GOVERNO DO ESTADO DO RIO DE JANEIRO
SECRETARIA DE ESTADO DE CIÊNCIA E TECNOLOGIA
FUNDAÇÃO DE APOIO À ESCOLA TÉCNICA
CENTRO DE EDUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO
FACULDADE DE EDUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO
FAETERJ/PETRÓPOLIS

Framework MVC - Lotus{PHP}

Guilherme Peixoto da Costa Louro

PETRÓPOLIS

Julho de 2015

Framework MVC - Lotus{PHP}

Guilherme Peixoto da Costa Louro

Trabalho apresentado no curso de Formação em Tecnologia da Informação e Comunicação da FAETERJ – Petrópolis como requisito parcial para obtenção do grau de tecnólogo.

Orientador: Matheus Bandini

Co-orientador: Lorem ipsum

PETRÓPOLIS

Julho de 2015

Monografia de Projeto Final de Graduação sob o título *"Framework MVC - Lotus{PHP} "*, defendida por Guilherme Peixoto da Costa Louro e aprovada em Julho de 2015, em Petrópolis, Estado do Rio de Janeiro, pela banca examinadora constituída pelos professores:

Orientador

Co-orientador

Nome do membro da banca
Intituição do Membro

Nome do membro da banca
Intituição do Membro

Resumo

Devido a forma complexa em que se encontra o desenvolvimento de sistemas web atualmente, torna-se cada vez mais importante o uso de ferramentas que facilitam a criação dos mesmos. Essas ferramentas que denominaremos de *Frameworks* são utilizadas visando o aumento de produtividade, esse aumento se deve as diversas ações que auxiliam nas principais atividades do desenvolvimento. Existe no mercado diversos *Frameworks* web que são opensource. Porém o foco desse trabalho será baseado na criação de um *Framework* próprio, onde foi escolhido um padrão de projeto, que será o modelo MVC, e uma linguagem de programação que será PHP. Serão apresentados cada passo dado para a criação do *Framework* além de explicações técnicas referentes a cada funcionalidade do sistema, a documentação completa para uso do *Framework* e algumas aplicações que já utilizam do *Framework*.

Abstract

Nowadays the meta heuristcs have been used for its simply of implementation and for be able to be applied in a very large range of problems whith the most levels of complexity. In this work, we have study the technique named Particle Swarm Optimization (PSO). Which one, as occurs in the most of meta heuristcs, the quality of application of this technique have to define some parameters that have direct influence in the performance of algorithm, like the inertia weight that prevent that the particle change its direction instatly, and the swarm learning ability and its own. Those parameters was evaluted by a group of tests largely used on literature and whith the goal of observe how the algorithm behave in each group of parameters in diferents situations applied.

Furthermore, a larger study about the technique has been done evaluting algorithms that adapt their parameters during the execution, thus searching a tool more effective applied on diferents groups of problems. Algorithms that adapt the inertia weight, for instance, generally allow that the particle keep more free in the opening execution e became more restrective during the search.

Particles on its turn, do not depend olny of the inertia wight to evolve. Its learning is give by two parameters known as cognitive aceleration and social aceleration, which one, define how much the particle is influenced by the swarm and how much she must search the best by itself. (Thus, if implement more social, in other words, that give more importance to the swarm learn, it can lead to failure of all group in case it is lost.) So, it is clear that the definition of those parameters commits the final result of the technique.

Beyond those types of adaptation, still exist algorithms that updates in diferents ways the particle's velocity, according with what those particles have learn, as well as inclusion of new necessary parameters to control the adaptation.

There fore, thiw work aims to evalute various proposals around the definition of PSO parameters in a group of optimization problems without restriction, aiming identify proposals more adequate for diferents classes of problems.

Dedicatória

Dedico esse trabalho a membros de minha família e amigos, principalmente a minha noiva por estar ao meu lado a todo momento me apoiando, mesmo nos momentos mais difíceis, nessa caminhada de dois anos e meio de faculdade e mais dois anos entre a criação do projeto e algumas pausas por motivos pessoais.

Gostaria de agradecer também aos que foram importantes em minha vida, me apoiando e motivando desde a escolha da faculdade até seus momentos finais.

Não podendo deixar de dedicar o trabalho aos companheiros de classe que viveram comigo os momentos fáceis e os mais complicados de toda a trajetória do curso, sem esquecer os que, de algum lugar, me mandou energia e motivação para a conclusão deste trabalho.

Agradecimentos

Ao meu orientador, professores e companheiros de trabalhor que se envolveram no desenvolvimento deste trabalho e deste projeto. Em especial agradeço a minha família e a minha noiva pela motivação e compreensão em momento dificeis e de ausência de minha parte em resultado à dedicação dada a este projeto.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Devido à grande necessidade de entregar projetos de grande porte e com prazos consideravelmente baixos, foi percebida a necessidade de se encontrar soluções que facilitassem esse desenvolvimento.

A primeira atitude a ser tomada foi a criação de um arquivo que reunia diversas funcionalidades, afim de facilitar futuros projetos, onde processos que se repetiam diversas vezes eram colocados em funções que poderiam ser usadas em novos projetos.

Aplicações em geral precisam de um padrão mais significativo como forma estrutural de um projeto, deixando claro que a criação de um arquivo contendo todas as funções do projeto não era o melhor padrão a ser seguido. Este trabalho apresenta, como uma de suas justificativas, uma pesquisa profunda que reune novos padrões para os processos e *Frameworks* web que poderiam ser mais úteis para um desenvolvimento ágil.

No final dessas pesquisas iniciais, alguns *Frameworks* foram testados e o CakePHP passou a ser usado como padrão. O CakePHP utiliza o padrão de projeto MVC *(Model, View, Controller)* que é um modelo de arquitetura de software que tem como objetivo básico separar a lógica de negócio da aplicação.

Os *Frameworks* são sempre muito robustos e com diversos tipos de funcionalidades, e com o CakePHP não é diferente. Com uma vasta documentação e uma quantidade considerável de arquivos em seu projeto mais simples, este passou a ser um problema ao invés de solução quando se busca um total domínio em uma aplicação.

Percebeu-se então a real necessidade de criar um Framework onde se tenha total controle de todas as funcionalidades, mantendo o padrão MVC, porém criando as próprias funcionalidades, mesmo que baseado em funcionalidades de outros *Frameworks*.

Identificar o problema foi o primeiro e principal passo para se iniciar o desenvolvimento do Framework, que se encontra sempre em evolução com novas implementações que resolvam determinados problemas.

2 Métodos e materiais

Com o intuito de apresentar todo o processo de criação e utilização das ferramentas e funcionalidades do *Framework Lotus{PHP}*, neste capítulo serão apresentados e descritos os elementos e métodos utilizados para o desenvolvimento e funcionamento do *Framework*.

2.1 PHP

O PHP (*um acrônimo recursivo para PHP: Hypertext Preprocessor*) é uma linguagem de script open source de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML.i(Php.net,2015).

O PHP contém um HTML com código embutidos que permite unir linguagem de marcação a códigos extremamentes dinâmicos utilizando as tags '*<?php*' e '*?>*' que separam o *HTML* do *PHP*.

É uma linguagem server-side (*lado do servidor*) e tem seu código executado diretamente no servidor, retornando somente o HTML que será exibido para o cliente, omitindo o acesso ao código PHP da aplicação.

2.2 POO *(Programação Orientada a Objetos)*

Programação Orientada a Objetos é um padrão de desenvolvimento com um conjunto de ideias, conceitos e princípios utilizados para facilitar e organizar melhor o desenvolvimento de aplicações. Tem como principais caracteristicas facilitar a manutenção de códigos, utilizar com frequencia o reaproveitamento de códigos além de dimuniur a complexidade no desenvolvimento de sistemas.

2.2.1 Principais conceitos de POO

- **Abstração**: Utilizada para a definição de entidades do mundo real.

Entidade	Características	Ações
Carro, Moto	tamanho, cor, peso, altura	acelerar, parar, ligar, desligar
Elevador	tamanho, peso máximo	subir, descer, escolher andar
Conta Banco	saldo, limite, número	depositar, sacar, ver extrato

Figura 1: Abstração do mundo real.

- **Classes**: Definição dada para a estrutura de um objeto, onde são definidas os atributos e métodos referentes a cada objeto.
- **Objetos**: É a instância de uma classe. Um objeto é a construção de software que encapsula estado e comportamento nos permitindo modelar a aplicação em termos reais e abstrações.
- **Herança**: É a possibilidade de uma classe *(subclasse)* herdar métodos e atributos de outra classe *(superclasse)*

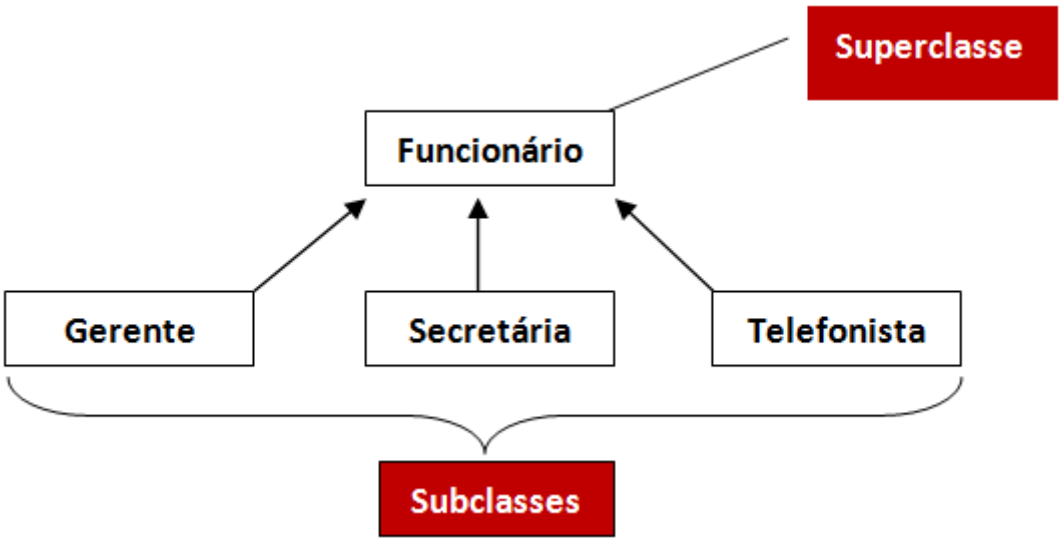

Figura 2: Hieraquia de classes

- **Polimorfismo**: Se trata da capacidade de um método ou comportamento da *superclass* ser implementado de diversas maneiras nas *subclasses*.
- **Encapsulamento**: Forma de proteção dos atributos de uma classe, não permitindo que este seja acessado diretamente.

2.3 MVC *(Model, View e Controller)*

O MVC é um Design Pattern *(Padrão de projeto)* utilizado para separar as camadas de modelo, visão e controle no desenvolvimento de um sistema. A camada de modelo *(Model)* contém classes que implementam a regra de negócios da aplicação, já a camada de visão *(View)*, por sua vez, são responsáveis pela exibição e apresentação dos dados para o usuário, e por fim a camada de controle *(Controller)*, onde é processado todas as requisições realizadas pelo usuários.

A separação da aplicação em camadas, como é feita no padrão MVC, trás uma série de vantagens no processo de desenvolvimento, uma delas é a de permitir a reutilização do mesmo objeto de modelo em visualizações distintas, além de organizar seu projeto de forma onde tudo tenha seu lugar, e cada camada com sua responsabilidade, permitindo um trabalho muitos mais "centrado"e modularizado.

2.4 CRUD *(Create, Read, Update e Delete)*

CRUD é o acrônimo da expressão do idioma inglês, *Create Read Update and Delete* e é utilizado para designar as quarto operaçõs básicas de um banco de dados: Criar, Ler, Atualizar e Deletar.

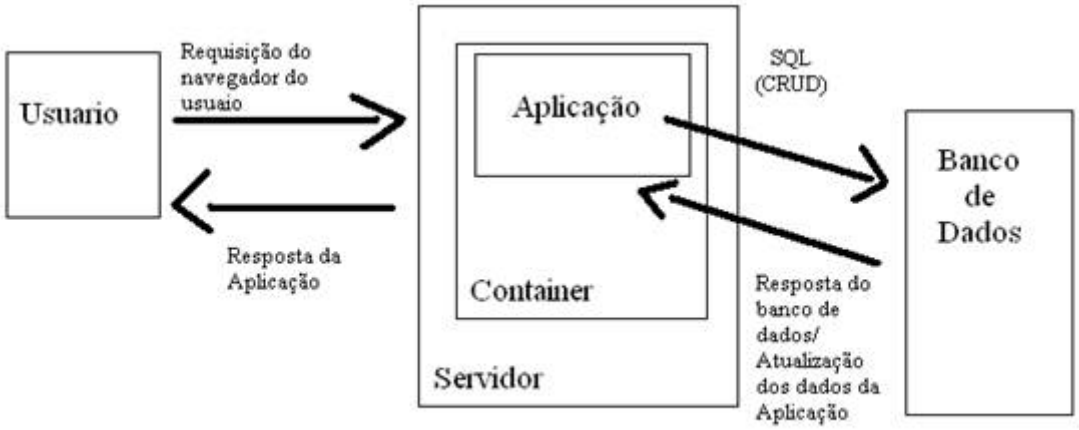

Figura 3: Execução de uma requisição CRUD

Tratando como uma forma mais técnica o CRUD se transforma em um facilitador, criado através de diretivas de programação, para ações ligadas ao *INSERT*, *UPDATE*, *DELETE* e *SELECT* do banco de dados.

2.5 UML

2.6 Mysql

2.7 PDO

2.8 HTML

2.9 Node.js

Node.js é uma plataforma constuida sobre o motor de Javascript que tem como principal objetivo fornecer uma maneira fácil de se construir programas de rede escaláveis. Mesmo sendo um servidor de programas não podemos confundi-lo com um servidor *ready-to-install*(prontos para instalar), que são servidores que estão prontos para instalar aplicativos instantâneamente. O *Node.js* segue o conceito de módulos que podem ser adicionados em seu núcleo. Há literalmente centenas de módulos para rodarem com o Node, e a comunidade é bastante ativa em produzir, publicar e atualizar dezenas de módulos por dia.

2.10 Automatizador Grunt

Grunt é uma ferramenta que roda via termina e serve para automatizar tarefas de uma aplicação, como: concatenação, minificação e validação de arquivos, otimização de imagem, testes unitários, deploy de arquivos por ftp ou rsync, entre outras. O *Grunt* é feito totalmente em *Javascript* e roda no *Node.js*, portanto para ser utilizado, depende da instalação do *Node.js* e do pacote *NPM* previamente instalados.

2.11 Sass

É um pre-processador de folhas de estilo feito em *Ruby* e responsável em auxiliar na produtividade de códigos *CSS*. Literalmente falando, *Sass* é uma extensão do *CSS* que adiciona potência e elegância à linguagem básica. Ele permite ao desenvolvedor o uso de variáveis, mixins, importações, ampla organização do código, entre outras funcionalidade totalmentes compatíveis com *CSS*. *Sass* trabalha com dois tipos de *sintax* diferentes,

.sass e *.scss* e suas particularidas são: enquanto no arquivo *.scss* são utilizados chaves *"{}"* e ponto e vírgula *";"* para delimitar o inicio e fim de atributos e valores, no *.sass* essa delimitação é feita apenas por identação. Para ser utilizado em uma aplicação em produção utilizamos o arquivo *CSS* gerado através da compilação do arquivo *.sass* ou *.scss*.

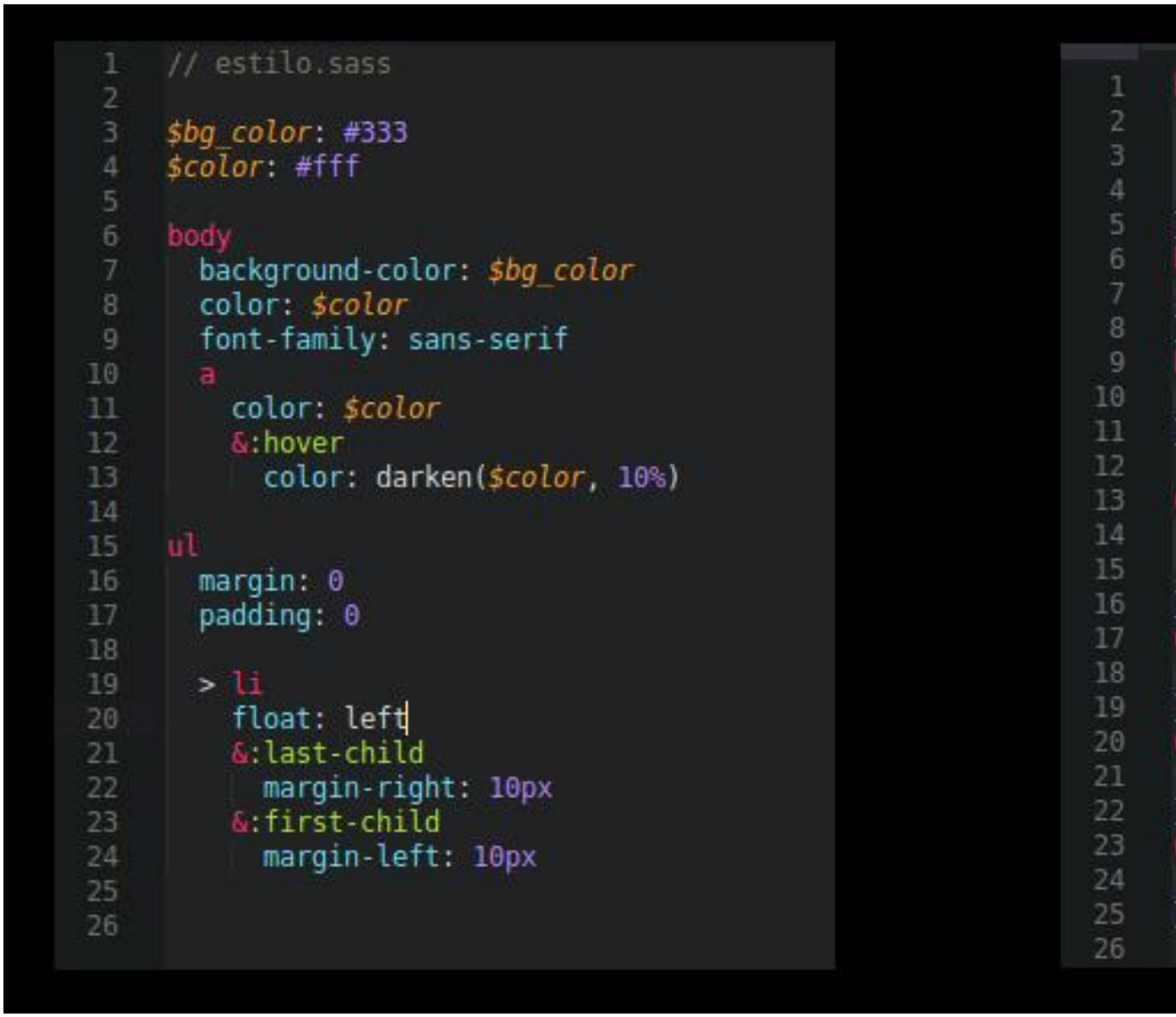

Figura 4: Compilação de um arquivo sass para css

2.12 Compass

2.13 Uglify

2.14 Rsync

2.15 Controle de Versão: Git

2.16 Github

2.17 Twitter Bootstrap

2.18 Javascript

2.19 Jquery

3 *A importância de se usar um Framework*

Neste capítulo será apresentado, com dados técnicos, a importância do uso de um *Framework* em projetos de desenvolvimento, detalhando algumas de suas vantagens e desvantagens no processo de codificação.

O *framework* é, como princípio básico, uma arquitetura "padrão"que tem como objetivo fornecer ferramentas comuns a todo tipo de projeto, utilizando os mais variados tipos de Design Pattern (Padrões de Projeto) afim de proporcionar um ambiente de desenvolvimento extremamente produtivo.

Grande parte dos *frameworks* trabalham com um padrão principal denominado MVC (Model View Controller) que tem como base trabalhar com Modelo Lógico (Model), onde acontece toda a interação com a base de dados do projeto, Visualização (View), que é a parte responsável pela exibição de dados e o Controle (Controller), que é a regra de negócios do projeto, pode-se dizer que o Controller é responsável por fazendo toda a comunicação com o Model e tratar os dados para serem exibidos pela View, resumindo, o padrão MVC separa claramente o Design do Conteúdo e de sua Lógica.

3.1 Vantagens em usar um Framework

- **Padronização em projetos**: A grande vantagem de um *framework* é sua padronização no desenvolvimento. Por utilizar um conjunto já definido de Classes e Métodos, a necesidade em trabalhar conforme a ferramenta possibilita ajuda a garantir um aproveitamento maior de código projetos futuros.

- **Velocidade no desenvimento**: O fato de se fazer uso de módulos genéricos faz com que o *framework* fique responsável por controlar o uso de funcionalidades repetitivas fazendo com que o desenvolvedor se concentre totalmente na regra de negócios de cada projeto.

- **Qualidade**: *Frameworks* em geral são testados e atualizados a todo momento, tornado cada vez mais seguro e com melhores funcionalidades.
- **Re-uso de códigos**: A padronização de projetos torna capaz o re-uso de cógido sem dificuldades de adaptação.
- **Segurança**: Uma das vantagens mais importantes é segurança que o *framework* pode dar ao projeto.
- **Fácil manutenção**: A separação do *framework* utilizando padrões de projetos permite uma fácil manutenção em determinada ferramenta sem que afete outras.
- **Utilitários e Bibliotecas**: Classes e métodos embutidos no *framework* afim de solucionar o problema de repetição contínua de códigos.

3.2 Desvantagens em usar um Framework

Esses pontos não são necessariamente uma desvantagem, porém são os principais motivos pelo qual inibem o desenvolvedor de iniciar em um *framework*.

- **Performance e peso**: A grande quantidade de arquivo e a chamada de métodos e criação de objetos nem sempre necessário para determinados projetos tornam a aplicação pesada em alguns casos.
- **Curva de aprendizado**: Ao se trabalhar com códigos de terceiros existe uma curva de aprendizado elevada e que fica dependente de uma boa documentação para conseguir atingir um bom ritmo de trabalho.
- **Conhecimento técnico**: É necessário que se tenha conhecimento em OOP (Programação Orientada à Objeto), boas práticas de programação e entenda padrões de projetos para poder utilizar o *framework* da melhor forma.

4 Experimentos de Frameworks

4.1 Cake PHP

4.1.1 Descricao da ferramenta

O CakePHP é um projeto de código aberto mantido por uma comunidade bastante ativa de desenvolvedores PHP. Possui uma estrutura extensível para desenvolvimento, manutenção e implantação de aplicativos. Utiliza o padrão de projeto MVC*(Model-View-Controller)* e ORM*(Object-relational mappring)* com os paradigmas das convenções sobre configurações.

4.1.2 Objetivo

CakePHP tem como objetivo principal a simplificação do processo de desenvolvimento e construção de aplicações web, utilizando um núcleo onde organiza o banco de dados e alguns recursos que reduzem a codificação pelo desenvolvedor. Alguns desses recursos são a validação embutida, ACLs *(lista de controle de acesso)*, segurança, manipulação de sessão e cache de Views e sanitização de dados.

4.1.3 Características

- Possui licença flexível ... completar
- Ativo e com comunidade amigável
- Compatível com PHP5
- Geração de CRUD (*Create, Read, Update and Delete, ou Criar, Ler, Atualizar e Excluir*)
- Funciona em qualquer subdiretório web, com poucas configurações no apache

- Utiliza templates

4.1.4 Primeiros Passos

Referências

[Kennedy e Eberhart 1995]KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE. *Neural Networks, 1995. Proceedings., IEEE International Conference on*. [S.l.], 1995. v. 4, p. 1942–1948.