# Avaliação de Ferramentas CMS Ruby on Rails

Rafael Strecker Coelho de Souza *

November 13, 2010

Introdução

A quantidade de conteúdo e informações digitais vem crescendo de forma significativa. Segundo o ISC o crescimento de dominios registrados é muito grande, chegando próximo ao bilhão de domínios.

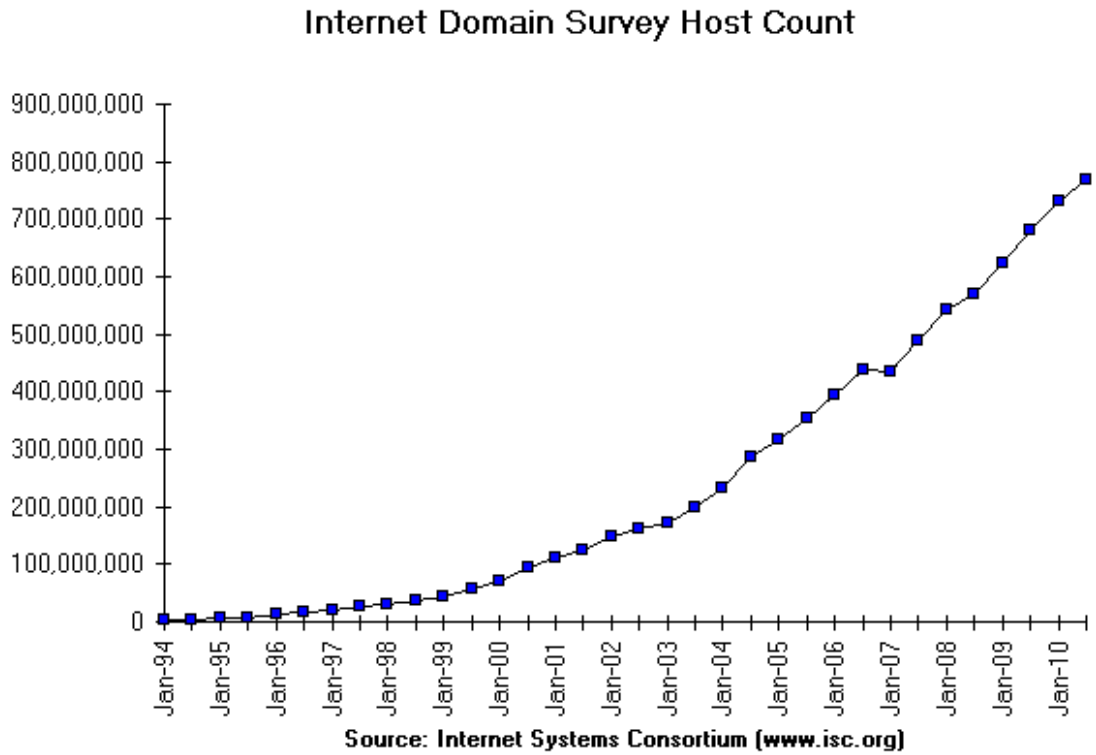


Figure 1: Número de domínios registrados dos últimos anos

*bloghttp://www.cnxs.com.br – contact mailto:ebellani@gmail.com

Diante desses fatos, é intrínseco a necessidade de gerenciar e publicar esses conteúdos de forma rápida, tanto por indivíduos como organizações. A partir desta necessidade, surgem os CMS (Content Management Systems) que proporcionam versatilidade, quando o objetivo é produzir de forma rápida e organizada, soluções integradas para a exibição de conteúdo na internet.

Na escolha de um CMS para desenvolvimento deve-se pesar também não só a ferramentas mas também a plataforma a qual é desenvolvida, neste trabalho o foco será nas ferramentas da plataforma Ruby on Rails, que está se tornando muito popular nos ultimos anos.

A usabilidade, por sua vez com o passar dos anos, tem recebido maior atenção, pois foi reconhecida como uma propriedade fundamental no sucesso das aplicações web.

Definir os métodos para a usabilidade é, portanto, uma das metas atuais de pesquisas. Além disso, as empresas tem reconhecido a importância dos métodos de usabilidade durante o processo de desenvolvimento, para verificar a usabilidade das aplicações, antes da sua implantação.

Alguns estudos têm demonstrado, de fato, como o uso de tais métodos permite reduzir custos, em uma relação custo-benefício, pois reduzem a necessidade de mudanças após a entrega dos aplicativos.

A satisfação do usuário final, ao atingir seus objetivos e resultados com eficiência, está relacionado diretamente a usabilidade e qualidade dos sistemas interativos. Principalmente se estes forem fáceis de serem utilizados e permitam de forma rápida criar e atualizar, também econômico, pois o acesso a esses CMS são gratuitos.

Fazendo uma analogia, percebe-se que gerenciamente de conteúdos e usabilidade estão diretamente relacionados, pois a usabilidade é um fator fundamental e intrínseco para o sucesso e qualidade de um projeto e satisfação do usuário, que trará benefícios significativos e tangíveis para o sistema de conteúdo.

# 1 Justificativa

Com a popularização do modelo open source de desenvolvimento, muitas ferramentas CMS já foram desenvolvidas e distribuidas sob licensas livres na internet. Diante deste cenário, um profissional de Sistemas de Informação deve avaliar entre as opções disponíveis e dependendo do contexto definir a utilização de uma ou outra ferramenta.

Este trabalho procura auxiliar esta avaliação, abordando a usabilidade e a comunidade ao redor das ferramentas CMS escritas em Ruby on Rails, como parâmetros para a avaliação.

# 2 Objetivos Gerais

Aprofundar o conhecimento em CMS, usabilidade e nas ferramentas avaliadas em Ruby on Rails, traçando um comparativo entre elas sob a ótica da usabilidade e da comunidade em torno das ferramentas.

# 3 Objetivos Específicos

1. Analisar as qualidades e limitações das ferramentas CMS avaliadas, utilizando técnicas de avaliação de usabilidade.

Estrutura do Trabalho

Nos próximos capítulos, são apresentados um aprofundamento sore Ruby, Ruby on Rails,CMS e Usabilidade. Nos capítulos à seguir são descritas as ferramentas e a metodologia empregada na avaliação. Em seguida é feita a avaliação entre as ferramentas e uma conclusão sobre o trabalho, com uma análise crítica e sugestões de trabalhos futuros.

Ruby

> Ruby é uma linguagem de programação de código aberto e orientada a objetos criado por Yukihiro 'Matz' Matsumoto. A primeira versão foi lançada no Japão em 1995. Ruby tem ganhado aceitação ao redor do mundo como uma linguagem fácil de aprender, poderosa, e expressiva, especialmente desde o advento do Ruby on Rails, um framework para aplicações voltadas para Web escrito em Ruby[1]. O núcleo de Ruby é escrito na linguagem de programação C e roda na maioria das plataformas. É uma linguagem interpretada e não compilada. Fitzgerald [2007]

[2]

Uma descrição pouco mais detalhada sobre Ruby:

> Ruby é uma linguagem de programação dinâmica com uma complexa, porém expressiva gramática e um núcleo de biblioteca de classes com uma rica e poderosa API. Ruby tem inspirações em Lisp,SmallTalk e Perl, mas usa uma gramática que facilita o entendimento de programadores Java e C. Ruby é uma linguagem

[1] http://www.rubyonrails.org

[2] Tradução livre do autor

> orientada à objetos pura, mas também suporta estilos de programação funcional e procedural.Ela inclui uma poderosa capacidade de metaprogramação e pode ser usada para criar linguagens de domínio específico ou DSLs. **?**

[3]

Ruby é escrita em C e atualmente está disponível para as mais diversas plataformas UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP/Vista/Seven, DOS, BeOS, OS/2, .NET, Solaris (www.ruby lang.org [2009]).

O criador de Ruby, Yukihiro 'Matz' Matsumoto, teve a idéia de criar uma linguagem de programação em 1993, durante uma conversa um um colega sobre linguagens de script. Através da conversa, o seu interesse cresceu muito sobre as linguagens de script, onde ele ficou impressionado pelo poder e pelas possibilidades. Como fora um fã de longa data de programação orientada à objetos, lhe pareceu , a orientação à objetos, muito adequada para linguagens de script. Então passou olhar pela rede e encontrou Perl 5, que ainda não havia sido lançada, contendo algumas características de orientação à objetos, mas ainda não era o que ele queria. Então decidiu abandonar Perl como uma linguagem de script orientada à objetos. Então ele procurou Python, era uma linguagem orientada à objetos interpretada. Mas ele não teve a sensação que era uma linguagem de script, era uma linguagem híbrida, procedural e orientada à objetos. Matz queria uma linguagem de script que seria mais poderosa que Perl e mais orientada à objetos que Python. Então ele decidiu criar a própria linguagem. O nome Ruby veio de uma brincadeira com um amigo durante o projeto de desenvolvimento da linguagem, ele queria um nome de uma pedra preciosa, a là Perl, e então o amigo sugeriu Ruby, que depois se tornou o nome oficial da linguagem. (**?**).

A filosofia de Ruby segundo o Yukihiro 'Matz' Matsumoto: *Ruby foi projetado para fazer programadores mais felizes* Flanagan and Matsumoto [2008]página 1[4]

Abaixo um exemplo de uma classe Ruby

No exemplo que segue podemos visualizar como Ruby define uma classe e cria um objeto. A definição de classe se dá pelo comando *class*. No exemplo o nome da classe é *Pessoa*. Na linha 2 há uma declaração de um atributo da classe $attr_acessor, estadeclaracãofazcomqueoRubygereautomaticamentemétodosdeacessoresem$

*Nas linhas 15 e 16 criaremos 2 objetos da classe pessoa passando argumentos diferentes para cada um deles. Nas linhas subsequentes iremos escrever na tela do console o valores retornados pelo método agradecer que vai retornar o agradecimento correto para* homem *e* mulher.

[3]Tradução livre do autor

[4]Tradução livre do autor

*[ language=Ruby, emph=catch,class,def,else,end,if,require,return,then,throw,unless,attr$_a$cesso*
*r, emph = [2]initialize, new, emphstyle = [2], emph = [3]Pessoa, emphstyle = [3], emph = [4]@nome, emphstyle = [4], basicstyle =, title =, numbers = left]code/exemplo$_r$uby.rb*

*Em Ruby, como é uma linguagem orientada à objeto pura, tudo é um objeto. Desde números aos valores booleanos true e false ao nil (versão de Ruby para null, indicação de falta de um valor).(*

*O exemplo abaixo demonstra que valores numéricos, valores booleanos, e o nil podem invocar o método* class *que retorna a sua classe e imprimiremos o resultado na tela do console.*

*[ language=Ruby, emph=catch,class,def,else,end,if,require,return,then,throw,unless, emphstyle=***, emph=[2]initialize,new, emphstyle=[2], emph=[3]Pessoa, emphstyle=[3], emph=[4]@nome, emphstyle=[4], basicstyle=, title=, numbers=left ]code/exemplo$_o$bjetos.rb**

***O resultado impresso na tela pelo exemplo é:***
***$ ruby exemplo$_o$bjetos.rb***
*Fixnum*
*TrueClass*
*NilClass*

***Cada linha é a resposta do nome da classe ao qual cada valor pertence. Por exemplo o número 1 é da classe* Fixnum*, o número 1.1 pertence a classe* Float *e assim por diante.***

***O interpretador mais usado e conhecido de Ruby é o MRI, ou Matz Ruby Interpreter, escrito em C e a sua versão atual é 1.9.2. Além dele há outros interpretadores como JRuby, Rubinius, Maglev, MacRuby e IronRuby.***

## 3.1 JRuby

***JRuby é um interpretador Ruby rodando sobre a máquina virtual Java (JVM). Ele está na versão 1.5.2 que é compatível com o MRI 1.8.7.***

## 3.2 Rubinius

***Rubinius que foi originalmente idealizado para ser o interpretador de Ruby feito em Ruby, mas em virtude da compatibilidade com o MRI parte do seu código é escrito em C++. Está na versão 1.0.1 e é compatível ao MRI 1.8.7*** **Um grande aspecto de linguages populares como Java e C, é que a maioria das funcionalidades disponíveis para os programadores é escrito na própria linguagem. O objetivo de Ribinius é adicionar Ruby à estas linguagens. Desta forma Rubystas podem facilmente adicionar funcionalidades à linguagem, arrumar defeitos, e**

**aprender como a linguagem funciona. Onde é possível Rubinius é escrito em Ruby. Onde (ainda) não é escrito em C++.** ***rub***[5]

## 3.3 Maglev

***Maglev é o um interpretador Ruby desenvolvido pela empresa Gemstone. Conhecida pelo seu interpretador de Smalltalk, a empresa se lançou como uma plataforma alternativa, já que inclui um mecanismo de persistência de objetos distribuídos. Está em versão Alpha ainda.*****Maglev é uma implementação de Ruby rápida e estável com uma integrada persistência de objetos e cache compartilhado e distribuído.** ***mag [a]***[6]

## 3.4 MacRuby

***Falar sobre MacRuby***

## 3.5 IronRuby

***IronRuby que é um interpretador Ruby feito sobre a plataforma .NET. Ele era patrocinado pela Microsoft até Julho de 2010, onde a empresa cortou o último desenvolvedor remanescente do projeto. Está na versão 1.0 e com um futuro incerto pois ainda não apareceu nenhum outro mantenedor oficial.***

***Ruby On Rails***

**Ruby on Rails é um framework projetado para escrever aplicações web de forma rápida e simples. Extraído do Basecamp, ferramenta de gerenciamento de projetos da 37 Signals, o criador David Heinemeier Hanson escreveou Rails com soluções já praticadas para problemas comuns do mundo real. Estas soluções e decisões são o que torna o Rails prazeroso de usar - as chatas, tarefas servis, ja estão feitas, deixando você apenas para concentrar no seu problema. ?*** ***página 1***[7]

***Ruby on Rails é um framework escrito em Ruby, lançado em 2004, se tornou popular em 2005 à partir de uma palestra feita no Fórum Internacional de Software Livre pelo seu criador David Heinemeier Hanson, onde mostrou como criar uma weblog usando Rails em 15 minutos.***

***Rails foi extraído de uma aplicação real, o Basecamp, uma ferramenta de gerenciamento de projetos da empresa 37 Signals. Seu criador teve base em outros frameworks web, utilizando idéias de cada um***

[5]Tradução livre do autor
[6]Tradução livre do autor
[7]Tradução livre do autor

*e combinado à expressividade da linguagem Ruby, uma boa divulgação, atingiu uma grande popularidade. O framework introduziu a linguagem Ruby para o ocidente, e trouxe bastante atenção de programadores interessados nas facilidades providas pelo framework.*

*Uma de suas filosofias é a convenção ao invés de configuração (CoC-Convention over Configuration). David era contra os frameworks que abusavam de configurações em arquivos XML, então ele partiu para abolir ao máximo possível arquivos de configuração. Mesmo os arquivos de configuração de Rails não utilizam XML e sim um outro formato de serialização chamdo YAML (Yet Another Markup Language). Outra filosofia adotada é o DRY (Don't Repeat Yourself), onde dita que a informação deve ser localizada em um único local.*

*Uma característica da arquitetura é o padrão MVC, Model View Controller, que ele utiliza. Este padrão é seguido pelos frameworks web mais populares. A figura abaixo 2 demonstra o funcionamento do MVC.*

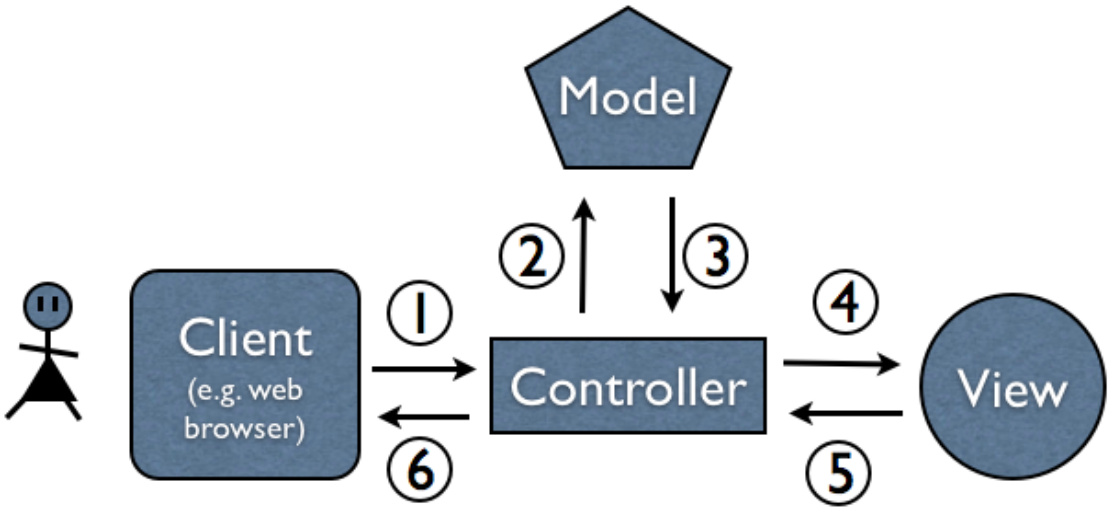


Figure 2: Arquitetura MVC

***Podemos observar na figura que:***

1. ***O cliente a partir do seu browser faz uma requisição para uma página.***

2. ***Esta requisição é encaminhada ao controller, que tem a responsabilidade de determinar o destino da requisição. Ele deve pedir ao model alguns dados necessários para completar a requisição.***

3. *O model que é responsável pela comunicação com a base de dados, então faz uma chamada SQL para buscar os dados requisitados pelo controller e os encaminha.*

4. *O controller então encaminha para a view estes dados oriundos do model.*

5. *A view detém a responsabilidade de combinar estes dados com um modelo de html e css, gera a página html de resposta para o browser do cliente.*

*A estrutura do framework vem separada em diversos módulos, são eles:*

- *ActiveRecord - Módulo de Mapeamento Objeto Relacional, usado para facilitar o acesso e a manipulação da base de dados.*
- *ActionController - Módulo responsável pelo controle, usado para gerenciar todo o ciclo de vida de uma requisição.*
- *ActionView - Módulo responsável pela renderização dos dados, é a parte visível para o browser.*
- *ActionMailer - Módulo de envio de emails, age similar ao ActionView, lidando com e-mails.*
- *ActiveResource - Módulo de web services, auxilia a gerar e consumir web services RESTful.*

*Além destes módulos Rails conta uma arquitetura de plugins que fomenta a criação e extensão do framework. A partir desta arquitetura é possível extender, modificar e criar novas funcionalidades para o framework. Esta arquitetura fácil levou a criação de milhares de plugins para Rails.*

*Uma característica que faz do Rails uma opção popular são os seus geradores. São pedaços de códigos gerados automaticamente a partir de comandos enviados pelo console. Os geradores tornaram bem simples a geração de cadastros CRUD (Create Retrieve Update Delete), uma das atividades mais repetitivas do desenvolvimento web.*

*CMS*

*CMS é a sigla para Content Management System, ou sistema gerenciador de conteúdo. É a idéia de um gerenciamento da informação de organizações que produzem muito conteúdo. São baseados na web, auxiliando em vários aspectos a publicação de conteúdo, desde sua forma de apresentação ao seu controle de versão. Estes CMS servem para a publicação de conteúdo,gerenciamento de transação de e-commerces,*

*Wikis, gerenciamento de documentos, entre outras atividades. Sistema de Gerenciamento de Conteúdo podem de maneira simples e óbvia serem definidos por sua sigla, um sistema que gerencia conteúdos. Para uma melhor definição de CMS - Content Management Systems, o assunto será abordado a teoria de conteúdo e gestão dos mesmos.*

**Podemos dizer que um CMS é um framework, "um esqueleto" de website pré-programado, com recursos básicos e de manutenção e administração já prontamente disponíveis. É um sistema que permite a criação, armazenamento e administração de conteúdo de forma dinâmica, através de uma interface de usuário via Internet. Um CMS permite que a empresa tenha total autonomia sobre o conteúdo e evolução da sua presença na internet e dispense a assistência de terceiros ou empresas especializadas para manutenções de rotina. Nem mesmo é preciso um funcionário dedicado (webmaster), pois cada membro da equipe poderá gerenciar o seu próprio conteúdo, diminuindo os custos com recursos humanos. A habilidade necessária para trabalhar com um sistema de gerenciamento de conteúdo não vai muito além dos conhecimentos necessários para um editor de texto.** *nav*

*Linguisticamente, (CMS) significa qualquer sistema que auxilia no gerenciamento de conteudo - criação, armazenamento, indexação, arquivamento, publicação e distribuição do conteúdo. wha*[8]

*Em suma, o grande diferencial de um CMS é permitir que o conteúdo de um website possa ser modificado de forma rápida e segura de qualquer computador conectado à Internet. Um sistema de gerenciamento de conteúdo reduz custos e ajuda a suplantar barreiras potenciais à comunicação web reduzindo o custo da criação, contribuição e manutenção de conteúdo. nav*

# 4 O que é Conteúdo

*Há várias definições para Conteúdo, no contexto de CMS, uma visão mais simples de conteúdo vem de ? "O termo conteúdo representa qualquer conteúdo eletrônico, incluindo registros, dados e metadados, como também documentos e websites". A definição de Boiko [2005] "Conteúdo, portanto, é a informação que você rotula com dados para então um computador poder organizar e sistematizar a coleção, gerenciamento e publicação". Estas definições apresentam algumas diferenças, mas nota-se que sem o auxílio de uma ferramenta específica para gerenciar conteúdos, seria uma tarefa deveras árdua.*

[8]Tradução livre do autor

# 5 Tipos de CMS

*Como CMS é um conceito amplo, existem várias classificações entre os Gerenciadores de Conteúdo. Alguns, podem ser mais específicos como o funcionamento de blogs ou wikis, outros mais generalistas como os Web Content Management Systems (WCMS).*

*Segundo Mehta [2009] eis as seguintes classificações*

## 5.1 Portais ou CMS genéricos ou WCMS

*Estes CMS são muito populares, geralmente encontrados na confecção de sites corporativos, de pequeno até grande porte no caso de Portais. Eliminam em grande parte a complexidade de o site ser administrado por uma pessoa técnica, conhecida como webmaster. Com este tipo de CMS pessoas com pouco conhecimento técnico podem publicar conteúdos.*

### Principais Características

- *Criar a gerenciar seções de conteúdos.*
- *Criar páginas e adicionar conteúdos de textos ou imagens.*
- *Editar conteúdo publicado.*
- *Administração por múltiplos usuários.*
- *Versionamento de conteúdo.*
- *Gerenciamento de Workflow.*

### Exemplos de WCMS

- *BrowserCMS*
- *RadiantCMS*
- *RefineryCMS*
- *ZenaCMS*
- *Drupal*
- *Joomla*
- *Liferay*

## 5.2 Blog CMS

***Blogs também são considerados CMS pois são autorais, publicam conteúdo de texto,imagem,vídeos.***

### Principais Características

- ***Criar posts.***
- ***Categorizar Posts.***
- ***Gerenciar comentários.***
- ***Adicionar imagens,videos, textos.***

### Exemplos de Blog CMS

- ***WordPress***
- ***Mephisto***
- ***Typo***
- ***Blogger***

## 5.3 Wiki CMS

**Wiki é uma página ou coleção de páginas web projetadas para permitir o acesso, a qualquer usuário, para contribuir ou modificar o conteúdo (excluindo os usuários bloqueados), utilizando uma linguagem de marcação simplificada. Wikis são geralmente usados para criarem sites colaborativos e ampliar sitemas de comunidades. *Mehta [2009]página 22*[9]**

### Principais Características

- ***Facilidade de criar páginas, chamadas de wikiweb.***
- ***Linguagem de marcação simples.***
- ***Criação de links automatizadas, mesmo que o link ainda não exista.***
- ***Sistema completo de versionamento.***
- ***Pode ter acesso restrito à usuários ou grupos.***

[9]Tradução livre do autor

### Exemplos de Wiki CMS

- ***MediaWiki***
- ***TWiki***
- ***Typo***
- ***Blogger***

## 5.4 eLearning CMSs

***eLearning CMS são gerenciadores de conteúdos especializados em ensino à distância. Também conhecidos como LMS Learning Management Systems, eles tem responsabilidade pela administração, documentação, registro e relatório de cursos.***

### Principais Características

- ***Gerenciar cursos, estudantes, professores.***
- ***Criar um curso e um programa de aprendizado.***
- ***Criar documentos, testes, discussões e anúncios.***
- ***Sistema de chat, forum, blogs, etc.***

### Exemplos de eLearning CMS

- ***Dokeos***
- ***Moodle***
- ***LAMS***

***Mehta [2009]***[10]
***Usabilidade***

***Por muitos anos, as interfaces encontradas, eram de difícil manuseio e confusas o que repelia parte de seus usuários, muitos quando se deparavam com tais ferramentes não sentiam-se confortáveis e abandonavam.***

***Mas, com o alto crescimento de usuários das ferramentes web, viu-se a necessidade de criar CMS mais simples que facilitassem a criação e manutenção de sites, pois o usuário precisa conseguir utilizar e desejar usar novamente, e isso só acontecerá se ele encontrar o que procura com facilidade.***

[10]Tradução livre do autor

> ***Se a interface com o usuário for muito rígida, lenta, desagradável, pessoas se sentem frustradas, abandonam e esquecem o produto. usa***[11]

***A norma da International Organization for Standardization 9241 define usabilidade como "É a medida pela qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com efetividade, eficiência e satisfação em um contexto de uso específico"***

> ***A usabilidade visa impactar positivamente sobre o retorno do investimento para a empresa. Ela será argumento de vendas, passará uma imagem de qualidade, evitará prejuízos para os clientes, ligados ao trabalho adicional e ao "retrabalho" de correções freqüentes, por exemplo. A empresa desenvolvedora economizará custos de manutenção e de revisões nos produtos, como mostra o texto sobre Engenharia de Usabilidade. nie [b]***

***te***

***Sistemas que visam uma boa usabilidade tem como dever fomentar a criação de interfaces simples, de modo a não dificultar os processos, ajudando o usuário a ter controle de todo o ambiente sem ser obstrusiva.***

***Projetar visando a usabilidade envolve estabelecer os requisitos de usuários para um novo sistema ou produto, desenvolver soluções de desing, protótipos do sistema e da interface com o usuário, e testar com os usuários significativos. No entanto, antes de qualquer atividade de projeto ou avaliação de usabilidade começar, é necessário compreender o Contexto de uso do produto, exemplificando, os objetivos da comunidade de usuários, o usuário principal, tarefas e as característias do ambiente em que o sistema irá operar. mag [b]***[12]

***Interfaces com baixa qualidade de uso trazem diversos problemas, dentre os quais: confusão aos usuários, treinamento excessivo, desmotivação a exploração, indução ao erro, entre outras. Estes problemas podem ser detectados por diversos métodos de avaliação, realizados ao longo do processo de desenvolvimento. Os métodos de avaliação mais utilizados se concentram em avaliar a usabilidade de um sistema. mag [b]***[13]

***A usabilidade pode ser dividida em cinco princípios básicos***

[11] Tradução livre do autor
[12] Tradução livre do autor
[13] Tradução livre do autor

- *Intuitividade: O sistema deve apresentar facilidade de uso permitindo que, mesmo um usuário sem experiência, seja capaz de produzir algum trabalho satisfatoriamente.*
- *Eficiência: O sistema deve ser eficiente em seu desempenho apresentando um alto nível de produtividade.*
- *Memorização: Suas telas devem apresentar facilidade de memorização permitindo que usuários ocasionais consigam utilizá-lo mesmo depois de um longo intervalo de tempo.*
- *Erro: A quantidade de erros apresentados pelo sistema deve ser o mais reduzido possível, além disso, eles devem apresentar soluções simples e rápidas mesmo para usuários iniciantes. Erros graves ou sem solução não podem ocorrer.*
- *Satisfação: O sistema deve agradar ao usuário, sejam eles iniciantes ou avançados, permitindo uma interação agradável.*

*nie [b]*

*A partir destes princípios, é possivel aprofunda-los e especializar em alguns critérios que podem ser avaliados entre diferentes métodos de avaliação.*

*Existem algumas maneiras de se avaliar a usabilidade e ergonomia de um sistema, explorando os critérios a partir dos princípios acima descritos.*

# 6 Métodos de Avaliação

*Se a interface não tem uma boa qualidade, muitos problemas podem ser desencadeados. As mesmas podem ser confusas ao usuário, o que gerará muito treinamento, também a desmotivação em explorar as ferramentas o que induzirá ao erro, entre outros aspectos. Esses problemas podem ser detectados por diversos métodos de avaliação, que são realizados ao longo do processo de desenvolimento, esses concentram-se em avaliar a usabilidade do sistema. ?*

*Avaliação de Usabilidade é um nome genérico para um conjunto de métodos que são baseados em avaliadores inspecionando a interface. Tipicamente, inspeção de usabilidade é direcionado a procurar problemas de usabilidade em uma interface. Vários métodos de inspeção focam nas especificações de interface com o usuário, que podem não estar necessariamente implementada, isso significa que a inspeção pode ser feita em estágios primários do ciclo de vida da engenharia de usabilidade. nie [a]*

*Há 3 principais técnicas de avaliação de ergonomia. São elas:*

- ***Técnicas Prospectivas***
- ***Técnicas Preditivas ou Diagnósticas***
- ***Técnicas Objetivas ou Empíricas***

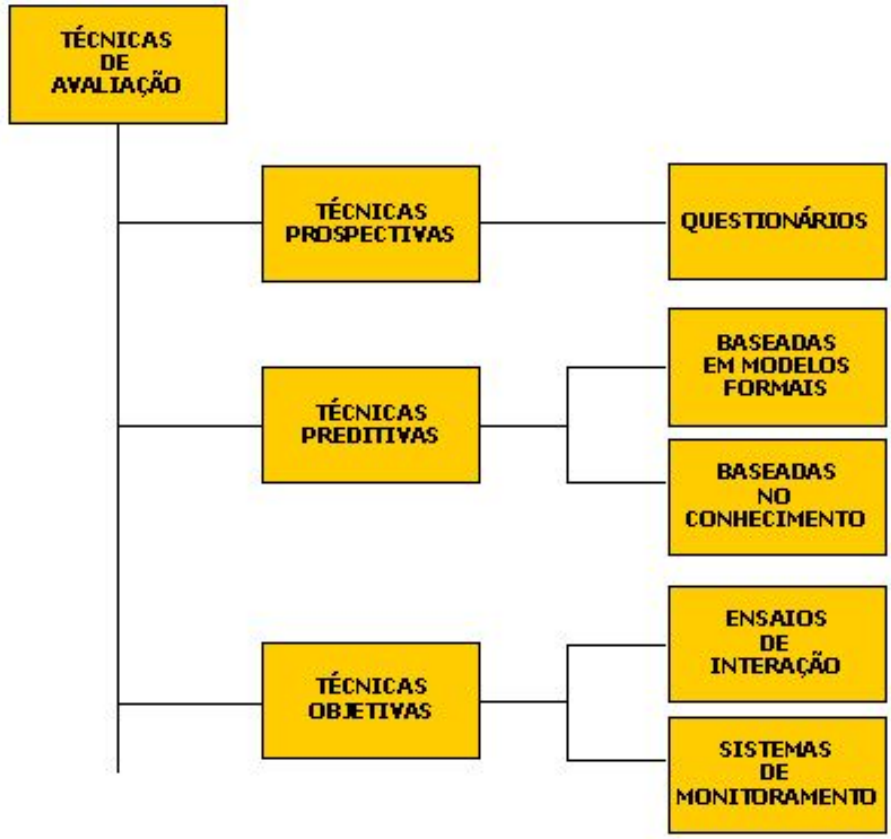


Figure 3: Diagrama das tecnicas de usabilidade

## 6.1 Técnicas Prospectivas

***Este tipo de técnica está baseada na aplicação de questionários/entrevistas com o usuário para avaliar sua satisfação ou insatisfação em relação ao sistema e sua operação. Ela mostra-se bastante pertinente na medida em que é o usuário a pessoa que melhor conhece o software, seus defeitos e qualidades em relação aos objetivos em suas tarefas. Nada mais natural em buscar suas opiniões para orientar revisões de projeto.***

***Muitas empresas de software elaboram e aplicam regularmente este tipo de questionário, como parte de sua estratégia de qualidade, porém***

*constatou-se que os questionários de satisfação têm uma taxa de devolução reduzida (máximo 30% retornam), o que indica a necessidade de elaboração de um questionário mais dinâmico, com questões mais sucintas e diretas e que tenham espaço para opiniões e sugestões, pois questionários longos, tornam-se cansativos. cyb*

*As técnicas prospectivas mais comuns são: questionários de opinão dos usuários, registros de uso do sistema e coleta de opiniões de especialistas.*

## 6.2 Técnicas Preditivas ou Diagnósticas

*Técnicas preditivas são baseadas em avaliações de especialistas, e na competência destes avaliadores.*

*As técnicas diagnósticas dispensam a participação direta de usuários nas avaliações, que se baseiam em verificações e inspeções de versões intermediárias ou acabadas de software interativo, feitas pelos projetistas ou por especialistas em usabilidade. cyb*

*As avaliações podem ser divididas em:*

## 6.3 Avaliações Analíticas

*Essa técnica é empregada nas primeiras etapas da concepção de interfaces humano-computador, quando ela não passa de uma descrição da organização das tarefas interativas. Mesmo nesse nível, já é possível verificar questões como a consistência, a carga de trabalho e o controle do usuário sobre o diálogo proposto e a realizar. A especificação da futura tarefa interativa, pode ser realizada nos termos de um formalismo apropriado como MAD, GOMS (Goals, Operators, Methods and Selections rules) e CGL (Command Grammar Language).*

*Em particular, GOMS propõe uma tabela associando tempos médios de realização aos métodos primitivos, que correspondem as primitivas ações físicas ou cognitivas. Com base na descrição da tarefa realizada segundo o formalismo é possível calcular os tempos prováveis para a realização das tarefas previstas. cyb*

*As avaliações analíticas se dividem em métodos formais e aproximados.O método formal exige inspeção cuidadosa de seqüências de ação que um usuário realiza para concluir uma tarefa. Isto também é chamado "keystrokelevel analysis". Isto envolve dividir a tarefa em ações individuais, como movimentar o mouse para o menu ou o digitar no teclado e calcular o tempo que leva estas ações.*

*O método aproximado é menos detalhado e provém resultados menos precisos, porém podem ser efetuados de maneira muito mais rápida.*

*Envolve um processo similar de sequencia de ações que um usuário realiza nos quesitos físico,cognitivo e perceptivo.*

*As vantagens de avaliações analíticas estão em uma predição precisa do quanto tempo leva a execução de uma tarefa e uma análise profunda do comportamento do usuário durante a mesma.*

*As desvantanges são o tempo e custo disto e ainda requerem uma avaliadores de alta capacidade. and*

### 6.3.1 Avaliações Heurísticas

*Uma avaliação heurística é um método de análise de interfaces e qualidades ergonômicas das interfaces humano-computador, com base no conjunto de critérios de usabilidade. Os princípios são chamados de heurísticas pois são desenvolvidos a partir de uma série de experiências prévias, sintetizando pontos recorrentes. Essa avaliação é realizada por especialistas em ergonomia, baseados em sua experiência e competência no assunto. Eles examinam o sistema interativo e diagnosticam os problemas ou as barreiras que os usuários provavelmente encontrarão durante a interação. cyb*

*Avaliação heurística é o mais informal de métodos de inspecção. Ela é formada por um pequeno conjunto de especialistas, para analisar a aplicação através de uma lista de princípios de usabilidade reconhecidos - a heurística. Esta técnica é parte dos métodos de usabilidade chamados de* discount*. Ela é um método muito eficiente de engenharia de usabilidade,com uma relação custo-benefício alta.*

*Jakob Nielsen sugere um conjunto de 10 regras heurísticas para guiar uma avaliação, são elas:*

- *Diálogos simples e naturais: O sistema deve apresentar facilidade de uso permitindo que, mesmo um usuário sem experiência, seja capaz de produzir algum trabalho satisfatoriamente.*
- *Saídas claramentes definidas: O usuário controla o sistema e pode a qualquer momento abortar uma função ou tarefa indesejada e retonar ao estado anterior.*
- *Atalhos: Para usuários experientes executarem as tarefas mais rapidamente.*
- *Consistência: Um mesmo comando deve ter o mesmo efeito sempre.*
- *Feedback: O sistema deve informar continuamente ao usuário o que ele está fazendo, atraves de uma resposta em um tempo curto.*

- *Memorização: Suas telas devem apresentar facilidade de memorização permitindo que usuários ocasionais consigam utilizá-lo mesmo depois de um longo intervalo de tempo.*
- *Prevenção de erros: A quantidade de erros apresentados pelo sistema deve ser o mais reduzido possível.*
- *Boas mensagens de erro: Linguagem clara, devem ajudar o usuário a entender o problema, não devem intimidar o usuário.*
- *Satisfação: O sistema deve agradar ao usuário, sejam eles iniciantes ou avançados, permitindo uma interação agradável.*
- *Ajuda e documentação: O ideal é que o sistema seja tão intuitivo que não precise de ajuda. Se for necessária, deve ser de fácil acesso online.*

### 6.3.2 Inspeções por Checklist

> *As inspeções de usabilidade por checklists, são vistorias baseadas em listas de verificação, através das quais profissionais, não necessariamente especialistas em ergonomia, como por exemplo, programadores e analistas, diagnosticam rapidamente problemas gerais e repetitivos das interfaces. Neste tipo de técnica, ao contrário das avaliações heurísticas, são as qualidades da ferramenta (checklists) e não dos avaliadores, que determinam as possibilidades para a avaliação. Checklists bem elaborados devem produzir resultados mais uniformes e abrangentes, em termos de identificação de problemas de usabilidade, pois os inspetores são conduzidos no exame da interface através de uma mesma lista de questões a responder sobre a usabilidade do projeto. cyb*

## 6.4 Técnicas Objetivas ou Empíricas

*As técnicas objetivas, se baseiam na participação direta de usuários através de 2 principais formas:*

- *Ensaios de interação*
- *Sistemas de Monitoramento de interação*

*Ferramentas Avaliadas*

*As ferramentas avaliadas pertencem ao grupo dos Web Content Management Systems, e dentro deste contexto elas serão avaliadas.*

# 7 RadiantCMS

> ***Radiant é um sistema de gerenciamento de conteúdo aberto, simples e projetado para pequenas equipes rad***[14]

***Radiant é um dos CMS mais antigos disponíveis para Ruby, lançado em 2006. De instalação e uso simples tem uma grande quantidade de plugins disponíveis e uma grande comunidade mantenedora da aplicação.***

## 7.1 Características

- ***Interface intuitiva***
- ***Bom sistema de extensões, com um número muito grande de extensões disponíveis.***
- ***Sistema de usuários e permissões simples.***
- ***Snippets***
- ***Linguagem específica de domínio Radius.***


Figure 4: Interface de administração do Radiant

***Radiant tem um sistema de usuários e permissões simplificado, bom para pequenas organizações que não necessitam de grande complexidade no acesso. Possui algumas extensões de código aberto que adicionam um maior leque de permissões e funcionalidades ao sistema.***

[14]Tradução livre do autor

*Os Snippets são pedaços reutilizáveis de código, que podem ser inseridos em qualquer página ou template (layout), são úteis e dentro da filosofia de Rails DRY, Don't Repeat Yourself.*

*Radius é uma linguagem específica de domínio escrita em Ruby, baseada em tags, semelhante a XML e HTML. É utilizada dentro do Radiant para prover algumas funcionalidades como os snippets e outras marcações que encapsulam parte da lógica da aplicação. É possível a partir dela gerar qualquer forma de texto puro ou html. Radiant utiliza várias tags Radius para auxiliar no desenvolvimento de uma aplicação como:*

- *Tags que geram conteúdos:* `<r:author />`,`<r:breadcrumb />`, `<r:comment />`, `<r:conte`
- *Tags de controle de fluxo* `<r:if_parent />`,`<r:unless_parent />`, `<r:if_children />`, `<r:unless_children />`, `etc.`
- *Tags que modificam o contexto da página* `<r:page>`, `<r:children>` , `<r:parent>`, `etc.`
- *Tags que trabalham com coleções* `<r:children:each >`

# 8 RefineryCMS

*RefineryCMS é um sistema gerenciador de conteúdos desenvolvido pela Resolve Digital. Ele é um sistema que começou em 2005 e que em meados de 2009 abriu o seu código para a comunidade. Projetado para pequenos projetos onde pessoas possam manter o conteúdo sem muita complexidade. Possui uma interface amigável, simples de se trabalhar, com um editor de conteúdo WYISWYM (What You See Is What You Mean) integrado. Sua instalação é simples e a documentação é pequena porém bem organizada. Ele recentemente foi portado ja para o Ruby on Rails versão 3 e utiliza engines, que são mini aplicações Rails dentro de um projeto Rails, como forma de extensão, facilitando para a comunidade escrever extensões para a ferramenta.*

*O sistema de modelos (templates) do Refinery é feito através de temas, gerados a partir de seu modelo de extensões. Seu ponto forte é que não é necessário aprender uma linguagem específica de domínio como o Radius para efetuar as customizações de layout. Uma desvantagem é que o usuário final pouco pode customizar um layout dentro da interface de administração.*

## 8.1 Características

- *Interface intuitiva, fácil para usuários não técnicos.*

- ***Sistema de extensões baseado em engines, fácil de ser usado, porém não há muitas engines disponíveis.***
- ***Instalação simples***
- ***Documentação bem organizada.***
- ***O foco do usuário final é em somente alterar conteúdo.***

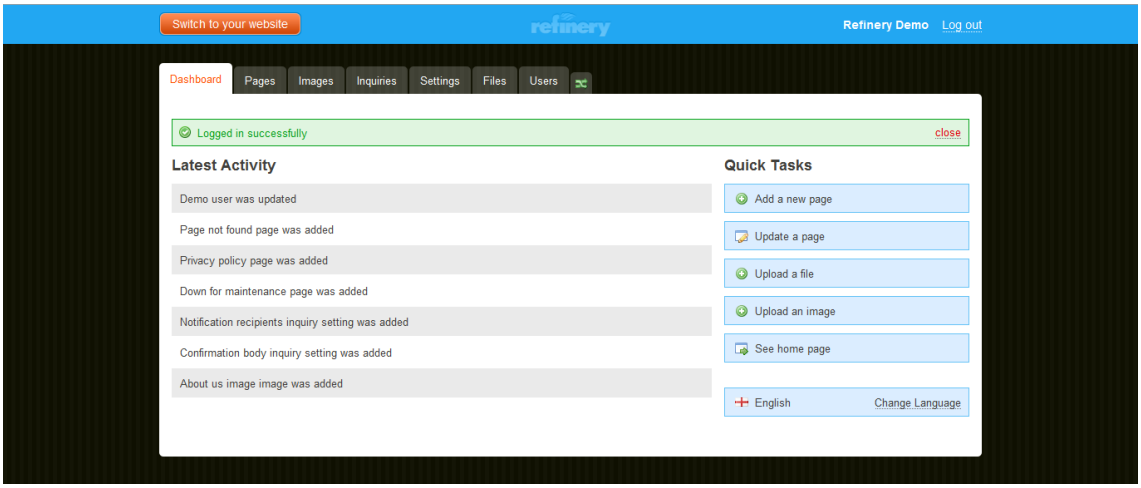


Figure 5: Interface de administração do Refinery


Figure 6: Interface de criação / edição de página do Refinery

# 9 BrowserCMS

***BrowserCMS é um sistema gerenciador de conteúdos desenvolvido pela empresa BrowserMedia, econtra-se na versão 3.1.2. Foi por vários anos um CMS de comercialmente licensiado, escrito na linguagem Java. Motivados pela "onda" de Ruby on Rails, e pela falta de um CMS em Ruby on Rails que suprisse as necessidades de seus clientes, reescreveram o seu CMS em 2009 e deixaram-o com uma licensa de código aberto.***

## 9.1 Características

- ***Sistema voltado ao mundo corporativo para grandes e médias organizações.***
- ***Sistema de workflow completo de publicação.***
- ***Sistema amplo de permissões.***
- ***Versionamento de conteúdo.***
- ***Poucas extensões***
- ***Documentação pequena.***

# 10 Zena

***Zena é um CMS baseado em Ruby on Rails, escrito a partir de 2007 por Gaspard Bucher que inicialmente o desenvolveu par ao seu uso pessoal. Após um certo tempo ele decidiu incrementar a flexibilidade do cms, adicionando uma linguagem de templates, Zafu, que poderia ser usada também por outras pessoas.***

***A linguagem específica de domínio Zafu, tem objetivo semelhante ao Radius, de abstrair e facilitar a inclusão de códigos ruby dentro***

## 10.1 Características

- ***Sistema voltado ao mundo corporativo para grandes e médias organizações.***
- ***Sistema de workflow completo de publicação.***
- ***Sistema amplo de permissões.***
- ***Versionamento de conteúdo.***

- *Poucas extensões*
- *Documentação pequena.*

*Metodologia*

*Este capítulo se dedica a informar quais as serão os métodos de avaliação entre as ferramentas CMS.*

# 11 Inspeção por Checklist

*Este método foi selecionado dentre os apresentados anteriormente, pois pode ser aplicado por um não expert, como o próprio autor nas ferramentas selecionadas.*

*A inspeção adotada é uma checklist mais voltada a websites, adaptadas as necessidades das ferramentas avaliadas. A inspeção foi retirada do site* http://www.userfocus.co.uk/resources/navchecklist.html *e se encontra em anexo.*

# 12 Avaliação da comunidade

*Devido as soluções serem open source, uma análise sobre a comunidade em torno da ferramenta é um aspecto importante na avaliação das ferramentas. Como todas as ferramentas possuem seu código disponibilizado dentro do github.com, o próprio site fornece números interessantes sobre os projetos, adicionados a estes números serão avaliados também numeros da lista de discussão de email e os canais de comunicação do projeto.*

*Os critérios para avaliar serão:*

**Watchers** - *Número de pessoas que acompanham o projeto (novos commits), chamados* watchers

**Forks** - *Número de versões de terceiros em paralelo do projeto (fork).*

**Commits** - *Número de commits no ano de 2010, até 12 de novembro.*

**Contribuidores** - *Número de contribuidores do projeto.*

**Pageviews** - *Número de pageviews na página do projeto entre agosto - novembro 2010.*

**Lista de email** - *Possui lista de discussão por email ?*

**Número pessoas email** - *Número de pessoas cadastradas na lista de discussão por email.*

**Número mensagens lista** - ***Número de mensagens postadas no ano de 2010 na lista de email.***

**Twitter** - ***Possui Twitter ?***

**Seguidores twitter** - ***Quantos seguidores possui no Twitter ?***

**IRC** - ***Possui canal no IRC (Internet Relay Chat) ?***

***Avaliação das ferramentas***

***Este capítulo se dedica a fazer a avaliação entre as ferramentas Radiant, Refinery, BrowserCMS e Zena, conforme os aspectos apresentados anteriormente.***

# 13 Avaliação da Comunidade

***Esta tabela de avaliação se baseia em dados retirados dos repositórios de dados (http://www.github.com), lista de discussão de email, e redes sociais como o Twitter (http://www.twitter.com)***

Table 1: Tabela comparativa da comunidade open source de cada ferramenta

| Critério | Radiant | Refinery | BrowserCMS | Zena |
|---|---|---|---|---|
| Watchers | 1087 | 897 | 711 | 133 |
| Forks | 215 | 219 | 84 | 10 |
| Commits | 399 | 3090 | 99 | 386 |
| Contribuidores | 52 | 58 | 17 | 3 |
| Pageviews | 81942 | 70588 | 8259 | 723 |
| Lista de email | Sim | Sim | Sim | Sim |
| Número pessoas email | 409 | 344 | 369 | 70 |
| Número mensagens lista | 1716 | 1653 | 859 | 140 |
| Twitter | Sim | Sim | Sim | Sim |
| Seguidores twitter | 276 | 45 | 308 | 1750 |
| IRC | Sim | Sim | Sim | Sim |

# 14 Escolha da ferramenta

***Conclusão Análise Crítica***

# References


***Usability engineering methods for software developers.***

***Apostila de engenharia de usabilidade.***

***a.*** *http://ruby.gemstone.com/.*

***b.*** *http://www.cse.chalmers.se/research/group/idc/ituniv/kurser/05/ucd/papers/Maguire%20*

***O que é cms?*** *http://www.navita.com.br/portal/newsroom/definicoes/cms.html.*

***Usability inspection methods, a.***

***b.***

*http://radiantcms.org/.*

*http://rubini.us/.*

***An approach to usability evaluation of e-learning applications.***

***What is a cms?*** *http://www.cmscritic.com/what-is-a-cms/.*

***Bob Boiko.*** **Content Management Bible, 2nd edition.** ***Wiley Publishing, 2005.***

***Michael Fitzgerald.*** **Ruby - Pocket Reference.** ***O'Reilly, 2007.***

***David Flanagan and Yukihiro Matsumoto.*** **The Ruby Programming Language.** ***O'Reilly, 2008.***

***Nirav Mehta.*** **Choosing an Open Source CMS.** ***Packt Publishing, 2009.***

***www.ruby lang.org, 2009.*** *http://www.ruby-lang.org/.*

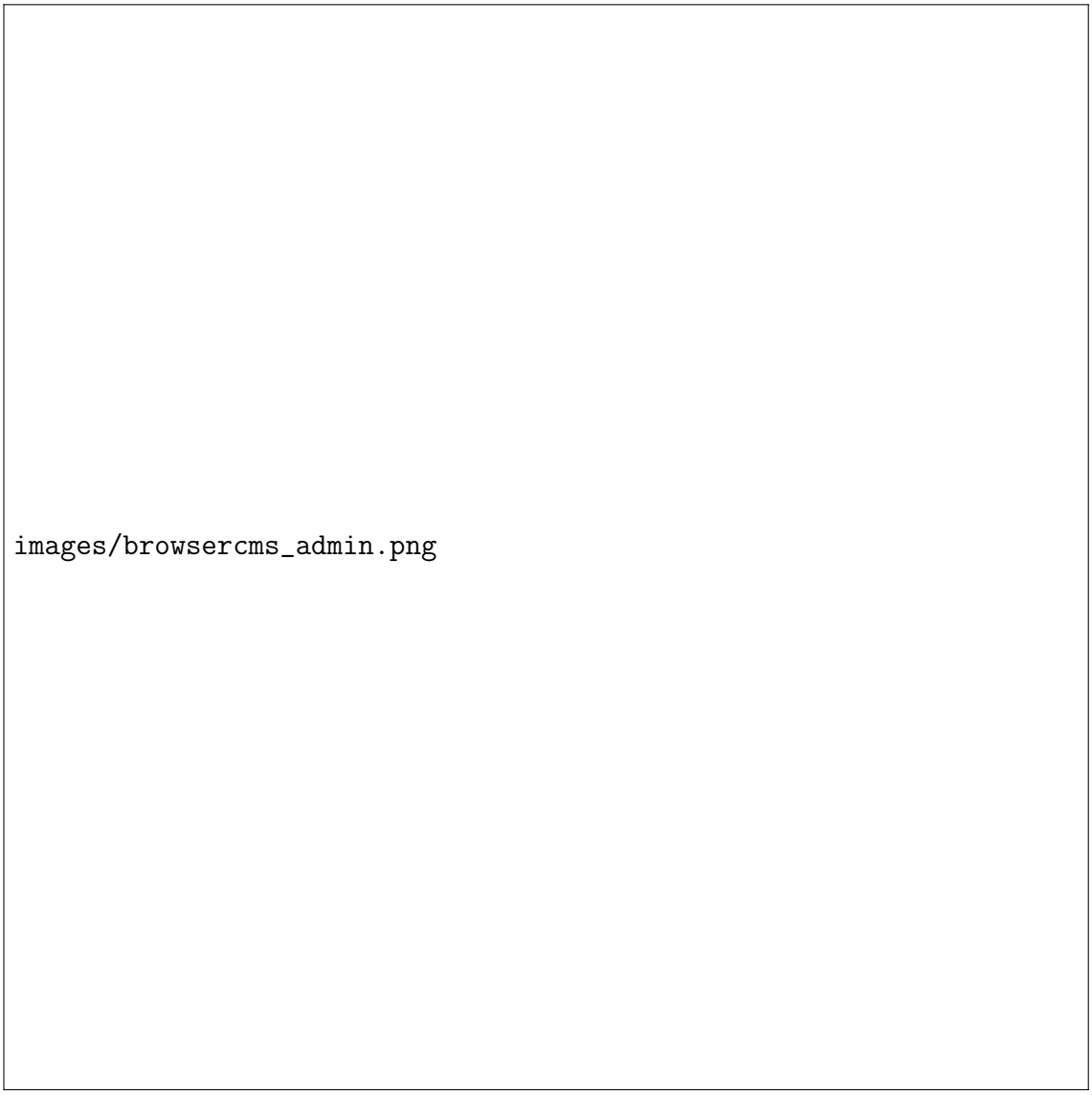


Figure 7: Interface de administração do BrowserCMS

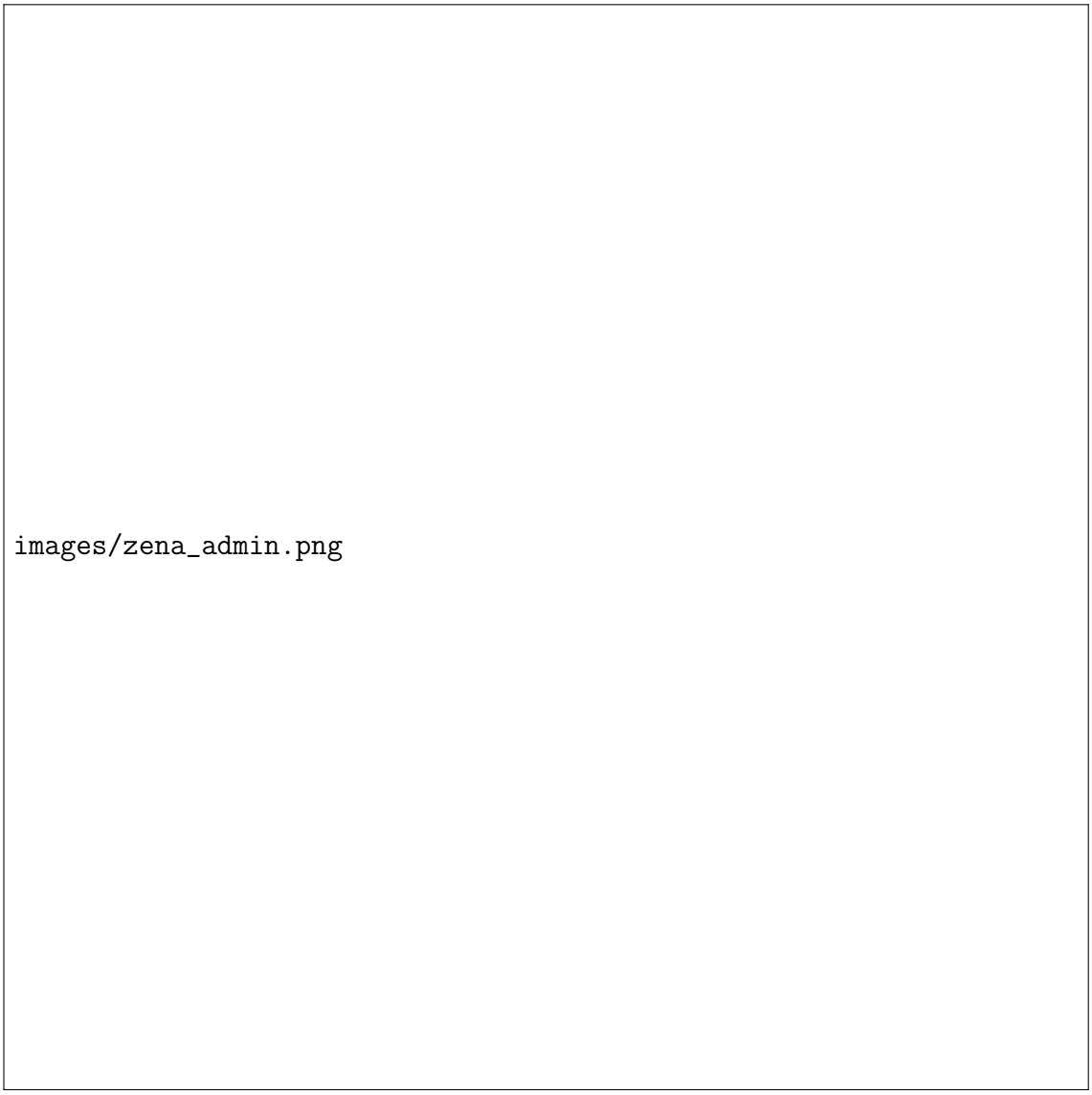


Figure 8: Interface de administração do Zena