicitação do usuário. Há normalmente um controlador para cada conjunto de funcionalidades relacionadas.

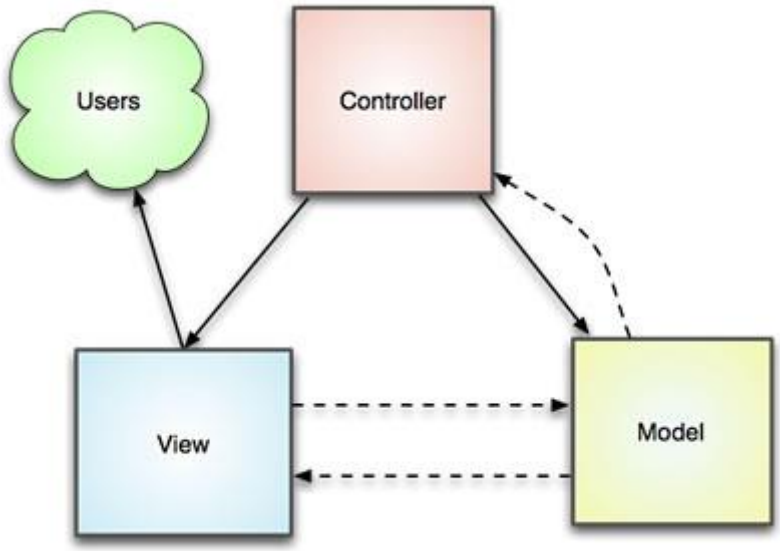

Figura 2.1: Modelo MVC.
Fonte: (ZEND, 2010)

2.6 ORM (Mapeamento objecto-relacional)

O Mapeador de Dados é uma camada de software que separa os objetos na memória do banco de dados. Sua responsabilidade é transferir dados entre os dois e também isolá-los um do outro. Com o Mapeador de Dados, os objetos na memória não precisam nem ao menos saber que há um banco de dados presente. Eles não precisam conter comandos SQL e certamente não precisam ter nenhum conhecimento do esquema do banco de dados.

O *pattern Data Mapper* se constitui em uma camada da aplicação que separa os objetos conceituais do banco de dados, permitindo transferir dados entre uma camada e outra de forma transparente, sem que os objetos do modelo conceitual precisem implementar métodos de persistência (DALL'OGLIO, 2009).

Para que o *Data Mapper* possa fazer isso, é necessário que o objeto conceitual (camada de negócio) ofereça métodos que permitam o *Data Mapper* inspecionar o valor de suas propriedades. A classe que implementará o *Data Mapper* deverá prover métodos de persistência e recuperação (obtenção) dos objetos de negócio, além de métodos para pesquisar objetos com base em diferentes padrões.

2.7 *Frameworks* PHP

Com o surgimento do *Rails*, vários programadores da comunidade *open source* em consenso de que programar para "o ambiente web" muitas vezes, era uma tarefa repetitiva e maçante, perceberam, que vários aspectos poderiam ser reaproveitados, e, principalmente, que programar poderia ser divertido (MINETTO, 2007).

O PHP é uma linguagem simples e fácil de aprender, e os *frameworks* usam dessa vantagem tornando a curva de aprendizado pequena. Também deve-se considerar que a comunidade PHP é enorme e, com isso, os *frameworks* PHP em geral possuem também uma comunidade muito forte, o que ajuda o desenvolvedor a tirar dúvidas e reaproveitar códigos de terceiros (SCHMITZ, 2010).

Atualmente existem diversos *frameworks* PHP disponíveis, entre os mais conhecidos podem ser citados: Zend Framework, CakePHP, Symfony, CodeIgniter e Yii. A escolha de um *framework* para desenvolvimento envolve alguns fatores que podem variar bastante dependendo do projeto. Mas existem pontos em comum, tais como a quantidade e a qualidade da documentação, a facilidade de encontrar desenvolvedores, o envolvimento da comunidade em relação ao *framework*, a curva de aprendizado, entre outros (FREIRE, 2009).

Com estes *frameworks* é possível trabalhar com a metodologia RAD, com isso é possível fazer tarefas triviais do desenvolvimento de uma aplicação, tais como criar um novo projeto, criar *controllers, view, models*, de forma muito simples e rápida.

2.7.1 Yii Framework

Yii é um *framework* de alta performance em PHP que utiliza componentes para o desenvolvimento de grandes aplicações Web. Permite máxima reutilização de códigos e pode acelerar significativamente o processo de desenvolvimento (YII, 2010).

O Yii é um *framework* genérico que pode ser usado para desenvolver praticamente todos os tipos de aplicações Web. Por ser um *framework* leve equipado com sofisticadas soluções em *caching*, é especialmente adequado para o desenvolvimento de aplicações com alto tráfego de dados, tais como portais, fóruns, sistemas de gerenciamento de conteúdo (CMS), sistemas de e-Commerce, etc (YII, 2010).

O Yii possui uma poderosa ferramenta para geração de código denominada de Yiic, esta ferramenta é acessada através de linha de comando, gerando os CRUD's com operações básicas como: *Create, Read, Update* e *Delete*, gerando também as telas HTML para manipulação destas operações.

Conforme (YII, 2010) a figura 2.2, ilustra como é o fluxo de dados através do *framework.*

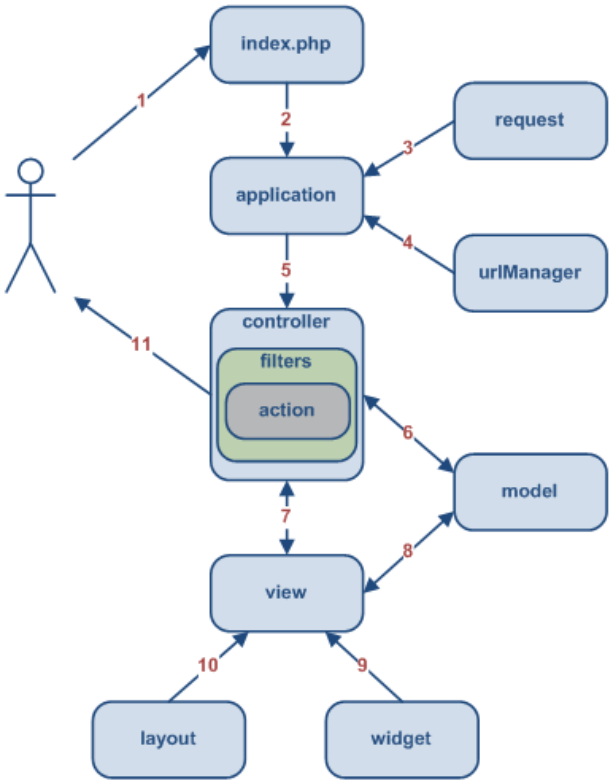

Figura 2.2: Ciclo de vida da requisição HTTP no Yii Framework
Fonte: (YII, 2010)

1. O usuário faz uma solicitação com a URL http://www.exemplo.com/index.php?r=post/show&id=1 e o servidor Web processa o pedido executando o script de *bootstrap* index.php.
2. O script de *bootstrap* cria uma instancia de aplicação (*application*) e a executa.
3. A aplicação obtém o as informações detalhadas da solicitação de um componente da aplicação chamado *request*.
4. A aplicação determina o controle e a ação requerida com a ajuda do componente chamado *urlManager*. Para este exemplo, o controle é post que se refere à classe PostController e a ação é show cujo significado real é determinado no controle.
5. A aplicação cria uma instancia do controle solicitado para poder lidar com a solicitação do usuário. O controle determina que a ação show refere-se a um método chamado *actionShow* no controle da classe. Em seguida, cria e executa filtros (por exemplo, o controle de acesso, *benchmarking*) associados a esta ação. A ação só é executada se permitida pelos filtros.
6. A ação lê um modelo Post cujo ID é 1 no Banco de Dados.
7. A ação processa a visão chamada *show*, com o *Post*.

8. A visão apresenta os atributos do modelo *Post*.
9. A visão executa alguns *widgets*.
10. O resultado do processamento da visão é embutido em um *layout*.
11. A ação conclui o processamento da visão e exibe o resultado ao usuário.

2.7.2 CodeIgniter

O CodeIgniter é um *framework* versátil e leve que possibilita a construção de aplicações e sistemas sob o paradigma da orientação a objetos e seguindo o design *pattern* MVC ou seja cada camada da aplicação tem suas responsabilidades e localizações físicas diferentes (CODEIGNITER, 2010).

O CodeIgniter tem uma abordagem mais solta em relação ao MVC, já que *Models* não são obrigatórios. Se não for necessária a utilização, ou entender que manter *models* irá gerar mais complexidade que o necessário, poderá ignorá-las e construir a aplicação apenas usando *Controllers* e *Views*. CodeIgniter também proporciona que incorpore seus próprios scripts, ou mesmo desenvolva bibliotecas para o sistema, lhe possibilitando trabalhar de um jeito que faça mais sentido para você.

O CodeIgniter foi escrito para ser compatível com o PHP 4, pois na época o PHP 5 não tinha seu uso disseminado, o CodeIgniter roda em PHP 5 só não utiliza as várias vantagens disponíveis nesta nova versão.

O *framework* veem com uma série de ferramentas úteis para tarefas genéricas. Vão desde manipulação de urls para criação de site com url amigável para os dispositivos de busca, até gerador de formulários que reproduzem exatamente uma tabela de banco de dados previamente criada.

Qualquer pessoa com um pouco de conhecimento em PHP consegue utilizar o *framework* e, consequentemente, produzir código com mais qualidade. O CodeIgniter é extremamente fácil de instalar, configurar e usar, e possui um grande número de tutoriais em vídeos, fóruns e *winkis*. Sua sintaxe é bem simples e intuitiva, o que gera um tempo de aprendizado muito curto (CODEIGNITER, 2010).

O CodeIgniter gera aplicações de forma ágil devido às suas bibliotecas que já trazem boa parte das soluções prontas como: acessar um banco de dados, enviar e-mails, criticar dados de um formulário, trabalhar com sessões, etc.

O Scaffolding foi planejado para uso apenas durante o desenvolvimento. Ele provê mínima segurança, apenas através de uma palavra "mágica", portanto qualquer pessoa que tenha acesso ao seu site pode potencialmente editar ou apagar suas informações.

Conforme (CODEIGNITER, 2010) a figura 2.3, ilustra como é o fluxo de dados através do *framework.*

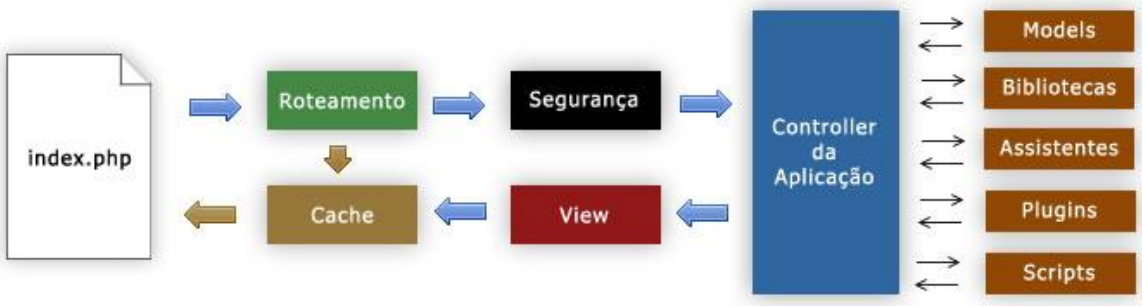

Figura 2.3: Ciclo de vida da requisição HTTP no CodeIgniter.
Fonte: (CODEIGNITER, 2010)

1. O index.php serve como um controlador primário, iniciando os recursos básicos necessários para rodar o CodeIgniter.
1. O roteador examina a requisição HTTP para determinar o que deve ser feito com ela.
2. Se já existe o arquivo “cacheado”, ele é enviado diretamente ao browser, pulando as outras etapas de execução.
3. Segurança. Antes da *controller* de aplicação ser carregado, a requisição HTTP e qualquer dado submetido pelo usuário é filtrado por segurança.
4. O *Controller* carrega o *model* as bibliotecas principais, *plugins*, assistentes e qualquer outro recurso necessário para processar a requisição específica.
5. A *View* finalizada é renderizada e então enviada ao browser para ser vista. Se o cache está habilitado, a *view* é 'cacheada' primeiro para que seja servida em requisições subseqüentes.

2.7.3 CakePHP

CakePHP é um *framework* em PHP gratuito, de código aberto, para desenvolvimento ágil, oferece uma estrutura fundamental para programadores criarem aplicações web de forma rápida e sem perda de flexibilidade.

CakePHP foi inspirado no Rails, tudo começou com a vontade de recriar no PHP o que os desenvolvedores do Rails diziam que não poderia ser feito.

O *framework* CakePHP provê uma base robusta para a aplicação. Ele pode manipular cada aspecto, desde o inicial da requisição do cliente até o ponto final da renderização da página. E uma vez que o *framework* segue os princípios do MVC, ele permite que personalize facilmente e estenda muitos aspectos da aplicação (CAKEPHP, 2010).

O *framework* também provê uma estrutura básica de organização, desde nomes de arquivos até nomes de tabelas de bancos de dados, mantendo toda sua aplicação consistente e lógica. Esse conceito é simples, porém muito poderoso.

Tendo um conhecimento básico a intermediário, conhecendo a sintaxe do PHP, o programador poderá desenvolver facilmente aplicações com o *framework*. É recomendado que o usuário procure se familiarizar e aprender mais sobre programação orientada a objeto. Isso facilitará bastante o aprendizado do CakePHP (SCHMITZ, 2010).

O Bake é uma ferramenta de geração de código do CakePHP, que é acessado via linha de comando, ajuda os programadores a criar as telas-padrão dos sites/sistemas que estão desenvolvendo. Ele gera os CRUD's básicos em cima de tabelas do banco de dados (*Create, Read, Update, Delete* - com paginação) e também gera, de forma simples, as telas de CRUD para tabelas relacionadas, permite adicionar validação aos formulários, entre outras coisas (SCHMITZ, 2010).

O recurso de scaffold é uma técnica que permite ao desenvolvedor definir e criar uma aplicação básica que possa inserir, selecionar, atualizar e excluir objetos (operações CRUD). Scaffold no CakePHP também possibilita que os desenvolvedores definam como os objetos estão relacionados entre si além de como criar e destruir estas relações (CAKEPHP, 2010).

Scaffold é uma excelente maneira de iniciar o desenvolvimento de partes prematuras de sua aplicação web. Primeiras versões de esquemas de bases de dados tendem a sofrer mudanças, o que é algo perfeitamente normal nas etapas iniciais do projeto da aplicação. Isto tem um lado negativo: um desenvolvedor web detesta criar formulários que nunca virão a ser efetivamente usados. Para minimizar o esforço do desenvolvedor, o recurso de scaffold foi incluído no CakePHP. O scaffold analisa as tabelas de sua base de dados e cria uma lista padronizada com botões de inserção, edição e exclusão, formulários padronizados para edição e visões padronizadas para visualização de um único registro da base de dados (CAKEPHP, 2010).

Conforme (CAKEPHP, 2010) a figura 2.4, ilustra como é o fluxo de dados através do *framework.*

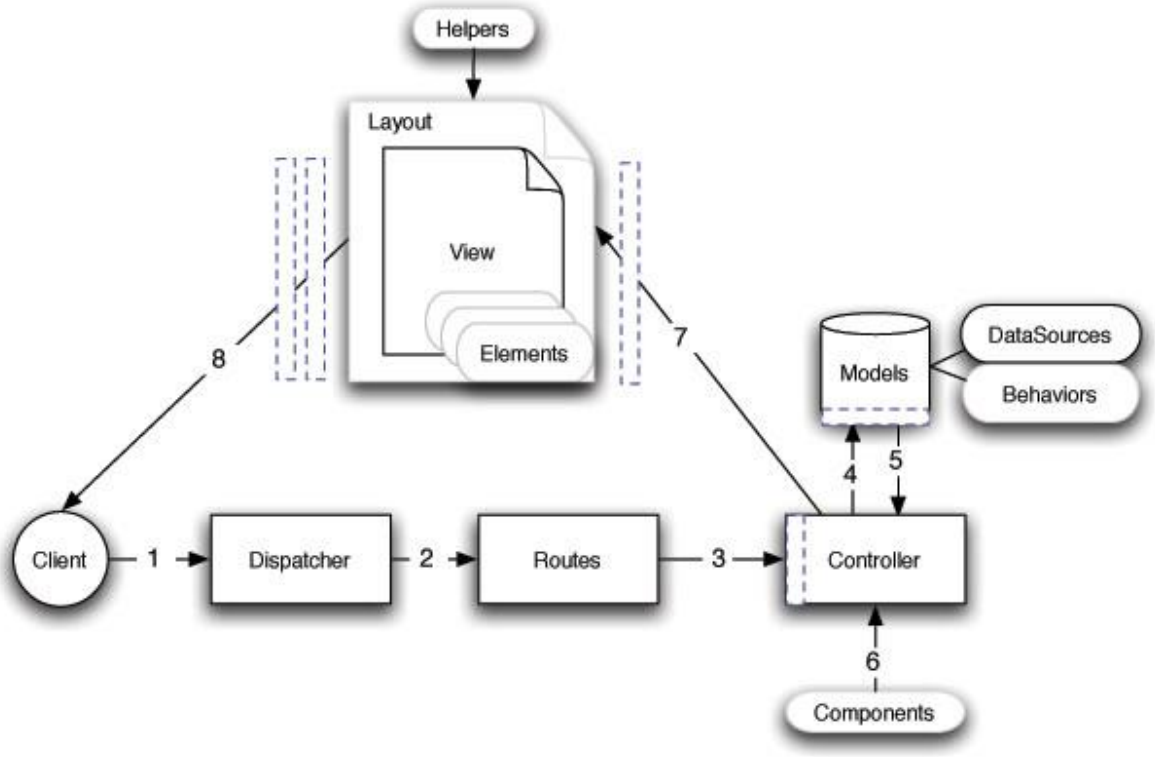

Figura 2.4: Ciclo de vida da requisição HTTP no CakePHP.
Fonte: (CAKEPHP, 2010)

1. Iniciada a requisição ao servidor web;
2. O roteador processa a URL, extraindo os parâmetros desta requisição: o controlador, ação e qualquer outro argumento que vai afetar na lógica do negócio durante esta requisição;

3. Usando rotas, a requisição da URL é mapeada para a ação do controlador (um método específico da classe do controlador). Neste caso, o método buy() do CakesController. O *callback* beforeFilter() do controlador é chamado antes de qualquer ação do controlador ser executada;
4. O controlador pode usar métodos para ter acesso aos dados da aplicação. Qualquer *callback* aplicável do modelo, *behaviors* e *DataSources* podem ser aplicados durante esta operação. Apesar de modelos não serem obrigatórios, todos os controladores do CakePHP inicialmente necessitam de pelo menos um modelo;
5. Depois de o modelo ter adquirido os dados, ele os retorna ao controlador. Podem ser aplicados *callbacks* no modelo;
6. O controlador pode usar componentes para refinar os dados ou efetuar outras operações (manipular sessões, autenticação ou enviar e-mails, por exemplo);
7. Uma vez que o controlador tenha usado os modelos e os componentes para preparar os dados suficientemente, estes dados são repassados às visões usando o método set() do controlador. *Callbacks* dos controladores podem ser aplicados antes dos dados serem enviados. A lógica da visão é efetuada, podendo incluir elementos ou ajudantes. Por padrão, as visões são sempre renderizadas dentro de um *layout*;
8. Além disso, *callbacks* dos controladores (como *afterFilter*) podem ser aplicados. Para completar, o código renderizado pela visão vai para o browser.

2.7.4 Zend Framework

Zend Framework é um *framework* de código aberto para o desenvolvimento de aplicações web e *WebServices* com PHP 5. Zend Framework é implementado usando 100% de código orientado a objeto. A estrutura dos componentes do Zend Framework é um tanto original; cada componente é projetado com poucas dependências em outros componentes. Essa arquitetura flexível permite que os desenvolvedores utilizem os componentes individualmente. Costuma-se chamar isso de modelo "use à vontade" (ZEND, 2010).

Uma das maiores vantagens do Zend Framework é que ele não te obriga a nada. Como o PHP sempre foi uma linguagem muito simples de usar (e isso a tornou uma das linguagens mais usadas do mundo), o Zend Framework segue esta mesma linha, tornando tudo muito simples (SCHMITZ, 2009).

Não é preciso estar em uma estrutura rígida de classes, pastas e arquivos. Não precisa carregar varias classes para fazer algo, e a documentação é extremamente rica (em inglês) recheada de exemplos (SCHMITZ, 2009).

Existem dois tipos de documentação fornecidos pelo *framework*: de API e de usuário final. A documentação de API é criada usando-se o PHPDocumenter e é gerada automaticamente usando-se comentários DocBlock especiais no código-fonte. Estes comentários encontram-se tipicamente logo acima de todas as classes, funções e declarações de variáveis membros. Uma das principais vantagens de se usar DocBlocks é que IDEs como o projeto PDT do Eclipse ou o Zend Studio são capazes de fornecer dicas em ferramentas de preenchimento automático durante a codificação, resultando em uma melhoria na produtividade do desenvolvedor (ALLEN, LO e BROWN, 2009).

O código-fonte do Zend Framework está licenciado sob a nova licença BSD. Isto dá bastante liberdade aos usuários para utilizarem o código em diversas aplicações diferentes, desde projetos com código aberto até produtos comerciais. Quando combinado com os requisitos de PI adequada, o Zend Framework está bem posicionado para ser usado por qualquer pessoa para qualquer propósito.

Alguns números sobre o *framework*, em agosto de 2009, o Zend Framework havia ultrapassado a marca de 10 milhões de *downloads*. Tinha mais de 500 contribuidores (que escreviam código), um guia de referência com mais de 1000 páginas e 500 exemplos e quase 6 milhões de hits no Google. Mais de 120 projetos no Sourceforge e mais de 280 no GoogleCode estendiam ou usavam-no como base, e mais de 4000 chamados abertos no tracer foram resolvidos (LISBOA, 2010).

Empresas como Google, Microsoft e StrikeIron mantêm parcerias com a Zend para fornecer interfaces de WebServices e outras tecnologias que pretendam disponibilizar aos desenvolvedores do Zend Framework (ZEND, 2010).

A partir da versão 1.6.0 do Zend Framework, passou a estar disponível um novo componente, o Zend_Tool, que é de grande utilidade e que facilita muito na hora de criar novos projetos e componentes.

O componente Zend_Tool ajuda a aplicar o desenvolvimento rápido de aplicações (RAD). Com essa ferramenta é possível acelerar o tempo de desenvolvimento, permitindo

através de linha de comando a criação de toda a estrutura do projeto, *controllers, views* e outros recursos relacionados ao projeto. Este componente reduz drasticamente o tempo necessário para a criação da estrutura de diretório e livre de erros, o que muitas vezes ocorre se comparado à criação manualmente.

O componente Zend_Tool representa um passo significativo não só em termos de ajudar os desenvolvedores Zend Framework mais experientes a estabelecer as bases rapidamente para novas aplicações, mas também em termos de ajudar os menos experientes a ultrapassar rapidamente o que historicamente tem sido um dos obstáculos mais significativos.

2.8 Avaliação dos *Frameworks*

A tabela 2.1 apresenta as principais características de cada *framework* para uso do desenvolvimento de aplicações. Os dados foram extraídos do site <www.phpframeworks.com> que é o principal avaliador de *frameworks* PHP.

	PHP4	PHP5	MVC	Multiple DB's	ORM	DB Objects	Templates	Caching	Validation	Ajax	Modules
CakePHP	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓
CodeIgniter	✓	✓	✓	✓	-	✓	✓	✓	✓	-	-
Yii	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zend	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓

Tabela 2.1: Tabela de comparação entre *frameworks*.
Fonte: Adaptada de <www.phpframeworks.com>

- **MVC**: Indica se o *framework* vem com suporte para uma configuração MVC.
- **Multiple DB's**: Indica se a estrutura suporta múltiplos bancos de dados sem ter que mudar nada.
- **ORM**: Indica se a estrutura suporta um mapeador de registro do objeto, geralmente uma implementação do *pattern* ActiveRecord.
- **DB Objects**: Indica se inclui outros objetos de banco, como por exemplo, o *pattern* TableGateWay.
- **Templates**: Indica se o *framework* tem uma ferramenta para compilar templates.
- **Caching**: Indica se a estrutura inclui cache de objetos ou de alguma outra forma de cache.

- **Validation**: Indica se o *framework* tem uma validação embutida ou componente de filtragem.
- **Ajax**: Indica se o *framework* vem com suporte para utilizar Ajax.
- **Modules**: Indica se o *framework* tem outros módulos, como um analisador de feed RSS, módulo de PDF ou qualquer outra coisa (útil).

Além destes quatros *frameworks* existem vários outros *frameworks* de desenvolvimento para PHP, inclusive alguns bem populares, como o Symphony e o Prado, a avaliação ficou restrita a somente a estes quatros, devido a eles estarem nas primeiras posições dos mais populares conforme o site <http://www.phpframeworks.com>.

3 DESENVOLVIMENTO DO GERADOR

Para este trabalho foi utilizado o Zend Framework, pois algo que deve ser levado em consideração para essa escolha é que, além da quantidade e qualidade da documentação, conforme (FREIRE, 2009), é a existência de grandes empresas apoiando o desenvolvimento, tais como Zend, Adobe, IBM, entre outras. Outro fator decisivo para a escolha foi o processo de desenvolvimento do *framework* ser bem definido e controlado pela empresa Zend. As alterações são feitas tomando-se cuidado para manter compatibilidade com versões anteriores, o que é muito importante para que o ciclo de vida do *software* seja estendido.

Além de uma ótima documentação e de empresas mantendo seu desenvolvimento, existe uma grande comunidade de desenvolvedores que utilizam o Zend Framework. Através da participação de listas de discussões sobre PHP, pode ser verificado que existe uma grande quantidade de empresas que estão utilizando esse *framework.*

O Zend Framework utiliza as melhores práticas de orientação a objetos, tem uma base de código rigorosamente testada, com foco na construção de aplicações Web 2.0, foi projetado para ser um *framework* simples, fornecendo suas bibliotecas de componentes leve e fracamente acoplados, oferecendo diversas funcionalidades que os desenvolvedores necessitam, permitindo ao desenvolvedor que personalize aquilo que não o atende plenamente.

Conforme (LISBOA, 2010), algo que deve ser levado em consideração, é o fato de o *framework* ser sustentado por uma grande empresa. No mundo do software livre, não há garantia de uma nova funcionalidade porque a comunidade não é uma empresa. E a palavra “garantia” é algo muito importante para uma organização.

Embora o *framework* apresente diversas características, atualmente ele não possui um gerador de código capaz de gerar tanto código *front-end* quanto *back-end.* Neste trabalho foi desenvolvida uma ferramenta que, através de uma interface gráfica, o usuário configura o banco de dados a ser utilizado; após isso ele monta o formulário HTML, definindo nas propriedades do campo, a qual tabela e campo ele pertence. Nas propriedades de cada campo além de poder definir a qual propriedade e tabela pertencem, haverá um modelo pré-definido onde o usuário poderá aplicar algumas configurações, tais como: editar o título do campo, se o campo é obrigatório, se existe algum tipo de validação a ser aplicada, etc.

Tendo realizado as configurações acima descrita, estas informações serão enviadas para o módulo PHP, que será responsável por gerar todo o código PHP necessário para o funcionamento, conforme a estrutura de projeto do Zend Framework:

Todos os arquivos gerados poderão ser facilmente editados através de qualquer editor de texto, sem qualquer dependência do gerador. Em um primeiro momento o gerador desenvolvido não cria uma aplicação completa. Neste momento o objetivo é o desenvolvimento de um gerador capaz de gerar códigos rotineiros, códigos que seguem um padrão e que muitas vezes tomam muito tempo do programador.

Para o desenvolvimento desta aplicação algumas tecnologias foram definidas para facilitar e agilizar o processo de desenvolvimento, evitando assim que a criação de cada componente fosse realizada a partir do zero. Para a manipulação da interface foi utilizada a biblioteca *JavaScript* jQuery. Inicialmente o gerador atenderá somente aos SGBDs MySQL e PostgreSQL podendo ser facilmente estendida para outros SGBDs.

3.1 jQuery

A biblioteca jQuery destina-se a adicionar interatividade e dinamismo às páginas web, proporcionando ao desenvolvedor funcionalidades necessárias à criação de *scripts* que visem incrementar, de forma progressiva e não obstrutiva, a usabilidade, a acessibilidade e o *design*, enriquecendo a experiência do usuário (SILVA, 2009).

JQuery é uma biblioteca JavaScript disponibilizada como *software* livre e aberta, cujo emprego e uso é regido segundo as regras de licença estabelecidas pelo MIT – *Massachusetts Institute of Technology* e pelo GPL – GNU *General Public License*. Isto,

resumidamente, significa que você pode usar a biblioteca gratuitamente tanto em desenvolvimento de projetos pessoais como comerciais.

A jQuery foi criada com a preocupação de ser uma biblioteca em conformidade com os Padrões Web. Desta forma, ela é compatível com qualquer sistema operacional e navegador, além de oferecer suporte total para as CSS 3 (SILVA, 2009).

Por ser distribuída como software livre, jQuery tem o apoio e o envolvimento de uma considerável comunidade. Desenvolvedores do mundo todo tem contribuído em larga escala com novas ideias, *scripts*, *plug-ins*, extensões e toda sorte de implementações, com a finalidade de incrementar não só a biblioteca, mas também as técnicas de desenvolvimento jQuery (SILVA, 2009).

3.2 SGBD

Neste primeiro momento esse trabalho será desenvolvido para os SGBDs (Sistema Gerenciador de Banco de Dados) PostgreSQL e MySQL por serem alternativas de software livre, além de ferramentas *open source.* De acordo com (HOSTWEB, 2010):

- O PostgreSQL é otimizado para aplicações complexas, isto é, que envolvem grandes volumes de dados ou que tratam de informações críticas. Assim, para um sistema de comércio eletrônico de porte médio/alto, por exemplo, o PostgreSQL é mais interessante, já que esse SGBD é capaz de lidar de maneira satisfatória com o volume de dados gerado pelas operações de consulta e venda.
- O MySQL, por sua vez, é focado na agilidade. Assim, se sua aplicação necessita de retornos rápidos e não envolve operações complexas, o MySQL é a opção mais adequada, pois é otimizado para proporcionar processamento rápido dos dados e tempo curto de resposta sem exigir muito do hardware.

Um dos objetivos desse trabalho não é ficar restrito somente a estes dois SGBDs. Portanto essa ferramenta poderá facilmente ser estendida para outros SGBDs tais como MS SQLServer, Oracle, Firebird entre outros.

3.3 Protótipo

Enxergou-se como essencial a criação de um protótipo da interface da ferramenta, a fim de identificar necessidades, falhas e melhorias a serem aplicadas às ideias pré-concebidas. O protótipo desenvolvido para essa primeira etapa tem as características demonstradas na figura 3.1:

1. Corresponde ao menu com as opções de criação de formulário. Ao clicar sobre um item será criado automaticamente um novo item na área do formulário.
2. Ao clicar sobre um item, a linha do item selecionado ficará em destaque.
3. Nesta opção que representa o menu de ações para o item selecionado, tem-se:
    a. O primeiro item é responsável em mover a linha para cima ou para baixo.
    b. A segunda opção abrirá uma janela contendo as propriedades do campo conforme figura 3.2, essas propriedades são nome do campo, tamanho, alinhamento, tipo de validação, etc, onde o usuário poderá definir conforme sua necessidade.
    c. O terceiro item será responsável em remover a linha em destaque.

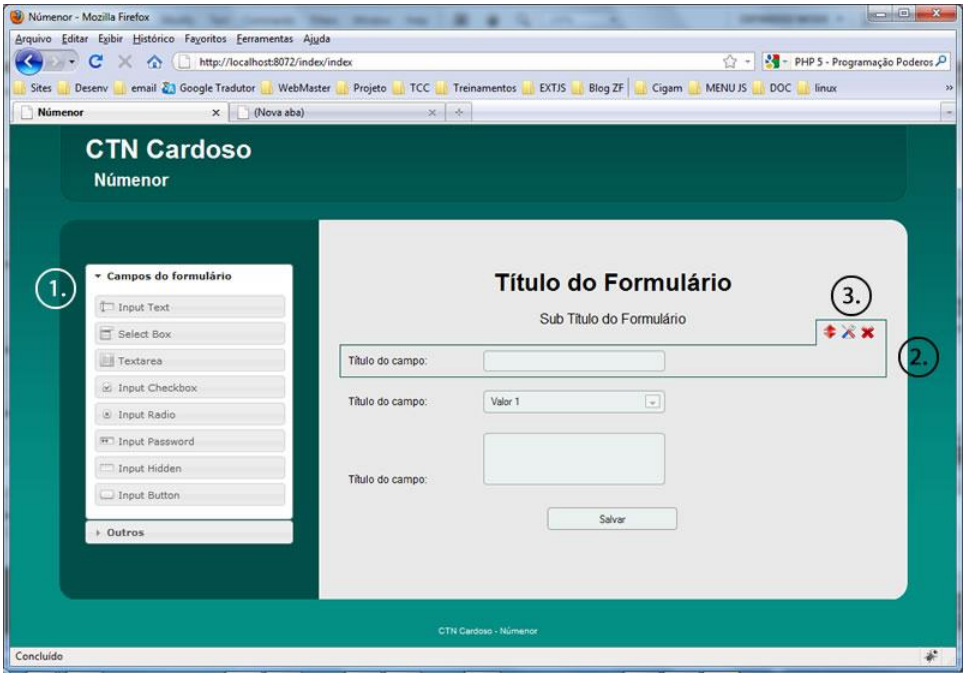

Figura 3.1: Protótipo.

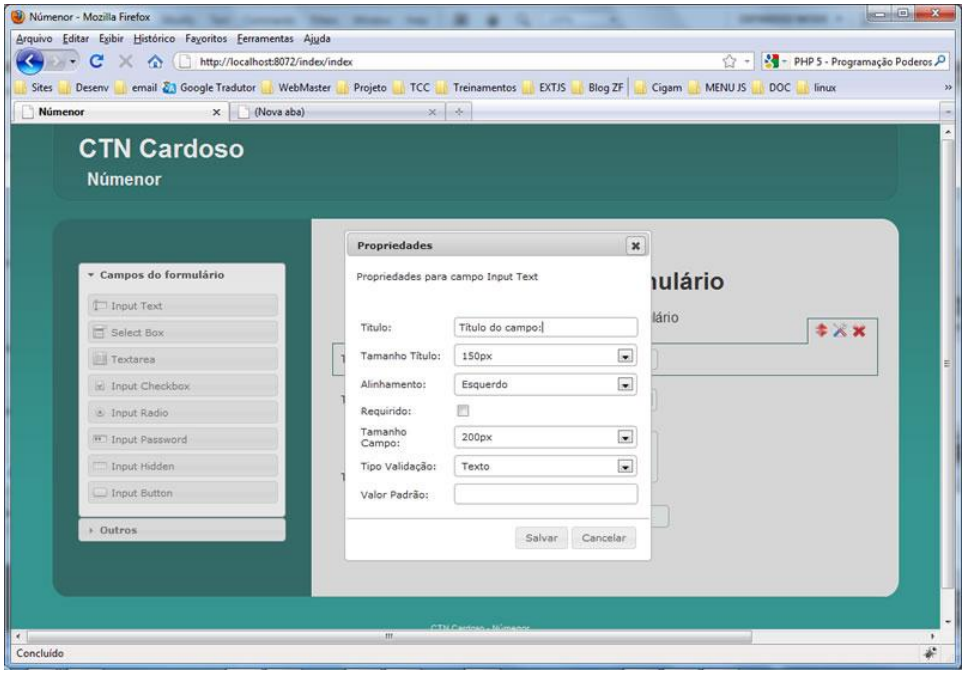

Figura 3.2: Propriedades protótipo.

Ao efetuar alguns testes de usabilidade com este protótipo foram identificados alguns itens que deveriam ser melhorados tais como, a possibilidade de trocar o valor de uma propriedade sem a necessidade de abrir uma nova janela, facilitando assim, o processo de criação dos formulários. Para isso foi desenvolvido um novo protótipo, e conforme figura 3.3, podemos notar algumas mudanças como:

- Layout similar a IDEs de desenvolvimento, como por exemplo, o Visual Studio;
- Menu do lado esquerdo para edição das propriedades, ao invés de abertura de *popup*, como no protótipo anterior;
- Ao trocar o valor de uma propriedade, o seu comportamento já pode ser visualizado no formulário, sem a necessidade de abrir uma nova janela e salvar para visualizar as alterações;
- Ao invés de ter que clicar em cima do ícone para mover uma linha, basta agora simplesmente clicar sobre a linha e arrastá-la ao local desejado.

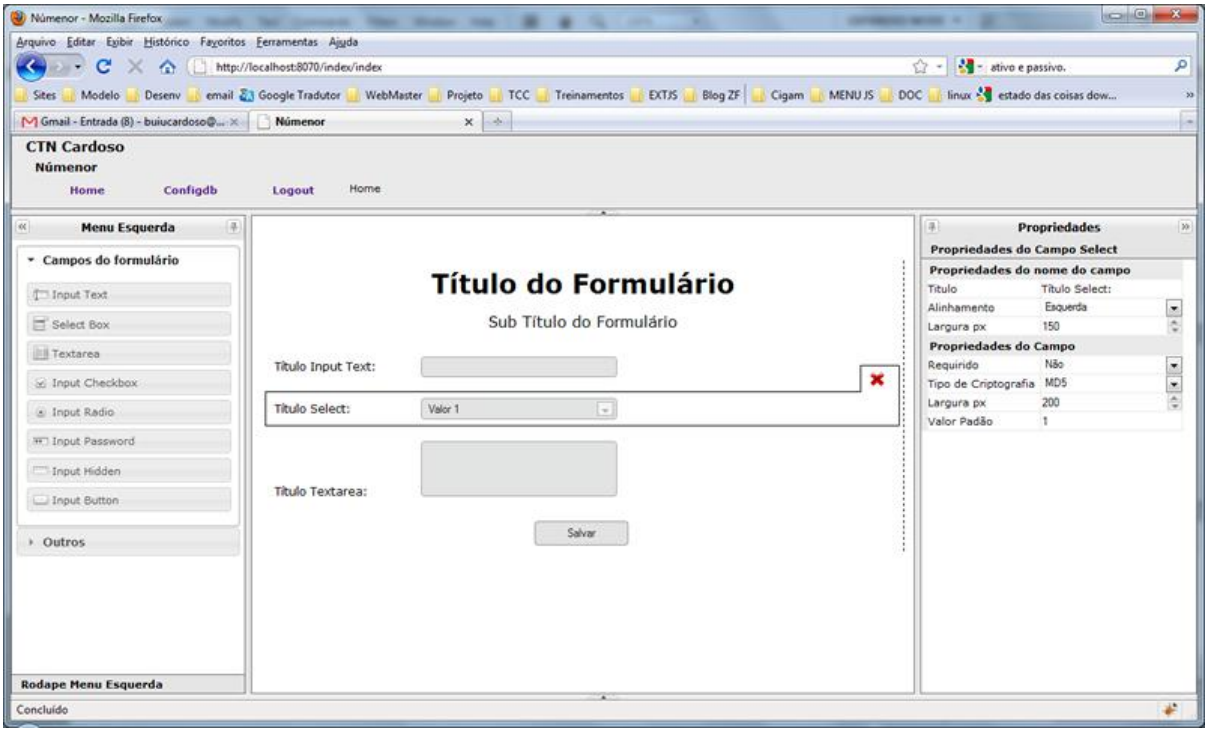

Figura 3.3: Novo protótipo.

3.4 Recuperação dos Metadados dos SGBDs

Para buscar as informações das tabelas do banco de dados foram desenvolvidas classes responsáveis em retornar a estrutura de todo o banco de dados, estrutura de apenas uma tabela ou informações especificas como as *primary key* e *foreign key*. Cada classe é responsável em retornar informações especificas de um único banco de dados.

Foi definido um padrão de como deverá ser o retorno das informações contendo a estrutura do banco de dados, facilitando assim que o gerador seja estendido para outros SGBDs, onde às classes responsáveis em buscar estas informações devem respeitar este padrão para que o gerador funcione de forma correta.

O padrão de retorno é um *array* conforme pode ser observado na figura 3.4. Caso o banco de dados trabalhe com *schemas*, a chave do primeiro nó do *array* deverá ser o nome do *schema*. Caso o banco de dados não trabalhe com *schemas* este nome deverá ser o do banco de dados. A estrutura segue com o nome da tabela, nome dos campos e suas propriedades. Caso o campo faça referência à outra tabela deverá de ser criada a estrutura contendo os dados da outra tabela e o campo de referência.

```
$banco = array(
    'nome schema' => array(
        'nome tabela' => array(
            'nome campo' => array(
                'tipo' => 'tipo de campo',
                'nulo' => 'true | false',
                'tamanho' => 'quantidade de caracteres',
                'chave_primaria' => 'true | false',
                'chave_estrangeira' => 'true | false',
                'referencia' => array(
                    'nome_schema_ref' => 'nome do schema',
                    'nome_tabela_ref' => 'nome da tabela',
                    'nome_campo_ref' => 'nome do campo',
                    'nome_chave_ref' => 'nome da chave de ligação'
                )
            )
        )
    )
);
```

Figura 3.4: Estrutura de retorno.

A busca pelas informações da estrutura das tabelas é realizada através de consultas SQL às tabelas *information_schema.* Essas tabelas compõem um banco de informações (metadados) sobre o próprio banco de dados, e consiste em um conjunto de visões contendo informações sobre os objetos definidos no banco.

4 FERRAMENTA PROPOSTA - NÚMENOR

O objetivo deste capítulo é descrever a ferramenta proposta, denominada de Númenor, suas funcionalidades, e demonstrar os passos necessários para gerar o código-fonte desejado.

4.1 Funcionamento da Ferramenta

A ideia principal da ferramenta é permitir a geração de código-fonte para as camadas *Model*, *View* e *Controller* do padrão MVC, para o Zend Framework, isso será realizado a partir de informações baseado em informações definidas através do gerador de formulário e de informações das tabelas do banco de dados. Para atingir este objetivo, o usuário da ferramenta deve:

1. **Recuperar os metadados do banco de dados:** este processo se dá informando os dados necessários para efetuar a conexão com o banco de dados, ao criar o projeto.

2. **Configurar o formulário a ser gerado:** neste passo o usuário seleciona os campos que o formulário deverá conter, associando-os a um determinado campo da tabela.

3. **Gerar o código-fonte:** através das definições do passo anterior são geradas as classes necessárias para o funcionamento do formulário.

A figura 4.1 mostra os passos apresentados anteriormente através de um diagrama de atividades.

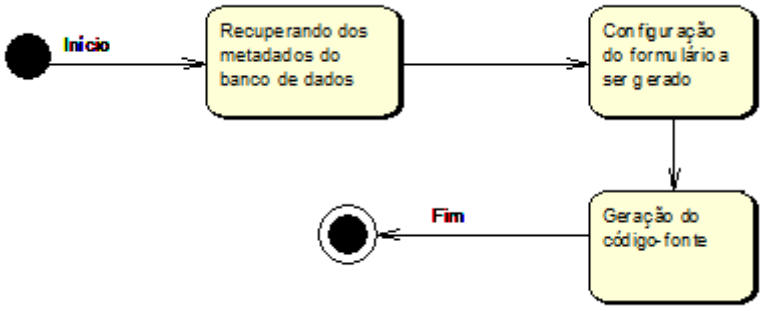

Figura 4.1: Diagrama de atividades para o gerador Númenor.

4.2 Instalação

O código fonte do projeto Númenor estará disponível através do site <http://numenor.com.br> para efetuar a instalação é necessário efetuar o download dos arquivos do projeto. Deve ser feito também o download do framework Zend através do site <http://framework.zend.com>. Efetuado os downloads, descompacte o primeiro arquivo que contem o projeto Númenor, em seguida descompacte os arquivos do Zend Framework, copiando o diretório "library/Zend" para dentro do diretório "library" no projeto Númenor.

O Apache deverá de estar configurado apontando o *documentRoot* para o diretório "*public*", o Apache também deverá de ter o modulo "*mod_rewrite*" habilitado, este modulo é utilizado para a reescrita das urls tornando-as urls amigáveis.

Tendo feito isso o projeto Númenor estará pronto para uso, não será necessário instalação de banco de dados para o funcionamento da ferramenta, devido ao Número trabalhar com o banco de dado SQLite e o mesmo já estar embutido na estrutura de pastas do projeto.

Ao executar pela primeira vez o gerador Númenor irá solicitar usuário e senha, por padrão o gerador vem com usuário e senha definido como numenor, podendo ser alterada a qualquer momento através do menu "Opções –> Usuário e Senha".

4.3 Demonstração da ferramenta

Esta seção objetiva apresentar o funcionamento da ferramenta através de um passo-a-passo.

Ao entrar no sistema existe apenas o menu superior com uma única funcionalidade habilitada, possibilitando ao usuário criar um novo projeto, conforme pode ser visualizado na figura 4.2.

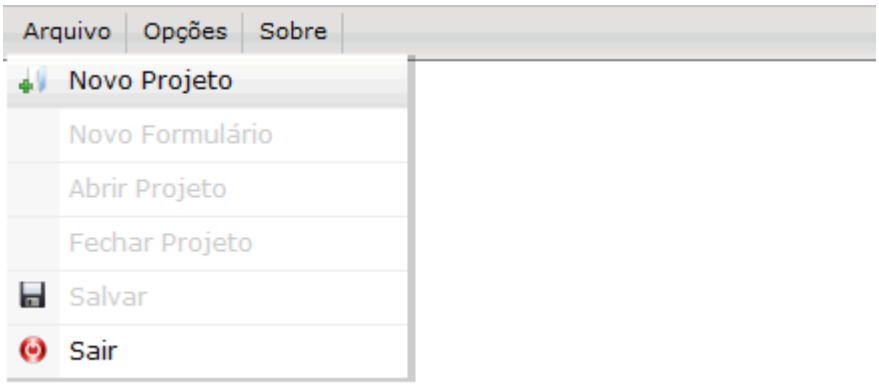

Figura 4.2: Novo Projeto.

Ao selecionar a opção “Novo Projeto” será exibida uma nova janela onde o usuário poderá definir algumas informações a respeito do projeto, estas informações são divididas em três abas. Na primeira aba, conforme pode ser visualizada na figura 4.3 será definido o nome do projeto.

Figura 4.3: Novo Projeto, definição do nome do projeto.

Na figura 4.4 é exibida a segunda aba, onde serão definidos os dados de conexão com o banco de dados. Existe a opção de criar um projeto sem necessidade de utilizar uma conexão com o banco de dados, deixando assim disponível a possibilidade de apenas criar e gerar código-fonte para formulários, para isso basta selecionar a opção “Não utilizar conexão com o banco de dados”. Caso esta opção não seja habilitada o usuário deverá fornecer os dados de conexão com o banco de dados e em seguida validar estes dados, clicando sobre o botão “Testar conexão”.

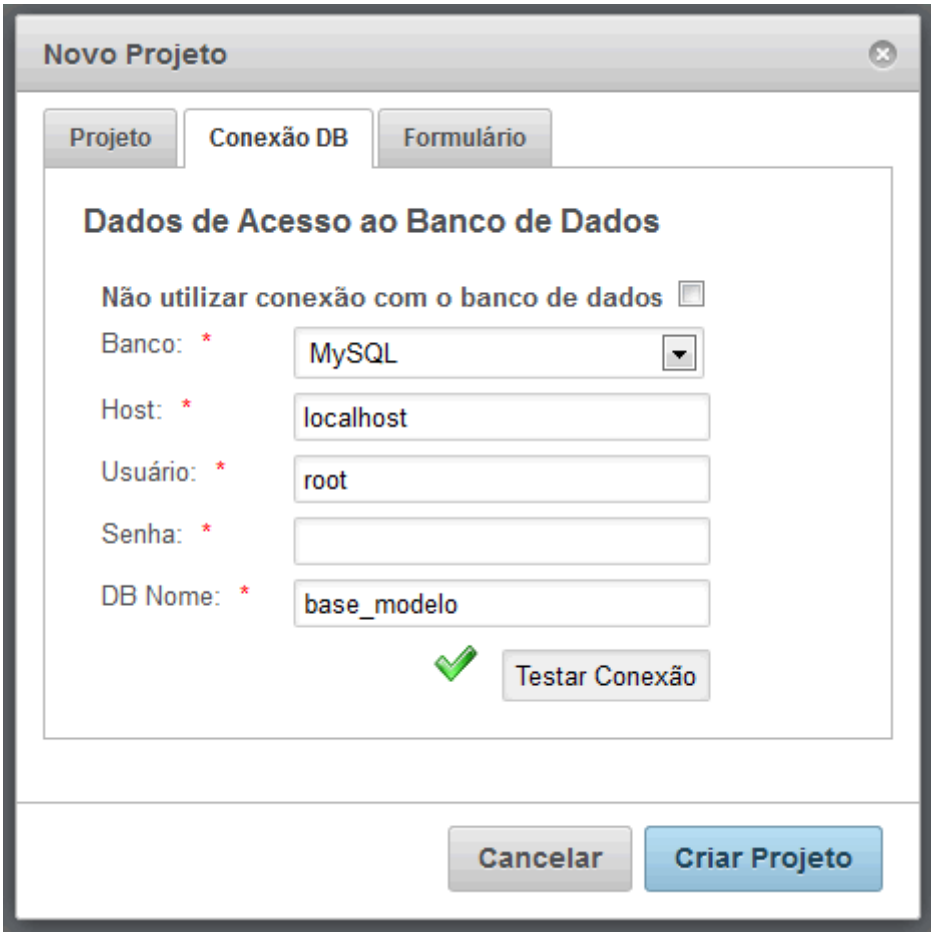

Figura 4.4: Novo Projeto, definição dados de conexão.

A figura 4.5 exibe a terceira aba, nela será informado o nome do formulário que será criado automaticamente ao criar projeto. Existem ainda outros dois campos: “Nome Controller” e “Nome Action” cujo o preenchimento não é obrigatório. Essas informações representam qual o *controller* e *action* deverão fazer a chamada ao formulário. Caso estas informações não tenham sido inseridas, o gerador assumirá o nome “index” como padrão para o *controller* e *action*.

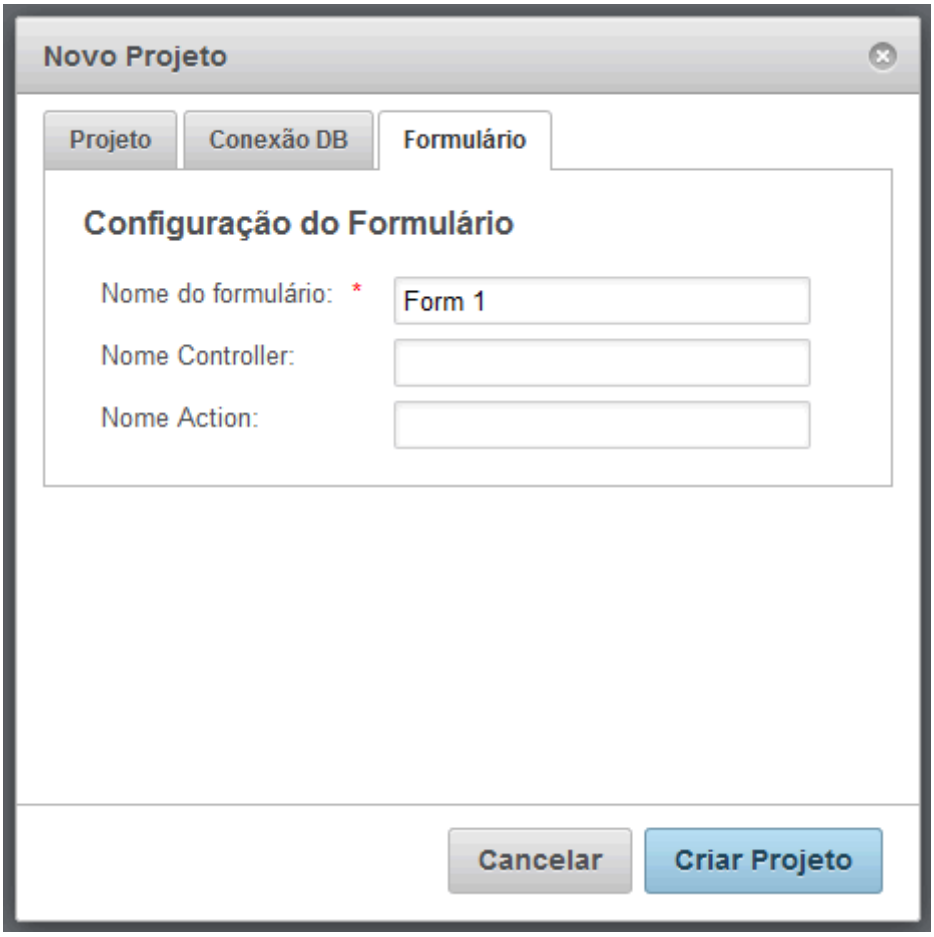

Figura 4.5: Novo Projeto, Dados do formulário.

Após definir todos os dados necessários para criar o projeto, caso tenha sido definido os dados de conexão, aparecerá a tela exibida na figura 4.6 que irá buscar a estrutura de todas as tabelas do banco de dados, deixando assim disponível para que o usuário possa verificar a qualquer momento a estrutura de uma tabela, e utilizá-la para criar os formulários.

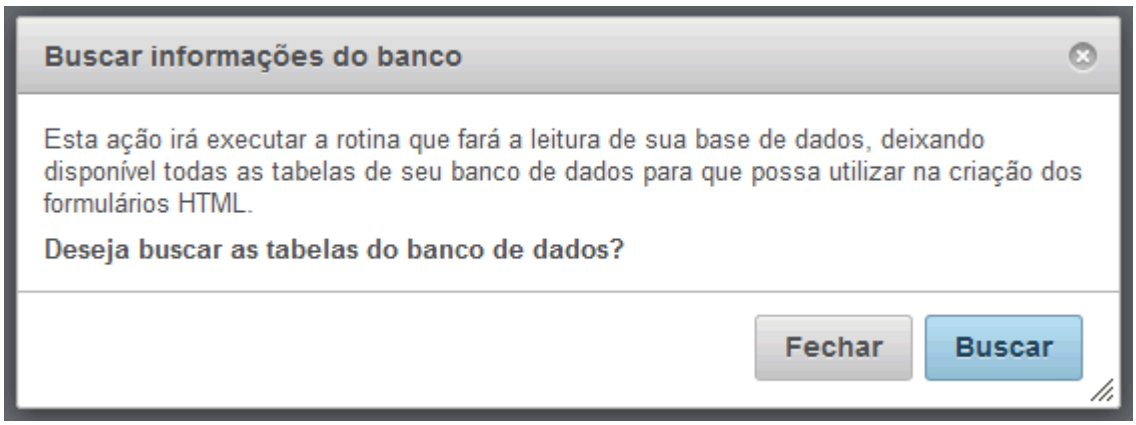

Figura 4.6: Buscar informações do banco.

Seguindo a mesma linha do segundo protótipo, a ferramenta desenvolvida apresenta outras características, buscando sempre uma interface amigável, de fácil utilização e muito próxima a outras IDEs existentes no mercado. Dessa forma pretende-se facilitar bastante a adaptação dos usuários à nova ferramenta. Ela segue algumas características observadas na figura 4.7 descritas abaixo:

1. Menu superior que representa onde o usuário poderá executar ações e configurações referentes ao projeto e do gerador Númenor.

2. Menu contendo todos os objetos que poderão ser utilizados para a construção dos formulários, como por exemplo, campos, botões, etc.

3. Menu com ações exclusivas para os formulários, como ação para gerar código, visualizar tabelas do banco de dados.

4. Área destinada à geração dos formulários, local onde o usuário poderá manipular e posicionar os campos de acordo com sua necessidade.

5. Menu contendo as propriedades dos objetos, onde o usuário poderá definir os valores dos atributos de cada objeto, como por exemplo, o titulo do campo, se é obrigatório ou não, largura do campo, etc.

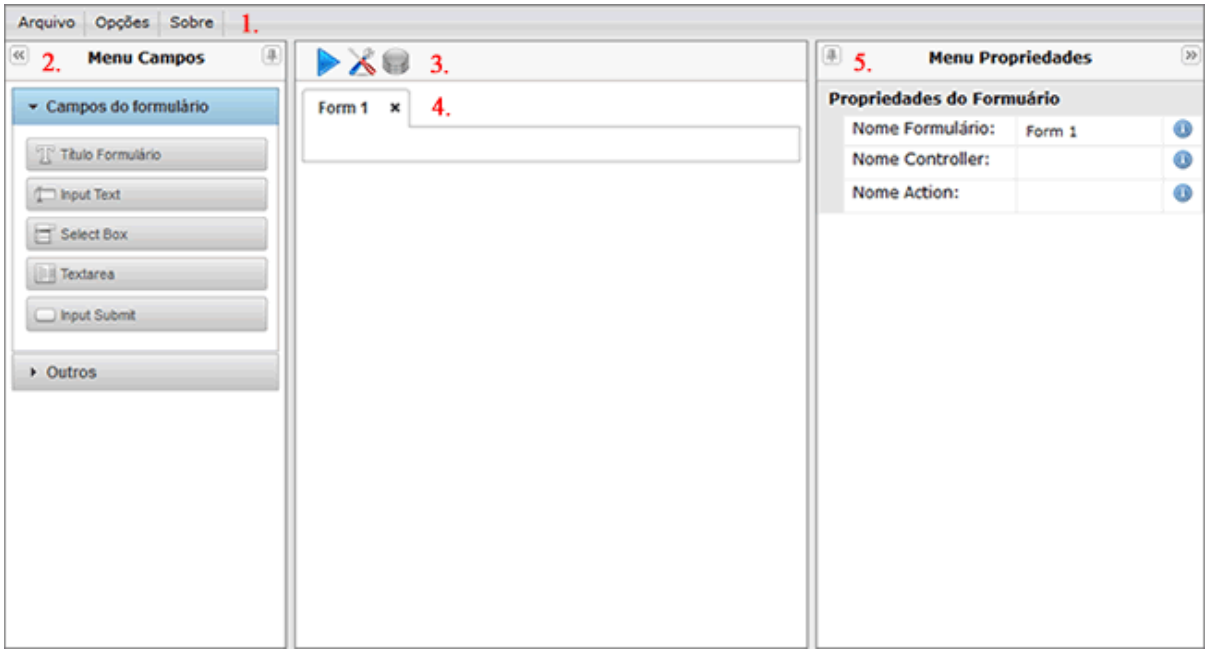

Figura 4.7: Interface da ferramenta.

Ao clicar sobre um dos objetos que poderão ser utilizados na construção dos formulários, será criada automaticamente uma linha no final da área do formulário deixando assim disponível para que o usuário possa editar suas propriedades. Para isso basta o usuário clicar sobre a linha desejada, a linha selecionada ficará em destaque e suas propriedades serão exibidas no menu à direita, conforme podemos visualizar na figura 4.8. Para cada linha no menu de propriedades existe um ícone contendo informações do que representa e qual a utilidade de determinada propriedade.

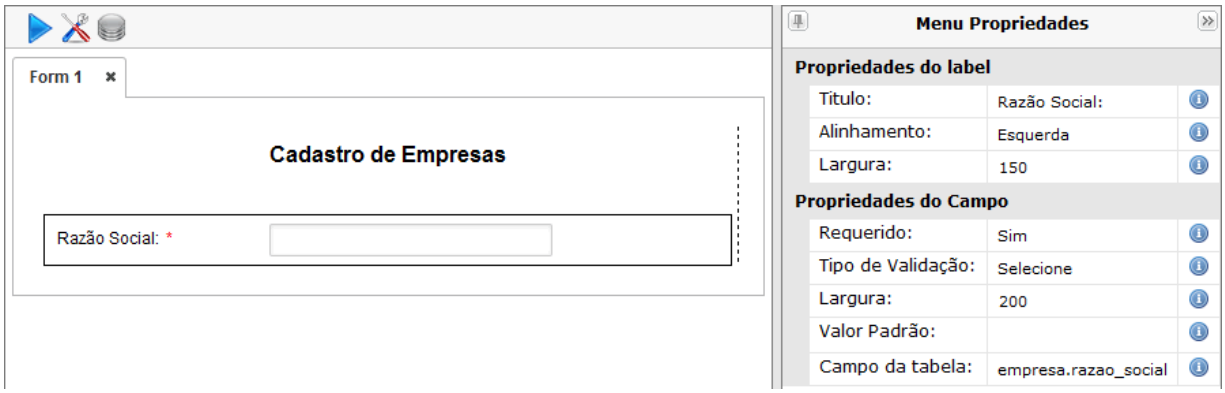

Figura 4.8: Área do formulário e propriedades.

Campos *text* possuem uma propriedade que é exclusiva desse campo, a propriedade "Tipo de Validação". Ao ser preenchida, essa propriedade será utilizada na hora de validar o formulário, verificando se os dados fornecidos estão de acordo com o esperado. Atualmente existem algumas opções de validações já cadastradas, como por exemplo, a validação de máscaras de CPF, CNPJ, Telefone, etc.

A validação é feita através de expressão regular. Conforme (JARGAS, 2009) expressão regular é uma composição de símbolos e caracteres com funções especiais, que agrupados entre si e com caracteres literais, formam uma sequência, uma expressão. Esta expressão é interpretada como uma regra, que indicará sucesso se uma entrada de dados qualquer combinar com essa regra, ou seja, obedecer exatamente a todas as suas condições.

Caso o usuário sinta a necessidade de utilizar outro tipo de validação que não esteja cadastrado, poderá efetuar o cadastro de uma nova opção através do menu "Opção –> Tipo Validação" e será exibida a tela conforme figura 4.9.

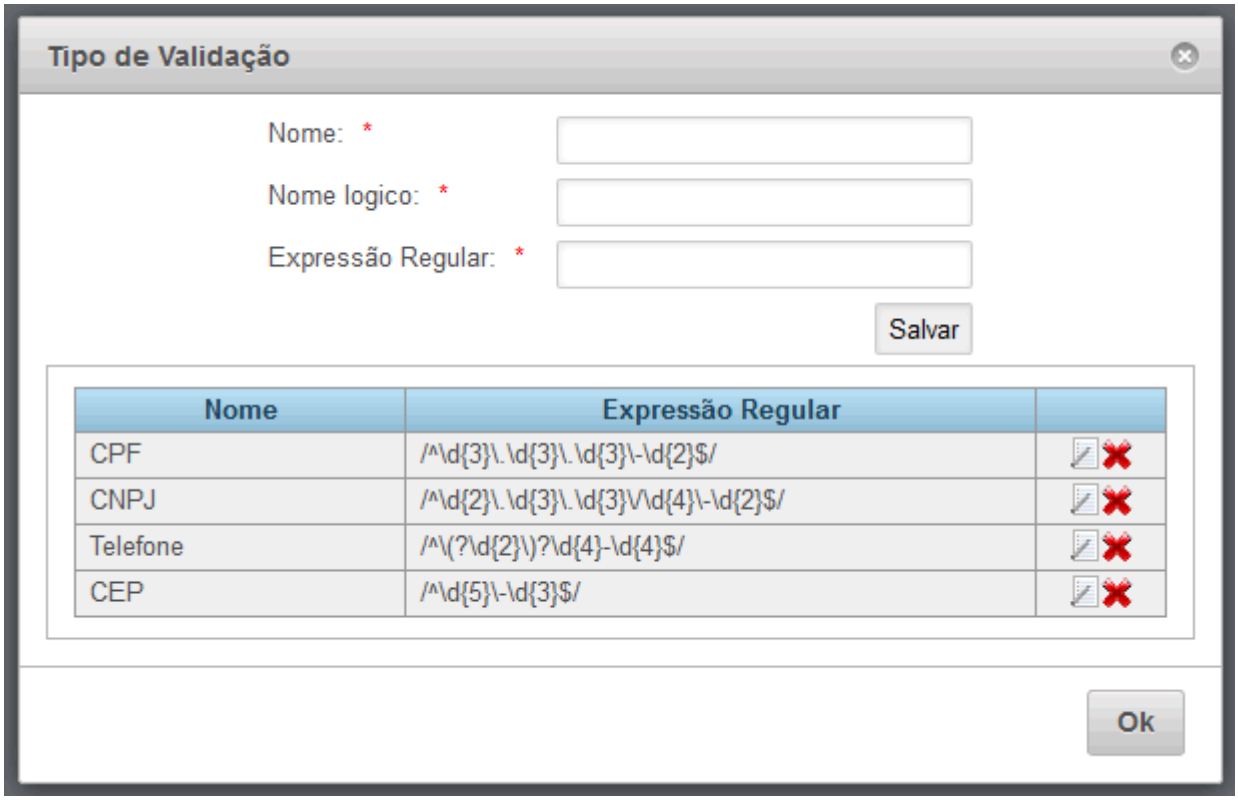

Figura 4.9: Cadastro dos tipos de validações.

Para efetuar o cadastro de uma nova opção de tipo de validação, será necessário o preenchimento de três campos:

1. **Nome**: o nome que será exibido no menu de propriedades.

2. **Nome Lógico**: nome que ficará nos atributos do campo para que possa ser validado através de JavaScript antes que o formulário seja submetido. Este nome não poderá ter espaços e caracteres especiais.

3. **Expressão Regular**: expressão que irá efetuar a validação dos dados, tanto no servidor quanto no cliente.

Outra opção que merece destaque no menu de propriedades é a opção “Campo da tabela”. Através dessa opção o usuário poderá selecionar em qual campo e tabela os dados deverão ser salvos. Conforme podemos visualizar na figura 4.10, na primeira coluna serão exibidas todas as tabelas do banco de dados. Ao selecionar uma tabela, na segunda coluna serão exibidos os nomes dos campos dessa tabela.

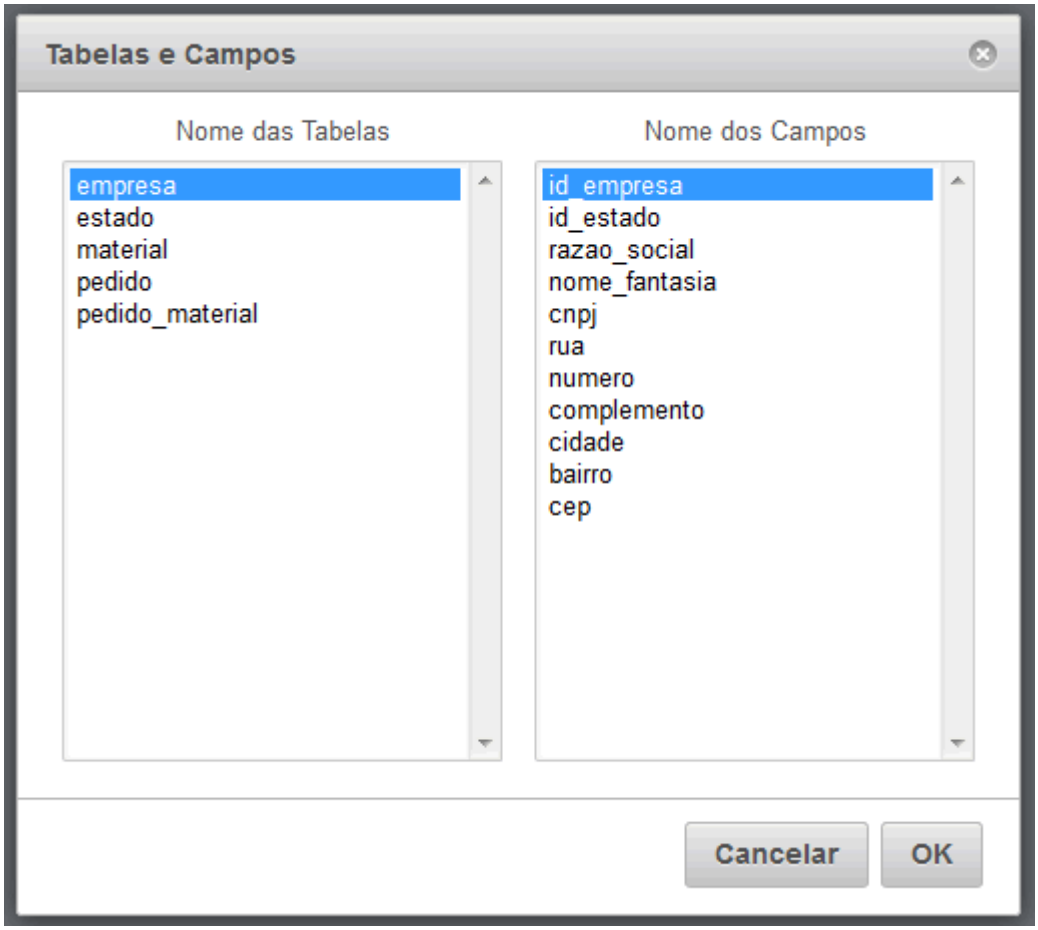

Figura 4.10: Tabelas e campos.

Os campos do tipo *selectbox* também possuem uma propriedade que merece ser destacada, a opção "Options do campo". A partir dela o usuário definirá a forma de preenchimento para as *options* de um campo *selectbox*. Conforme podemos visualizar através da figura 4.11 existem duas formas de fazer isto: a primeira opção é através do cadastro manual das *options*, onde o campo "Texto" deverá ser preenchido com a informação que será exibida aos usuários e o campo "Value" com a informação que será gravada no banco de dados.

Caso o usuário queira que os *options* do campo sejam preenchidos com dados de uma tabela, deverá selecionar a opção "Preencher *options* com os dados de uma tabela", habilitando assim a segunda opção. Ao selecionar uma tabela, os campos desta serão exibidos em dois campos, no primeiro o usuário irá selecionar o campo que será gravado no banco de dados e no segundo campo os dados que deverão ser exibidos aos usuários.

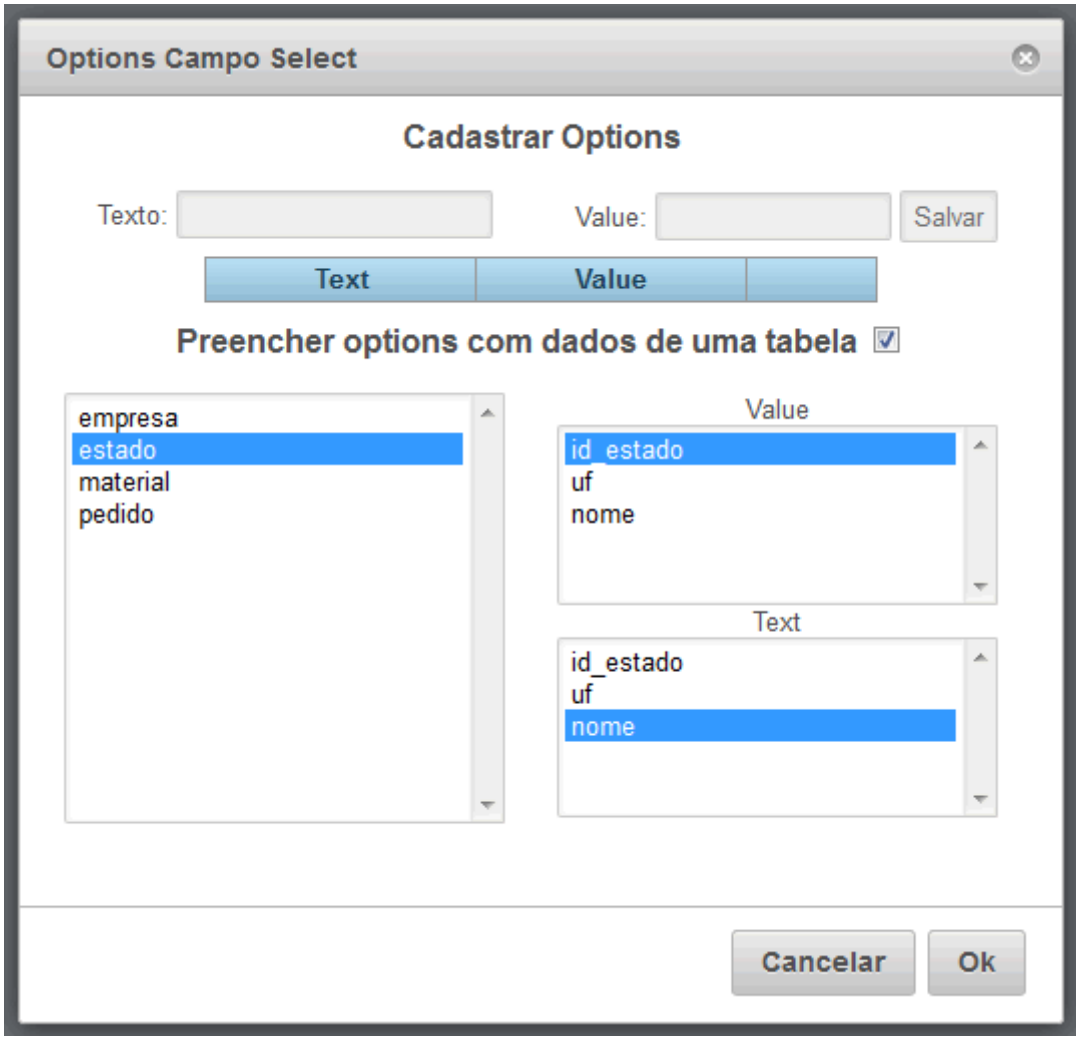

Figura 4.11: Options campo select.

Cada linha inserida na área do formulário fica posicionada ao final do formulário, portanto é possível mover a linha para cima ou para baixo, basta que o usuário clique sobre a linha para arrastá-la até a posição desejada. Para cada linha existe um menu auxiliar, conforme pode ser visto na figura 4.12 e descrito abaixo. Para exibir este menu, basta clicar com o botão direito do *mouse* sobre uma linha.

1. **Subir**: move a linha selecionada uma posição acima.
2. **Descrer**: move a linha selecionada uma posição para baixo.
3. **Duplicar**: cria uma cópia exata da linha selecionada.
4. **Excluir**: exclui a linha.

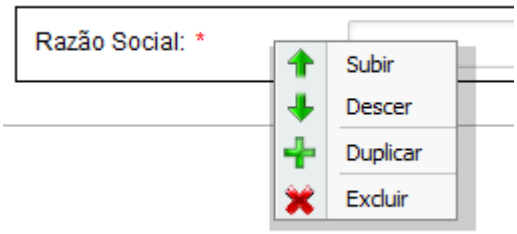

Figura 4.12: Menu contexto.

Existindo dúvidas sobre a estrutura das tabelas do banco de dados, o usuário poderá verificar e tirar suas dúvidas diretamente na ferramenta. Para isso poderá selecionar a opção do menu “Database” que irá exibir uma janela conforme a figura 4.13, contendo a estrutura completa de suas tabelas.

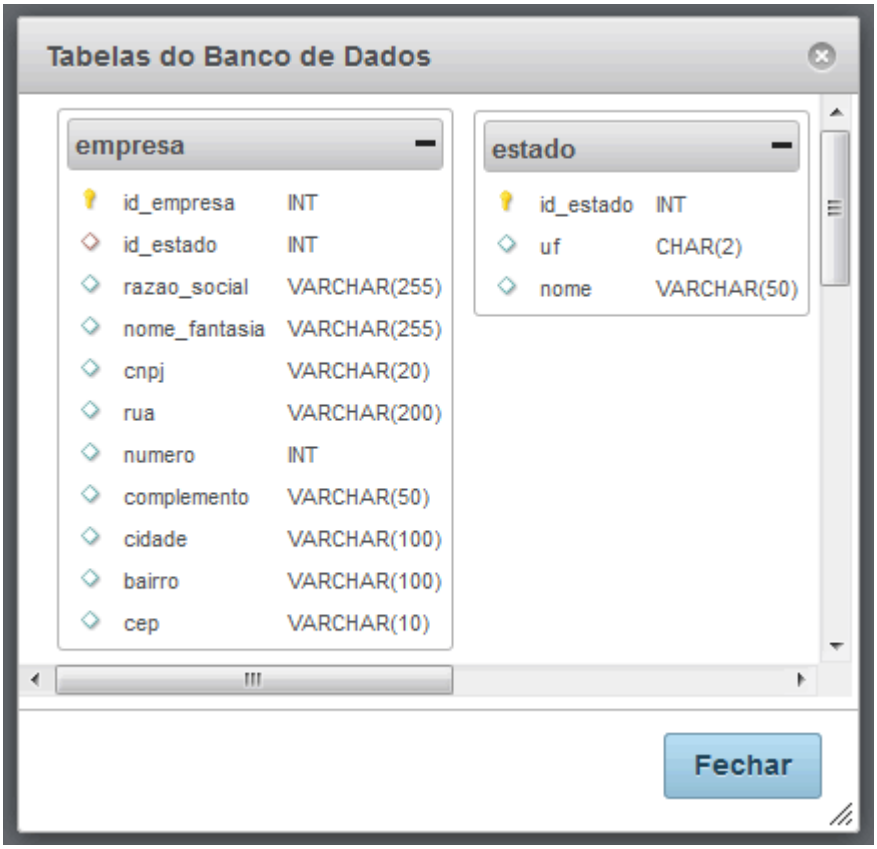

Figura 4.13: Atributos e tabelas do banco de dados.

A partir do momento que o formulário estiver pronto para ser gerado, o usuário poderá clicar sobre a opção do menu “Gerar Código”. A ferramenta então irá varrer campo a

campo, coletando as informações definidas e em seguida enviando tais informações para o gerador, a fim de que o mesmo possa gerar o projeto.

Conforme podemos verificar na figura 4.14 será exibido um menu contendo a estrutura de arquivo do projeto. Ao clicar sobre um arquivo o conteúdo de mesmo será exibido no espaço ao lado.

Caso o projeto esteja pronto o usuário poderá efetuar o *download* do mesmo para que possa utilizar em sua aplicação.

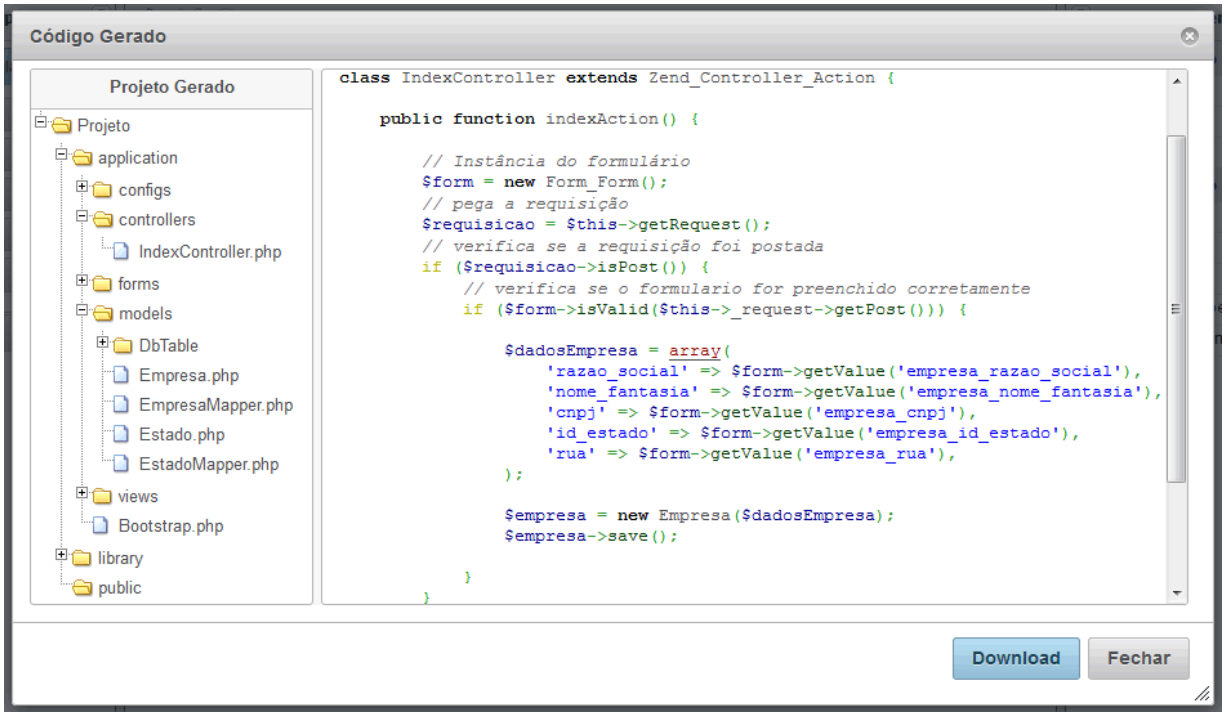

Figura 4.14: Tabelas do banco de dados.

4.4 Código gerado

O Zend Framework fornece uma estrutura padrão para organizar cada parte de uma aplicação. Baseando-se nesta convenção, o gerador Númenor gera código para cada camada da aplicação, conforme pode ser observado através da figura 4.15 que demonstra a hierarquia de pastas definida para a aplicação.

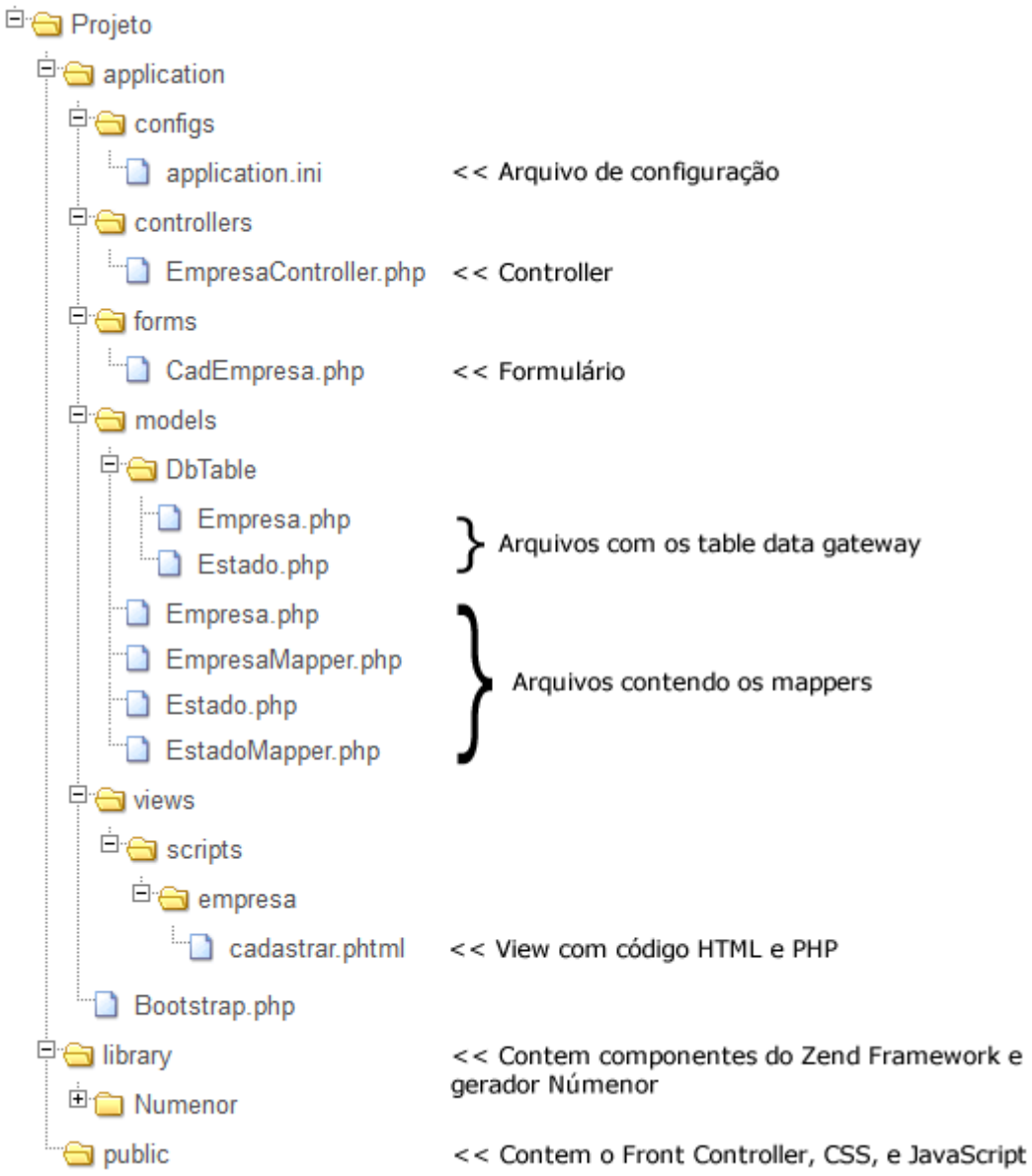

Figura 4.15: Estrutura de diretórios.

Essa estrutura de pastas exibida através da figura 4.15, é resultado da geração de código do Númenor para o formulário da figura 4.16. Os trechos de código exibidos a seguir foram gerados através da ferramenta para este formulário.

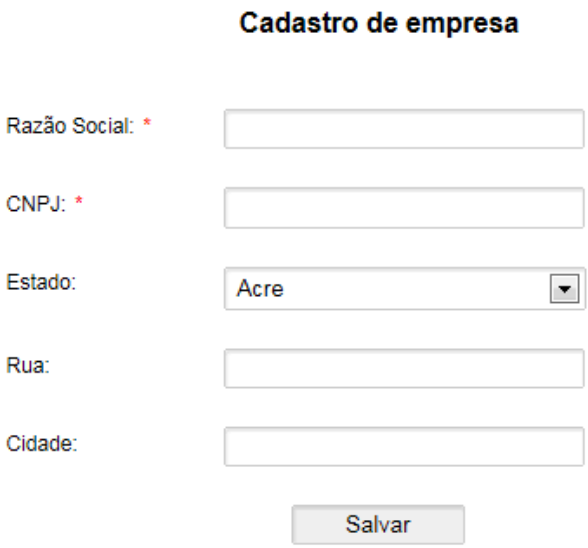

Figura 4.16: Formulário.

O arquivo “application.ini” é o arquivo de configuração, utilizado para definir alguns parâmetros de componentes, assim como os dados de acesso a banco de dados. Conforme pode ser visualizado no quadro 4.1, o arquivo é baseado na sintaxe INI utilizada para estruturação de arquivos de configuração no geral.

```
includePaths.models = APPLICATION_PATH "/models"
appnamespace = ""

resources.db.adapter = pdo_mysql
resources.db.params.host = localhost
resources.db.params.username = root
resources.db.params.password =
resources.db.params.dbname = base_modelo
resources.db.isDefaultTableAdapter = true
```

Quadro 4.1: Arquivo “application.ini”.

A camada de mais alto nível é composta pelo *Controller* e pela *View*, e possui código muito mais simplificado, nele apenas é feita a chamada ao formulário, verificando se o mesmo foi submetido ou não.

No quadro 4.2, é exibido o código do *Controller*, onde ele é baseado em *Action* para determinar qual operação está sendo requisitada pelo usuário.

```php
<?php

class EmpresaController extends Zend_Controller_Action {

    public function cadastrarAction() {

        // Instância do formulário
        $form = new Form_CadEmpresa();
        // pega a requisição
        $requisicao = $this->getRequest();
        // verifica se a requisição foi postada
        if ($requisicao->isPost()) {
            // verifica se o formulario for preenchido corretamente
            if ($form->isValid($this->_request->getPost())) {

                $dadosEmpresa = array(
                    'razao_social' => $form-
>getValue('empresa_razao_social'),
                    'cnpj' => $form->getValue('empresa_cnpj'),
                    'id_estado' => $form->getValue('empresa_id_estado'),
                    'rua' => $form->getValue('empresa_rua'),
                    'cidade' => $form->getValue('empresa_cidade'),
                );

                $empresa = new Empresa($dadosEmpresa);
                $empresa->save();

            }
        }

        $this->view->form = $form;
    }

}
```

Quadro 4.2: Arquivo “EmpresaController.php”.

O componente Zend_Form abstraiu a parte de criação de formulários e de validação. Com o Zend_Form basta criar uma classe que herda deste componente e definir os elementos do formulário, junto com as validações para cada elemento. O código do formulário para cadastrar uma nova empresa é apresentado no quadro 4.3.

Foram retirados deste trecho de código os elementos para os campos: Razão Social, Rua e Cidade, para que o código não fique muito extenso nesta demonstração. Foram deixados dos campos CNPJ e Estado devido aos mesmo terem algumas particularidades como por exemplo:

- CNPJ: faz a utilização dos tipos de validações cadastradas pelo usuário para validar uma determinada informação utilizando expressão regular.

- Estado: para o preenchimento das *options* deste campo *select*, é feito uma consulta na tabela estado, retornando como chave o id_estado e o valor a ser exibido ao usuário o nome do estado. Este método é criado automaticamente a partir do momento em que é definida nas propriedades do campo *select*, que as *options* devem ser preenchidas através das opções existentes em uma tabela do banco de dados.

```
<?php

/**
 * Classe Zend_Form para o formulário CadEmpresa
 *
 * Versão: 1.5
 * Criado Por: Númenor
 * Data Criação: 01/06/2011 10:50:17
 * Modificado Por:
 * Data Modificação:
 */
class Form_CadEmpresa extends Zend_Form {

    public function __construct($option = null) {
        parent::__construct($option);

        // Define o camino do decorator que irá utilizar em todos os
elementos.
        $this->addElementPrefixPath('Numenor_Form_Decorator',
'Numenor/Form/Decorator/', 'decorator');


        $empresaCnpj = new Zend_Form_Element_Text('empresa_cnpj');
        $empresaCnpj->setLabel('CNPJ:*')
            ->setRequired(true)
            ->setValidators(array('NotEmpty'))
            ->addValidator('regex', false,
array('/^\d{2}\.\d{3}\.\d{3}\/\d{4}\-\d{2}$/'))
            ->setOptions(array(
                'attr_label' => array(
                    'class' => 'width110 Left'
                ),
                'attr_campo' => array(
                    'class' => 'width200',
                    'valida' => 'cnpj'
                )
            ));

        // Preenche os options do select com os campos id_estado e nome.
        $estado = new Estado();
        $dadosOption = $estado->getSelectIdEstadoNome();

        $empresaIdEstado = new
Zend_Form_Element_Select('empresa_id_estado');
        $empresaIdEstado->setLabel('Estado:*')
            ->setRequired(true)
            ->setValidators(array('NotEmpty'))
```

```
            ->addMultiOptions($dadosOption)
            ->setOptions(array(
                'attr_label' => array(
                    'class' => 'width110 Left'
                ),
                'attr_campo' => array(
                    'class' => 'width201'
                )
            ));


        $submit = new Zend_Form_Element_Submit('submit');
        $submit->setLabel('Salvar')
            ->setOptions(array(
                'attr_linha_form' => array(
                    'class' => 'Center'
                ),
                'attr_campo' => array(
                    'class' => 'width92'
                )
            ));

        // Inclui os campos ao formulário.
        $this->addElements(array(
            $empresaRazaoSocial,
            $empresaCnpj,
            $empresaIdEstado,
            $empresaRua,
            $empresaCidade,
            $submit,
        ));

        // Define a existencia de uma div envolvendo as linhas do
formulário.
        $this->setDecorators(array(
            'FormElements',
            array(
                array(
                    'data' => 'HtmlTag'
                ),
                array(
                    'tag' => 'div',
                    'class' => 'formulario'
                )
            ),
            'Form'
        ));

        // Sobrescreve decorators existentes com os customizados.
        $this->setElementDecorators(array('Composite'));
    }
}
```

Quadro 4.3: Arquivo “CadEmpresa.php”.

O quadro 4.4 é possível perceber a abstração de todo código pertinente ao acesso a banco. A classe herda de Zend_Db_Table, que é uma interface para tabelas de bancos de dados. Ela fornece métodos para muitas operações comuns sobre tabelas. A classe base é extensível, assim pode ser adicionada a lógica customizada.

```php
<?php

/**
 * Classe para a tabela estado
 *
 * Versão: 1.5
 * Criado Por: Númenor
 * Data Criação: 01/06/2011 10:50:19
 * Modificado Por:
 * Data Modificação:
 */
class DbTable_Estado extends Zend_Db_Table_Abstract {

    protected $_name = 'estado';
    protected $_primary = array(
        'id_estado',
    );

}
```

Quadro 4.4: Arquivo “DbTable/Estado.php”

O *Model* possui apenas os atributos que definem uma entidade do banco e os métodos auxiliares a estes atributos. O Quadro 4.5 demonstra um *Model* gerado pelo Númenor.

```php
<?php

/**
 * Classe para a tabela estado
 *
 * Contém a mesma estrutura da tabela, os campos da tabela com os seus
Get's e Set's
 *
 * Versão: 1.5
 * Criado Por: Númenor
 * Data Criação: 01/06/2011 10:50:19
 * Modificado Por:
 * Data Modificação:
 */
class Estado extends Numenor_Db_DomainObjectAbstract {

    // Define as propriedades
    protected $_mapper = "EstadoMapper";
    protected $_primary = array('id_estado');
```

```
    private $idEstado;
    private $uf;
    private $nome;

    /**
     * Seta um valor à propriedade $idEstado
     *
     * @access public
     * @param int $idEstado
     */
    public function setIdEstado($idEstado) {
        $this->idEstado = $idEstado;
    }

    /**
     * Retorna o valor da propriedade $idEstado
     *
     * @access public
     * @return int
     */
    public function getIdEstado() {
        return $this->idEstado;
    }

    /**
     * Seta um valor à propriedade $uf
     *
     * @access public
     * @param char $uf
     */
    public function setUf($uf) {
        $this->uf = $uf;
    }

    /**
     * Retorna o valor da propriedade $uf
     *
     * @access public
     * @return char
     */
    public function getUf() {
        return $this->uf;
    }

    /**
     * Seta um valor à propriedade $nome
     *
     * @access public
     * @param varchar $nome
     */
    public function setNome($nome) {
        $this->nome = $nome;
    }

    /**
     * Retorna o valor da propriedade $nome
     *
     * @access public
     * @return varchar
     */
    public function getNome() {
```

```
        return $this->nome;
    }

    /**
     * Método responsável em buscar as informações para o preenchimento
     * de um campo Select
     *
     * @return arry Retorna dados em um matriz de pares chave-valor.
     */
    public function getSelectIdEstadoNome() {
        return $this->getMapper()->getSelectIdEstadoNome();
    }

}
```

Quadro 4.5: Arquivo “Estado.php”

O *Data Mapper* mapeará dados vindos do *Table Data Gateway* que se comunica diretamente com o banco de dados. O *Table Data Gateway* herda de um componente do Zend Framework, o Zend_Db_Table, todos os métodos referentes a persistência, abstraindo ainda mais a SQL do desenvolvedor.

```
<?php

/**
 * Classe Mapper para a tabela estado
 *
 * Versão: 1.5
 * Criado Por: Númenor
 * Data Criação: 01/06/2011 10:50:19
 * Modificado Por:
 * Data Modificação:
 */
class EstadoMapper extends Numenor_Db_DataMapperAbstract {

    // Define as propriedades
    protected $_dbTable = "DbTable_Estado";
    protected $_model = "Estado";


    protected function _insert(Numenor_Db_DomainObjectAbstract $obj) {
        try {
            $dbTable = $this->getDbTable();
            $data = array(
                'uf' => $obj->getUf(),
                'nome' => $obj->getNome(),
            );
            $dbTable->insert($data);
            return true;
        } catch (Zend_Exception $e) {
            return false;
        }
    }
```

```
    protected function _update(Numenor_Db_DomainObjectAbstract $obj) {
        try {
            $dbTable = $this->getDbTable();

            $data = array(
                'uf' => $obj->getUf(),
                'nome' => $obj->getNome(),
            );

            $where = array();
            $where['id_estado = ?'] = $obj->getIdEstado();

            $dbTable->update($data, $where);
            return true;
        } catch (Zend_Exception $e) {
            return false;
        }
    }

    /**
     * Método responsável em buscar as informações para o preenchimento
     * de um campo Select
     *
     * @return arry Retorna dados em um matriz de pares chave-valor.
     */
    public function getSelectIdEstadoNome() {
        $db = $this->getDb();
        $query = $db->select();
        $query->from('estado', array('id_estado', 'nome'))
                ->order('nome asc');
        return $db->fetchPairs($query);
    }

}
```

Quadro 4.6: Arquivo “EstadoMapper.php”

O Quadro 4.7 representa a *View*, neste arquivo contem apenas código HTML e alguns trechos de código PHP.

```
<h1 class="Center">Cadastro de empresa</h1>

<?php echo $this->form; ?>
```

Quadro 4.7: Arquivo “cadastrar.phtml”

O quadro 4.8 demostra o código gerado para o *bootstrap*, que é um dos arquivos mais importantes, pois toda requisição passa por ele, ou seja toda vez que um usuário vai

executar uma funcionalidade que chame algum arquivo, vai antes passar pelo *bootstrap* para depois ser liberado.

```
<?php

class Bootstrap extends Zend_Application_Bootstrap_Bootstrap {

    protected function _initAutoLoader() {

        $autoloader = Zend_Loader_Autoloader::getInstance();
        $autoloader->setFallbackAutoloader(true); // Pega tudo
    }

    protected function _initTranslate() {

        $translator = new Zend_Translate(array(
                    'adapter' => 'array',
                    'content' => '../library/Numenor/Form/Translate',
                    'locale' => 'pt_BR',
                    'scan' => Zend_Translate::LOCALE_DIRECTORY)
        );
        Zend_Validate_Abstract::setDefaultTranslator($translator);
    }
}
```

Quadro 4.8: Arquivo “Bootstrap.php”

4.5 Avaliação do Númenor

Para fazer a avaliação do gerador foi disponibilizado um demonstrativo da ferramenta através do *link* <http://demo.numenor.com.br> para o grupo de discussão Zend Framework Brasil, juntamente com um tutorial disponibilizado através de vídeo no YouTube <http://www.youtube.com/watch?v=K-DQWHs3h8M>.

Após liberação do demonstrativo e do tutorial, foi desenvolvido um questionário conforme anexo 1 com o objetivo de obter a avaliação dos usuários sobre usabilidade, produtividade e qualidade do código gerado. Para isso o questionário foi enviado primeiramente apenas para 15 usuários, com o objetivo de efetuar um pré-teste, verificando se o questionário estava objetivo e claro, validando ainda se as respostas fornecidas pelos usuários atenderiam à necessidade de validação da ferramenta. Como critério foram selecionados os usuários mais participativos da comunidade e também usuários que mostraram interesse pela ferramenta, buscando mais informações da mesma através de *e-mail* e *chat*.

Na primeira questão foi verificado o tempo de experiência com desenvolvimento PHP, o resultado obtido foi satisfatório devido às pessoas que responderam terem acima de um ano de experiência, que é um indicativo de possuir uma boa experiência.

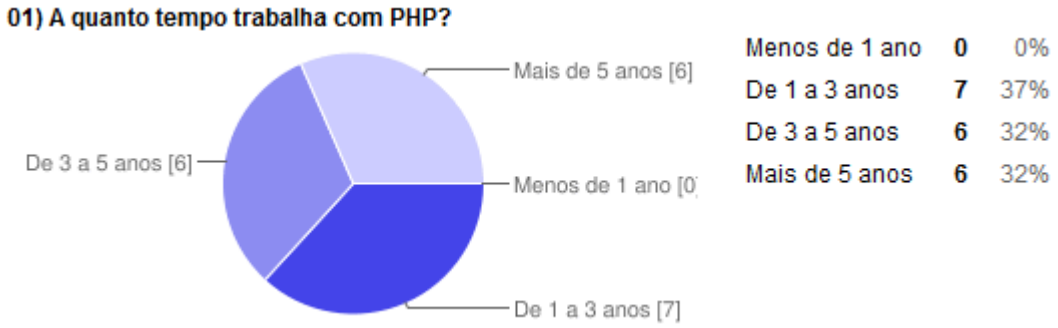

Figura 4.17: Resposta da questão 1.

A figura 4.18 apresenta o resultado da questão dois, onde foi questionado o tempo de experiência utilizando Zend Framework. Como resultado obtivemos respostas de usuários com pouco tempo de experiência utilizando o Zend Framework a grande maioria com menos de um ano.

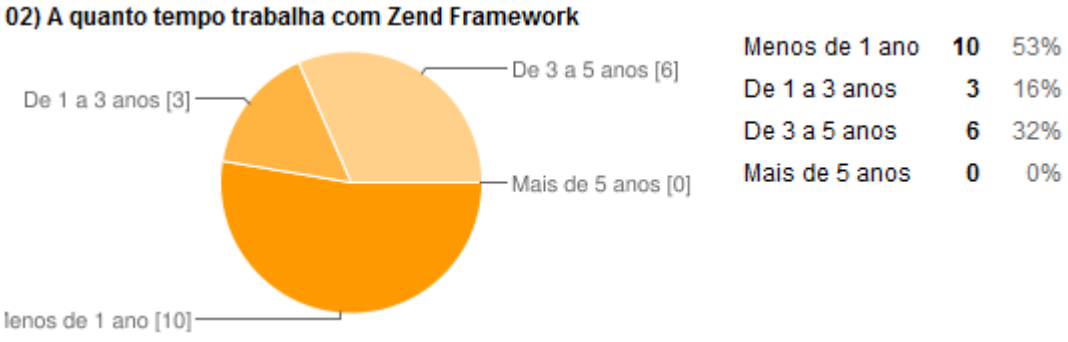

Figura 4.18: Resposta da questão 2.

Para a questão três foi solicitado aos usuários um tempo médio que os mesmos levariam para desenvolver um formulário conforme a figura 4.19, sem o auxílio de um gerador de código.

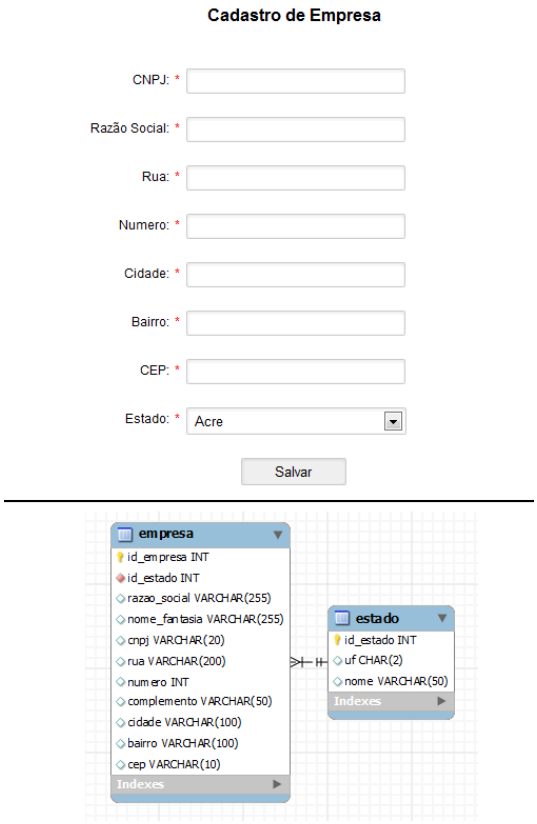

Figura 4.19: Formulário de exemplo.

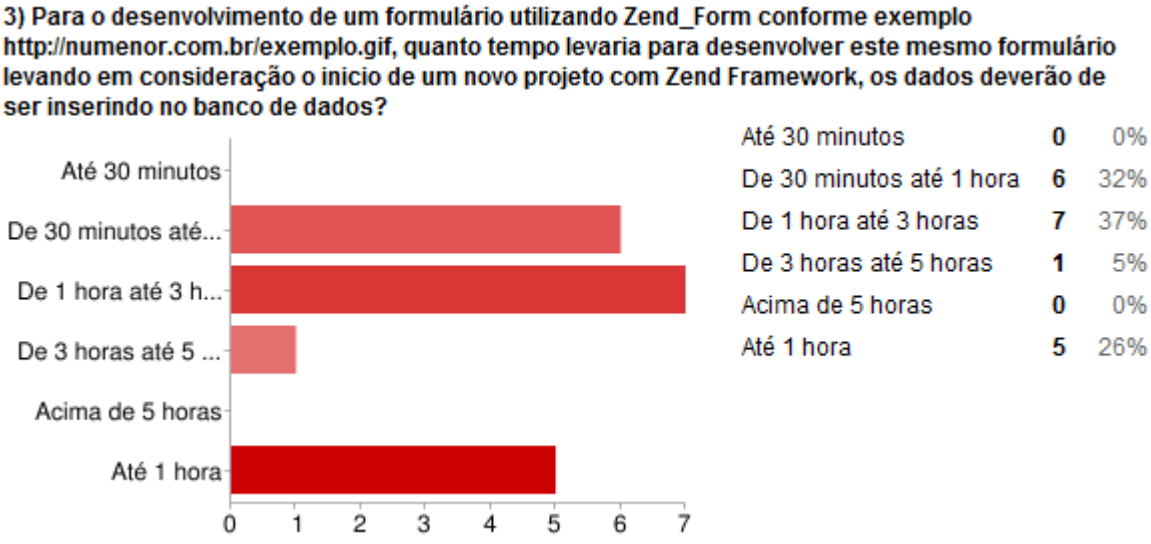

Figura 4.20: Resposta da questão 3.

A questão quatro avalia o tempo gasto para desenvolver o mesmo formulário da figura 4.21, só que utilizando como ferramenta de apoio o gerador Númenor, o objetivo entre

as questões três e quatro é avaliar quanto tempo médio necessário para desenvolver o formulário com e sem apoio da ferramenta Númenor.

Ao efetuar o pré-teste foi identificado um problema nas questões três e quatro, devido ao formulário utilizado como exemplo ser de baixa complexidade, o tempo médio ficaria em 1 hora. E como a grande maioria dos usuários respondeu “até 1 hora” para as duas questões, não foi possível encontrar uma diferenciação entre desenvolver o formulário com ou sem apoio do Númenor. Para isso foi modificada as possíveis respostas incluindo as novas opções de respostas “Até 30 minutos” e “De 30 minutos até 1 hora”.

Com essa mudança foram obtidos os resultados que desenvolver um formulário com o auxilio do Númenor acelera o processo de desenvolvimento conforme podemos observar nas questões três e quatro. Sem o auxilio do Númenor apenas 11 usuários conseguiriam desenvolver este formulário em até uma hora. Com o Númenor todos os usuários fariam este mesmo formulário em até uma hora, cerca de 42,11% das pessoas que antes fariam este formulário acima de uma hora, passam a fazer em no máximo uma hora.

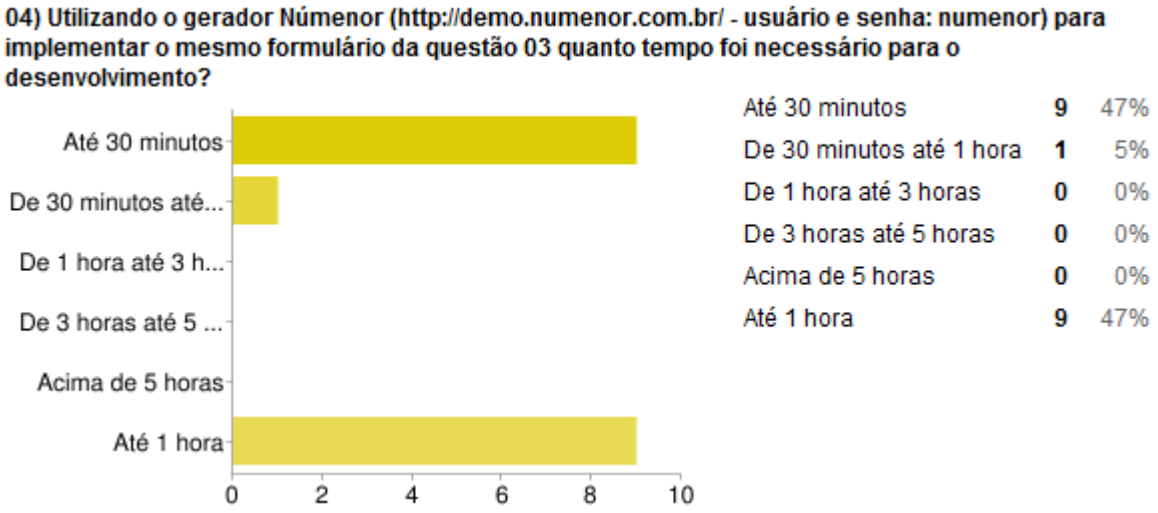

Figura 4.21: Resposta da questão 4.

Para a questão de número cinco foi verificada a experiência do usuário com o gerador, em relação à interface do Númenor. Como resultado obtido, todos os usuários responderam que a interface do Númenor está boa ou muito boa, conforme podemos observar na figura 4.22.

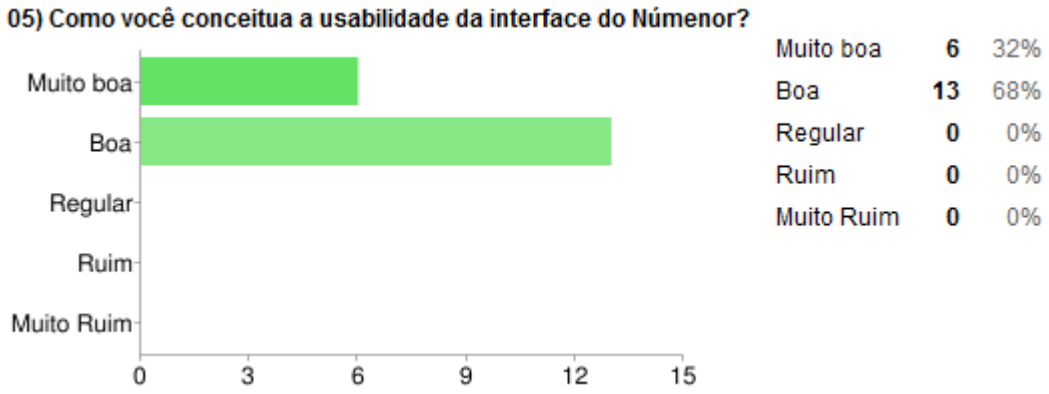

Figura 4.22: Resposta da questão 5.

A questão seis buscou identificar qual a percepção do usuário sobre a facilidade de estender o código gerado. O resultado obtido foi positivo, pois cerca de 79% dos usuários acharam que o código gerado pode ser facilmente estendido. O relato de um dos usuários mostrou exatamente aquilo que a ferramenta se propõe a fazer: “O númenor 'não se mete onde não é chamado'.”. Ou seja, ele gera código repetitivo, deixando assim tempo disponível para os programadores se preocuparem apenas com as regras de negócio.

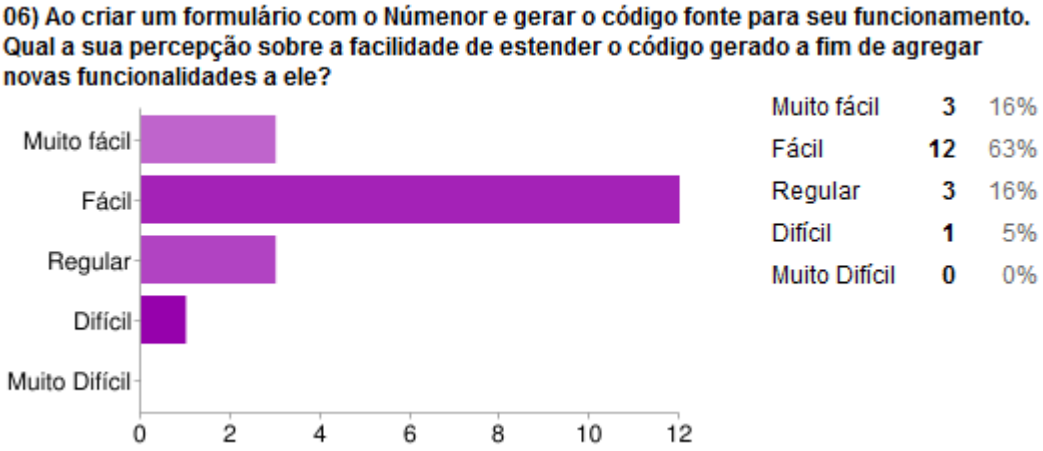

Figura 4.23: Resposta da questão 6.

A questão sete buscou saber se os SGBD’s atenderiam às necessidades dos desenvolvedores. A grande maioria respondeu que sim, e somente três pessoas responderam que não. Ou seja, a maioria dos projetos em PHP são feitos sobre SGBDs MySql e PostgreSql e iniciar a ferramenta dando suporte para estes SGBDs foi uma escolha correta.

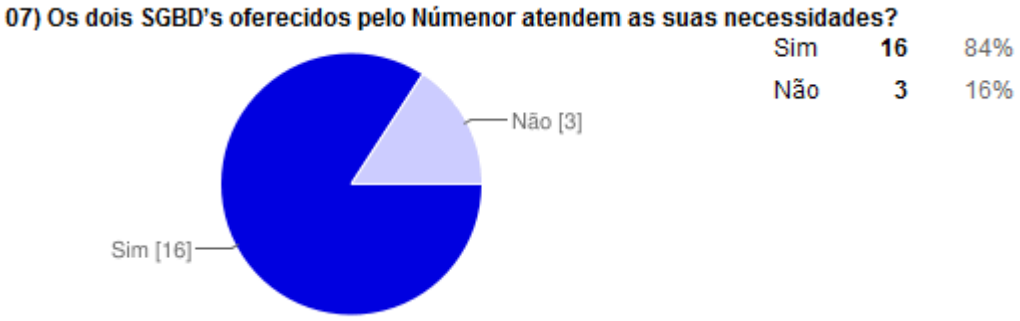

Figura 4.24: Resposta da questão 7.

A questão oito buscou saber quais SGBD's seriam necessários caso os dois fornecidos não atendessem.

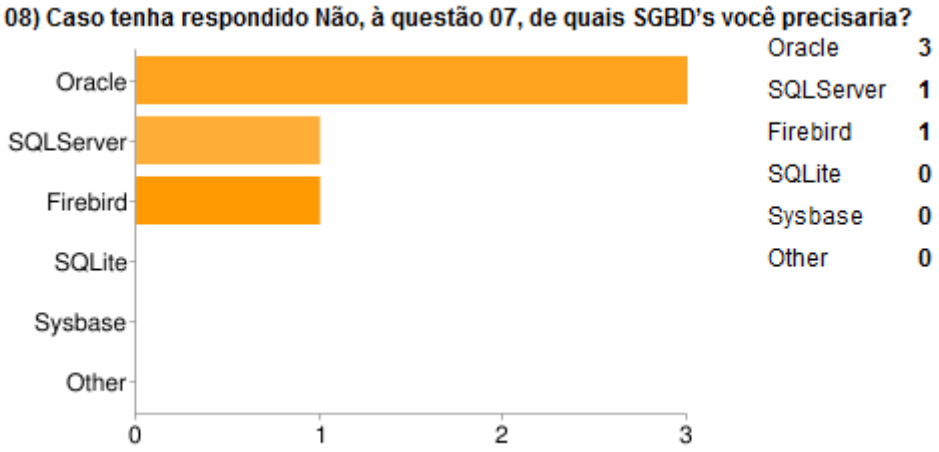

Figura 4.25: Resposta da questão 8.

A questão nove buscou saber se os usuários utilizariam o gerador Númenor como uma ferramenta auxiliar de desenvolvimento.

Figura 4.26: Resposta da questão 9.

E a décima questão tratou de verificar o motivo pelo qual o usuário respondeu sim ou não à questão nove. Pode ser visualizado as principais respostas positivas através do quadro 4.9.

• A facilidade que a ferramenta me proporcionaria em: rapidez, evitar erros humanos de programação, padrão de código.
• Facilidade de uso e ganho em produtividade.
• Apesar da Zend disponibilizar algo para facilitar o desenvolvimento dos formulários em questão, ainda é um trabalho demorado, o que torna o Númenor uma ferramenta útil, pois facilita e agiliza o trabalho de desenvolvimento do formulário.
• Facilidade de uso. Bom código gerado. O Númenor 'não se mete onde não é chamado'.
• Possibilidade de utilizar diretamente pelo navegador. Cria os mapeamentos dos objetos da tabela. Gera somente a estrutura inicial do projeto, o que facilita a integração com projetos existentes.

Quadro 4.9: Respostas positivas.

Como respostas negativas podem ser observadas através do quadro 4.10.

- Ainda não porque faltam alguns elementos que já utilizo na criação dos forms com Zend, como por exemplo, selects com ajax que trazem valores do banco de dados. No entanto, para formulários mais simples, é uma ótima ferramenta que tende a tomar espaço no mercado muito em breve, visto sua qualidade e simplicidade.

- Algo particular. Não gosto muito de geradores de códigos, mas gostei bastante do Númenor. Parabéns.

- Talvez usaria, mas precisaria de uma boa documentação.

Quadro 4.9: Respostas negativas.

Com este questionário pode-se perceber que o gerador Númenor teve uma boa recepção pelos usuários, principalmente usuários com pouca experiência com Zend Framework, pois o mesmo facilita bastante o processo de desenvolvimento, gerando apenas o código necessário para o funcionamento das principais operações tais como: *insert, update, delete* e *select*. É fácil de ser estendido e o código já vem todo comentado.

CONCLUSÃO

De acordo com os estudos realizados percebeu-se que a utilização de um gerador de código pode oferecer diversos benefícios e entre eles podemos citar: agilidade no processo de desenvolvimento, padrão de código e maior produtividade. Com base nessas informações foi desenvolvido o gerador Númenor buscando atender esses itens.

Durante o desenvolvimento da ferramenta foram identificados recursos e funcionalidades que poderiam ser implementadas, mas por estarem fora do escopo inicial do projeto ou por falta de tempo, não foram implementadas durante o desenvolvimento desta proposta. No entanto, estes recursos e funcionalidades são elencados a seguir com intuito de orientar futuros trabalhos que tenham como objetivo melhorar e ampliar a ferramenta desenvolvida.

Entre os recursos desejáveis à ferramenta destaca-se:

- Suporte a outros sistemas gerenciadores de banco de dados, como MsSQL, Oracle, SQLite, entre outros. O suporte a outros SGBDs trariam à ferramenta a possibilidade de ser utilizada em outros ambientes de desenvolvimento.
- Implementação de novos elementos de HTML e novas funcionalidades, isso ajudaria a criar formulários mais complexos, facilitando ainda mais o processo de desenvolvimento.
- Implementar internacionalização ao gerador. Com isso a ferramenta ganharia maior espaço no mercado podendo receber maior contribuição de pessoas de outros países.
- Implementar um gerador de formulário automático baseado na estrutura das tabelas.

- Efetuar testes com usuários, cronometrando tempo de desenvolvimento, utilizar alguma métrica de engenharia de software para essa avaliação.

O gerador de código Númenor é um *software open source* onde qualquer pessoa poderá contribuir com o projeto, seja através de testes ou implementação de novas funcionalidades. O projeto encontra-se hospedado no site <http://github.com>. o GitHub é um *website* que hospeda códigos fontes através de repositórios *git*, onde todos os projetos e códigos são compartilhados com todo mundo ou seja, é uma rede social para compartilhamento de código, onde qualquer pessoa poderá acompanhar o crescimento dessa ferramenta.

Conforme pode ser observado através da avaliação junto a programadores, o Númenor atendeu todas as expectativas, possui uma interface de fácil utilização, que facilita e agiliza o processo de criação e manipulação de formulários, o código gerado é de fácil entendimento e de ser estendido.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLEN, Rob; LO, Nick; BROWN, Steven. **Zend em Ação**. Rio de Janeiro: Alta Book. 2009. 360p.

BECKER, Vitor Hugo. **Um Estudo sobre a Ferramenta EGEN Developer**, Porto Alegre: UFRGS, 9 p.; [200-].

BELEM, Thiago Dutra da Fonseca. Frameworks no PHP: O que, quando, por que e qual? Disponível em: <http://imasters.com.br/artigo/13718/php/frameworks_no_php_o_que_quando_por_que_e_qual/>. Acesso em: 20 Nov. 2010.

CAKEPHP. **Manual do Framework CakePHP**. Disponível em: <http://book.cakephp.org/>. Acesso em: 20 Nov. 2010.

CODEIGNITER. **Manual do Framework CodeIgniter**. Disponível em: <http://www.codeigniter.com.br/manual/>. Acesso em: 20 Nov. 2010.

DALL'OGLIO, Pablo. PHP Programando com Orientação a Objetos Inclui Design Patterns. 2ª ed. São Paulo: Novatec. 2009. 574p.

DENNIS, Alan; WIXOM, **Barbara Haley. Análise e Projeto de Sistemas**. Rio de Janeiro: LTC. 2003. 461p.

EGEN. **Manual do e-Gen Developer**. Disponível em: <http://www.egen.com.br/>. Acesso em: 20 Nov. 2010.

FRANCA, Luiz Paulo Alves. **Um processo Para Construção de Geradores de Artefatos**. Tese de Doutorado; Pontífica Universidade Católica do Rio de Janeiro; 2000.

FREIRE, Flávia. **CakePHP Receita simples para quem quer agilidade**. TI Digital. Rio de Janeiro, n° 01, Ano 01, p. 26-36, Abril 2009.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. **Padrões de projeto: soluções reutilizáveis de software orientado a objetos**. Porto Alegre: Bookman, 2000. 364 p.

HERRINGTON, Jack. **Code generation in action**. Greenwich, CT: Manning, 2003.

HOSTWEB. **MySQL ou PostgreSQL, qual usar?**. Disponível em: <http://blog.hostweb.com.br/mysql-ou-postgresql-qual-usar/hostweb>. Acesso em: 5 Set. 2010.

JARGAS, Aurélio Marinho. **Expressões regulares uma abordagem divertida**. 3. Ed. São Paulo: Novatec. 2009. 207 p.

LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos.** 3. ed. Porto Alegre, RS: Bookman, 2007. 695 p.

LISBOA, Flávio Gomes da Silva. **Zend Framework Desenvolvimento em PHP 5 orientado a objetos com MVC**. São Paulo: Novatec. 2008. 184 p.

LISBOA, Flávio Gomes da Silva. **Zend Framework Componentes Poderosos para PHP**. São Paulo: Novatec. 2009. 352 p.

LISBOA, Flávio Gomes da Silva. **Criando Aplicações com Zend e Dojo**. São Paulo: Novatec. 2010. 192 p.

MACORATTI, José Carlos. **Padrões de Projeto: O modelo MVC - Model View Controller**. Disponível em <http://www.macoratti.net/vbn_mvc.htm> Acesso em Nov. 2010.

MCCONNELL, Steve. **Code complete: um guia prático para a construção de software.** 2. ed. Porto Alegre: Bookman, 2005, 928 p.

MINETTO, Elton Luís. **Frameworks para Desenvolvimento em PHP**. São Paulo: Novatec. 2007. 192p.

ODISI, Francis Benito. **Geração de Código Para Acesso a Dados Utilizando os Padrões MVC e DAO**. Trabalho de Conclusão Universidade do Vale do Itajaí. 2008.

OLIVEIRA, Alexandre de. **Tecnologias para desenvolvimento de sistemas web com ênfase na utilização de software livre:** estudo de caso. Monografia (Conclusão do Curso de Ciência da Computação) - Universidade Feevale, 141 f. 2005.

PAULA FILHO, Wilson de Pádua. **Engenharia de software: Fundamentos, Métodos e Padrões**. 2. ed. Rio de Janeiro: LTC. 2003. 602 p.

PRESSMAN, Roger S. **Engenharia de software**. 6. ed. Porto Alegre, RS: AMGH, 2010. 720 p.

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo. Makron Books, 1995. 1056 p.

PHP. **Manual do PHP**. 2010. Disponível em: <http://www.php.net/docs.php>. Acesso em: 20 Nov. 2010.

SAUVÉ, Jacques. **Vantagens e desvantagens no uso de frameworks**. Disponível em <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/porque.htm> Acesso em Nov.

2010.

SCHMITZ, Daniel; STEPHANOU, Lucas; TELLES, Marcos; JUNIOR, Oberaldo Büll; WILHELM, Ivan. **Frameworks PHP: compare os cinco mais famosos e agilize seus projetos.** TI Digital. Rio de Janeiro, n° 15, Ano 02, p. 26-36, Maio 2010.

SCHMITZ, Daniel Pace. **Dominando Flex e Zend**. São Paulo. Canal6, 2009. 263 p.

SHALLOWAY, Alan; TROTT, James. **Explicando Padrões de Projeto**. Porto Alegre: Bookman, 2004, 328 p.

SILVA, Alberto; VIDEIRA, Carlos. **UML: Metodologias e Ferramentas CASE**. Portugal: Centro Atlântico, 2001. 384 p.

SILVA, Maurício Samy. **Jquery A Biblioteca do Programador JavaScript**. São Paulo: Novatec. 2009. 430p.

YII. **Manual do Framework Yii**. Disponível em: <http://www.yiiframework.com/doc/>. Acesso em: 20 Nov. 2010.

ZEND. **Manual do ZEND Framework**. Disponível em: <http://www.zendframework.com/manual/>. Acesso em: 20 Nov. 2010.

ANEXO 1 – QUESTIONÁRIO DE AVALIAÇÃO DO GERADOR NÚMENOR

Anexo 1:

1) A quanto tempo trabalha com PHP?

2) A quanto tempo trabalha com Zend Framework

3) Para o desenvolvimento de um formulário utilizando Zend_Form conforme exemplo http://numenor.com.br/exemplo.gif, quanto tempo levaria para desenvolver este mesmo formulário levando em consideração o inicio de um novo projeto com Zend Framework, os dados deverão de ser inserindo no banco de dados?

4) Utilizando o gerador Númenor (http://demo.numenor.com.br/ - usuário e senha: numenor) para implementar o mesmo formulário da questão 03 quanto tempo foi necessário para o desenvolvimento?

5) Como você conceitua a usabilidade da interface do Númenor?

6) Ao criar um formulário com o Númenor e gerar o código fonte para seu funcionamento. Qual a sua percepção sobre a facilidade de estender o código gerado a fim de agregar novas funcionalidades a ele?

7) Os dois SGBD’s oferecidos pelo Númenor atendem as suas necessidades?

8) Caso tenha respondido Não, à questão 07, de quais SGBD’s você precisaria?

9) Você usaria o Númenor como ferramenta auxiliar de desenvolvimento de software?

10) Quais os motivos que lhe levaram a responder Sim ou Não na questão 09?"

Anexo 2:

Indicação de data e hora	1	2	3	4	5	6	7	8	9	10
23/5/11 19:31	Mais de 5 anos	Menos de 1 ano	De 1 hora até 3 horas	Até 1 hora	Boa	Fácil	Sim		Sim	complexo
23/5/11 23:53	De 1 a 3 anos	Menos de 1 ano	De 1 hora até 3 horas	Até 1 hora	Muito boa	Regular	Sim		Sim	Facilidade e uma boa implementacao do codigo
27/5/11 8:30	De 1 a 3 anos	De 1 a 3 anos	Até 1 hora	Até 1 hora	Boa	Fácil	Sim		Sim	A facilidade que a ferramenta me proporcionaria em: rapidez, evitar erros humanos de programação, padrão de código, enfim, seria de grande utilidade
27/5/11 8:48	De 3 a 5 anos	De 1 a 3 anos	Até 1 hora	Até 1 hora	Muito boa	Fácil	Sim		Não	Ainda não porque faltam alguns elementos que já utilizo na criação dos forms com zend, como por exemplo, selects com ajax que trazem valores do banco de dados. No entanto, para formulários mais simples, é uma ótima ferramenta que tende a tomar espaço no mercado muito em breve, visto sua qualidade e simplicidade.
27/5/11 9:50	Mais de 5 anos	De 3 a 5 anos	Até 1 hora	Até 1 hora	Boa	Muito fácil	Sim		Não	Algo particular. Não gosto muito de geradores de códigos, mas gostei bastante do Númenor. Parabens.
27/5/11 13:39	De 3 a 5 anos	Menos de 1 ano	Até 1 hora	Até 1 hora	Muito boa	Fácil	Sim		Sim	Uma ferramenta prática como essa poderia muito bem sem incorporada em um projeto que tenho, agilizando mais ainda o desenvolvimento das minhas ferramentas.
27/5/11 13:54	Mais de 5 anos	Menos de 1 ano	De 1 hora até 3 horas	Até 1 hora	Boa	Regular	Sim		Sim	Facilidade de uso e ganho em produtividade.
27/5/11 14:08	Mais de 5 anos	De 3 a 5 anos	Até 1 hora	Até 1 hora	Boa	Fácil	Sim		Sim	Gostei da forma como são gerados os códigos
30/5/11 15:27	De 3 a 5 anos	De 3 a 5 anos	De 3 horas até 5 horas	Até 1 hora	Boa	Difícil	Não	Oracle	Sim	Se ele me agregasse algum valor ou agilidade no desenvolvimento.
8/6/11 9:07	De 3 a 5 anos	Menos de 1 ano	De 1 hora até 3 horas	Até 30 minutos	Muito boa	Muito fácil	Sim		Sim	O trabalho de desenvolvimento do formulário é o mais enjoado que existe(na minha opinião) e apesar da zend disponibilizar algo para facilitar o desenvolvimento dos formulários em questão, ainda é um trabalho demorado, o que torna o Númenor uma ferramenta útil, pois facilita e agiliza o trabalho de desenvolvimento do formulário.
8/6/11 9:19	De 1 a 3 anos	Menos de 1 ano	De 30 minutos até 1 hora	Até 30 minutos	Boa	Fácil	Sim		Sim	- Possibilidade de utilizar diretamente pelo navegador. - Cria os mapeamentos os objetos da tabela. - Gera somente a estrutura inicial do projeto, o que facilita a integração com projetos existentes. É o melhor gerador do mundo.

8/6/11 9:31	De 1 a 3 anos	Menos de 1 ano	De 30 minutos até 1 hora	Até 30 minutos	Boa	Fácil	Sim	Firebird	Sim	Eu acho o numenor bem útil em muitos casos, é uma boa ferramenta.
8/6/11 9:35	De 1 a 3 anos	Menos de 1 ano	De 1 hora até 3 horas	Até 30 minutos	Boa	Fácil	Sim		Sim	Facilidade de uso. Bom código gerado. O numenor 'não se mete onde não é chamado'.
8/6/11 9:58	Mais de 5 anos	De 3 a 5 anos	De 30 minutos até 1 hora	Até 30 minutos	Muito boa	Fácil	Não		Sim	muito facil e pratico
8/6/11 10:11	De 1 a 3 anos	Menos de 1 ano	De 30 minutos até 1 hora	Até 30 minutos	Muito boa	Fácil	Sim		Sim	Por agilizar o desenvolvimento.
8/6/11 10:18	De 1 a 3 anos	Menos de 1 ano	De 1 hora até 3 horas	Até 30 minutos	Boa	Fácil	Não	Oracle	Sim	ganho de agilidade no desenvolvimento
8/6/11 10:29	De 3 a 5 anos	De 3 a 5 anos	De 30 minutos até 1 hora	Até 30 minutos	Boa	Fácil	Sim		Sim	Velocidade na construção de forms completos.
9/6/11 9:54	Mais de 5 anos	De 1 a 3 anos	De 30 minutos até 1 hora	Até 30 minutos	Boa	Muito fácil	Sim		Sim	Usaria para formularios simples... formularios mais complexos... acho que precisa ser lapidado.
9/6/11 10:29	De 3 a 5 anos	De 3 a 5 anos	De 1 hora até 3 horas	De 30 minutos até 1 hora	Boa	Regular	Sim	Oracle, SQLServer	Não	Talvez usaria, mas precisaria de uma boa documentação.