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

GUILHERME GREGIANIN TESTA

PrankDev: uma Ferramenta para Criação e Manutenção Dinâmica de *Websites* Baseada em *Templates*

Trabalho de Graduação.

Profa. Dra. Carla Maria Dal Sasso Freitas
Orientadora

Porto Alegre, dezembro de 2010.

AGRADECIMENTOS

Agradeço a todas as pessoas que de algum modo contribuíram para meu processo educativo e na conclusão deste curso de Bacharelado em Ciência da Computação. Em especial:

Meus pais Jacir e Maria Oliva, por sempre incentivarem e valorizarem muito o estudo e atividades culturais.

Meus irmãos Fernando, Maurício, Henrique, Lourenço e Francisco, pelo ambiente contínuo de discussões sadias sobre assuntos aleatórios.

Meus amigos, principalmente o pessoal de Guaporé, que sempre foram uma via de escape do mundo acadêmico para o mundo real.

Minha orientadora, Profa. Dr. Carla Maria Dal Sasso Freitas, pela orientação e por ter apoiado e acreditado no potencial deste trabalho.

A UFRGS e o Instituto de Informática, pelo ensino de qualidade excepcional ao qual tive a oportunidade de ter acesso nestes anos.

A todos, muito obrigado.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
MVC	Model-View-Controller
REST	Representational State Transfer
SGBD	Sistema de Gerenciamento de Banco de Dados
SGML	Standard Generalized Markup Language
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
W3C	World Wide Web Consortium
WWW	World Wide Web

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Devido à popularização da *Internet* e do aumento da acessibilidade aos computadores pessoais, o desenvolvimento *web* entrou em uma fase de grande expansão há alguns anos. Empresas dos mais variados portes e ramos de negócio estão investindo cada vez mais em aplicações e serviços disponibilizados na *Internet*, e usuários procuram a rede como meio de divulgação de ideias e trabalhos pessoais. Como conseqüência, existe um crescimento na demanda por sistemas e serviços que facilitem o desenvolvimento de *websites* na rede mundial de computadores.

Seguindo este aumento, este trabalho apresenta um sistema protótipo que oferece um serviço para criação e manutenção de *sites* através da aplicação de regras de transformação em *templates* pré-definidos, utilizando seletores CSS. O objetivo deste serviço é oferecer um meio simples mas eficaz para que qualquer usuário da Internet seja capaz de manter uma página de qualidade na rede.

Através da definição destas regras, as páginas HTML são modificadas por meio de um *parser* escrito na linguagem Ruby, chamado Nokogiri. Com este analisador, foi possível criar uma série de métodos para a modificação dos arquivos HTML, utilizando seletores CSS como meio de intercomunicação com o usuário.

Este trabalho apresenta os tópicos teóricos envolvidos no processo de análise de documentos HTML, e detalhes sobre a solução implementada com o *framework* de desenvolvimento *web* Ruby on Rails.

Palavras-Chave: HTML, CSS, Ruby on Rails, Internet, desenvolvimento *web*.

PrankDev: A Tool for Creating and Maintaining Dynamically Websites Based on Templates

ABSTRACT

Due to the popularization of the Internet and increased access to personal computers, the web development has entered in a phase of great expansion in the last few years. Companies of all sizes and kinds of businesses are increasingly investing in applications and services available on the Internet, and users look for the network as a place for disseminating ideas and personal work. As a consequence, there is a growing demand for systems and services that helps the development of websites on the World Wide Web.

Following this increase, this paper presents a prototype system that provides a service for creating and maintaining websites through the application of transformation rules in predefined templates, using CSS selectors. This service aims to offer a simple but effective way for the common user to maintain a page on the network.

By defining these rules, HTML pages are modified through an parser written in the Ruby language called Nokogiri. With this analyzer, it was possible to create a series of methods for modifying the HTML files using CSS selectors as a means of communicating with the user.

This paper presents the theoretical topics involved in the process of analysis of HTML documents, and details about the implemented solution with the web development framework Ruby on Rails.

Keywords: HTML, CSS, Ruby on Rails, Internet, web design.

1 INTRODUÇÃO

A grande expansão da *Internet* como meio de comunicação e divulgação de informações criou uma forte demanda por ferramentas que auxiliassem o processo de criação e manutenção de *websites*. Embora novos instrumentos para o desenvolvimento de páginas *web* tenham sido criados e aperfeiçoados, seu uso ainda é restrito a pessoas com um conhecimento técnico mais qualificado. Para uma inclusão digital mais profunda, ainda existe uma necessidade de facilitar a criação de *sites* para usuários comuns.

Páginas *web* de qualidade seguem uma série de boas práticas, que são resultado da contínua evolução no desenvolvimento voltado à *Internet*. Estas práticas foram se delineando com o passar do tempo e muitas delas tornaram-se unanimidade, sendo uma orientação no desenvolvimento de *websites* hoje. A partir do surgimento da primeira versão da CSS em 1996[1], a separação da parte estrutural (HTML) da parte de apresentação (CSS) na construção de *sites* tornou-se um destes padrões.

Graças a estes padrões de construção de páginas para a Internet, este trabalho apresenta **PrankDev**: um sistema *web* que oferece um serviço de criação e manutenção de *websites* baseado em *templates*. A partir da escolha ou *upload* de um *template*, o usuário cria regras simples baseadas em seletores CSS para alterar, criar ou excluir o conteúdo deste modelo, preservando sua estrutura.

Prank, em inglês, significa brincadeira, e este nome reflete o espírito que o usuário deve ter ao utilizar o sistema: desenvolver um *site* de maneira fácil, simples e divertida.

Este sistema protótipo foi desenvolvido com o uso da linguagem livre Ruby[2] e do *framework* de desenvolvimento *web* Ruby on Rails[3], que por sua vez é estruturado no paradigma MVC. O sistema ainda faz uso do *parser* Nokogiri[4] para analisar e modificar os elementos dos arquivos HTML.

1.1 Motivação

Apesar de existirem meios gratuitos para manter uma página na Internet disponível hoje, de maneira geral estes oferecem um serviço consideravelmente limitado, restringindo seu uso na forma de *blogs* ou ferramentas complicadas. Pequenas empresas ou pessoas que desejam divulgar seus produtos, serviços ou alguma maneira mais

[1] http://www.w3.org/TR/CSS1/

[2] http://www.ruby-lang.org/

[3] http://rubyonrails.org/

[4] http://nokogiri.org/

sofisticada de propagar informação, se sentem obrigadas a contratar pessoal qualificado, custeando a diferença de qualidade visual.

Por outro lado, a própria *Internet* oferece milhares de *templates* CSS livres, de qualidade[5], dos mais variados estilos e formas para serem utilizados para qualquer fim responsável. Estes modelos podem ser aproveitados para ajudar na inclusão do usuário comum como ator ativo na *Internet*, onde possa criar e divulgar informação em um meio de qualidade, sem ter que pagar um preço alto por isso.

Nos capítulos seguintes, é apresentada uma forma de aproveitar esta oferta de modelos livres que existe hoje de uma maneira simples e objetiva para o usuário final, facilitando a tarefa de manter um *website* de qualidade na rede mundial de computadores.

1.2 Objetivos

O objetivo deste trabalho é apresentar um roteiro para a manipulação de arquivos HTML utilizando um *parser* de forma automatizada a partir de seletores CSS, abordando os principais conceitos relacionados a este processo e detalhando pontos da implementação da solução proposta.

1.3 Organização do Trabalho

Este trabalho está organizado em capítulos da seguinte forma:

- O Capítulo 2, **Conceitos Básicos** (página 13), introduz a ferramenta implementada e aborda os principais conceitos sobre manipulação de arquivos HTML, seletores CSS e o desenvolvimento de *websites* de qualidade.
- O Capítulo 3, **Tecnologias** (página 21), apresenta uma base conceitual sobre as tecnologias utilizadas na implementação do sistema.
- O Capítulo 4**, Arquitetura do Sistema** (página 26), mostra uma visão geral da arquitetura e do funcionamento da solução proposta.
- O Capítulo 5, **PrankDev em Detalhes** (página 32), apresenta uma descrição detalhada dos aspectos mais importantes da implementação.
- O Capítulo 6, **Estudo de Caso** (página 38), expõe um exemplo prático de utilização do serviço e do funcionamento do sistema.
- O Capítulo 7, **Conclusão** (página 43), reúne conclusões relativas a este trabalho e apresenta sugestões de melhorias e incrementos na solução apresentada.

[5] Existem práticas regulamentadas por entidades como a W3C para a construção de *Websites* de qualidade, e ferramentas para sua validação. O Capítulo 2 aborda estas questões com maiores detalhes.

2 CONCEITOS BÁSICOS

A criação de *websites* hoje agrega diversos conceitos e tecnologias resultantes do processo natural de evolução do desenvolvimento *web* ao longo dos anos. O entendimento destes fundamentos é chave para a geração de *sites* de qualidade. O sistema protótipo apresentado neste trabalho depende diretamente destes indicadores de qualidade, uma vez que modifica a estrutura de páginas já existentes.

Mesmo que seja possível elaborar páginas para a *web* de baixa complexidade sem um conhecimento muito profundo sobre as tecnologias envolvidas, erros de construção e má estruturação no código podem comprometer o sucesso do *website*. Arquivos HTML descritos fora do padrão recomendado pela W3C (*World Wide Web Consortium*), correm o risco de não serem visualizados corretamente em *browsers* distintos, ou até mesmo em sistemas operacionais diferentes. Isto acontece porque os navegadores implementam uma especificação da W3C que descreve como estes devem interpretar a estrutura das *tags* HTML, e portanto não existem garantias de que um documento fora dos padrões seja exibido da maneira correta.

Por isso, para que a modificação dinâmica destes arquivos exiba seu conteúdo da forma desejada (que é o principal conceito discutido neste trabalho), é imprescindível que estes documentos estejam dentro do padrão recomendado pela W3C. E desejável que conservem boas práticas de programação[6]. Felizmente, há uma preocupação maior neste sentido nos últimos anos. Desenvolvedores já elaboram *websites* validando periodicamente seu trabalho, através de serviços disponíveis gratuitamente para tal fim, como, por exemplo, a ferramenta da própria W3C[7] (FREEMAN, 2008).

Nas seções que seguem deste capítulo, o propósito e funcionamento da ferramenta desenvolvida é introduzido, e os principais conceitos por trás da manipulação HTML são abordados.

2.1 A Ferramenta PrankDev

PrankDev é uma proposta de um serviço *web* para criação de *websites* a partir de *templates*. Ou seja, o usuário pode escolher um modelo que atenda seus requisitos pessoais e insere conteúdo a este modelo de modo a transformá-lo em seu *site*. Este serviço fornece uma estrutura para que o usuário consiga gerenciar os arquivos que

[6] Apesar de HTML ser uma linguagem de propósito específico, o termo foi utilizado aqui por fazer referência à qualidade do código em questão.

[7] http://validator.w3.org/

compõe seu *website*, como imagens e páginas HTML, assim como permite hospedá-lo em seu endereço público dentro do PrankDev.

Para modificar o modelo escolhido, o usuário utiliza uma abstração chamada regra. Uma regra é uma definição de tranformação de conteúdo, ou seja, insere, modifica ou exclui textos ou imagens do *template*. Uma regra pode alterar o nome de um título, inserir uma imagem ou parágrafo, excluir um elemento de uma lista, entre outros. Portanto cada ação sobre o modelo corresponde a uma regra.

As regras são definidas em um formulário simples, com quatro campos: "Seletor CSS", "Novo texto", "Link" e "Outras Opções". Com a combinação de preenchimento destes campos, é possível realizar as diversas ações de modificação ou transformação do *template* escolhido. Estes campos, e as correspondentes regras criadas a partir de seu preenchimento serão detalhadas no Capítulo 5.

Para cada página que compõe o *website* do usuário, é possível inserir quantas regras forem necessárias para transformá-la na página desejada. Estas modificações a partir das regras são realizadas sempre no documento HTML e não no arquivo de estilo CSS. Desta forma, a estrutura visual do *template* escolhido é sempre mantida.

Nas próximas seções deste capítulo, serão detalhados os principais conceitos por trás da manipulação de documentos HTML, de seletores CSS e sobre construção de *websites* de qualidade.

2.2 HTML

O HTML foi criado pelo renomado diretor da W3C, Tim Berners-Lee, mentor de outros projetos importantes para a *web* como o HTTP (*HiperText Transfer Protocol*) e a WWW (*World Wide Web*) e teve como objetivo inicial ser uma linguagem para comunicação e divulgação de informações entre seu grupo de pesquisa (REGGETT, 1998).

O HTML é uma linguagem de marcação baseada na SGML (*Standard Generalized Markup Language*) que acabou se definindo como a linguagem de descrição de documentos na *Internet*. Em 2000, o HTML tornou-se uma norma ISO/IEC[8] (ISO/IEC 15445:2000). A última versão oficial foi a 4.01, lançada em 1999 pela W3C, mas em 2008 surgiu a versão 5 experimental, que deverá se tornar um padrão definitivo em breve[9], e representa a maior revisão da linguagem desde sua criação, em 1991[10].

Um documento HTML nada mais é do que um texto com uma série de *tags* especiais (etiquetas) de marcação, que estruturam o texto e lhe dá um significado adicional à sua interpretação. De maneira geral, existem dois tipos básicos de *tags* HTML:

- Etiquetas que adicionam informação de estrutura ou formatação a um determinado texto. Por exemplo, `<p>Algum texto</p>`, que indica que `algum texto` é um parágrafo; e `<b>Ruby</b>`, que formata em negrito o texto `Ruby`.

[8] http://www.scss.tcd.ie/misc/15445/15445.HTML

[9] http://wiki.whatwg.org/wiki/FAQ#When_will_HTML5_be_finished.3F

[10] http://lists.w3.org/Archives/Public/www-talk/1991SepOct/0003.html

- Etiquetas com um significado próprio, como `<br>`, que indica uma quebra de linha; e `<img src="images/my_image.jpg">`, que adiciona a imagem `my_image.jpg` que se encontra no diretório `images` ao documento.

Esta marcação de etiquetas segue uma estrutura em árvore, onde as *tags* obedecem uma certa hierarquia. Todo documento HTML possui um cabeçalho (*head*), onde se define algumas especificações sobre o documento em questão; e um corpo (*body*), onde se encontra o conteúdo a ser exibido.

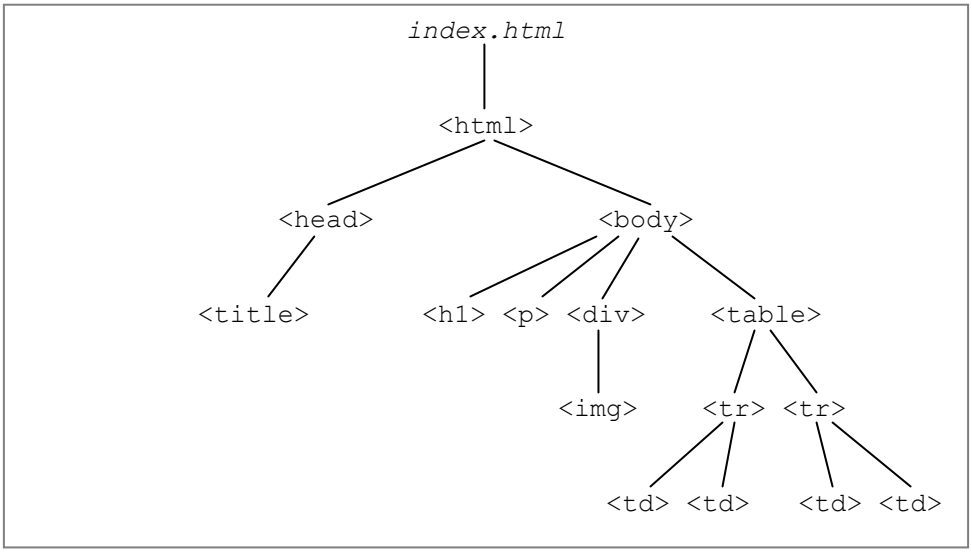

Figura 2.1: Exemplo de estruturação em árvore de um arquivo HTML.

Na Figura 2.1: Exemplo de estruturação em árvore de um arquivo HTML.Figura 2.1 pode se observar como o arquivo hipotético index.html se comporta estruturalmente. O elemento raiz `<html>` possui dois filhos: `<head>`, que contém o título da página; e `<body>`, que por sua vez contém um título principal, um parágrafo, uma imagem (a etiqueta `<div>` é utilizada para demarcação de estilo) e uma tabela com duas colunas e duas linhas (cada elemento `<td>` representa uma célula na tabela).

A manipulação dos arquivos HTML pelo PrankDev atua diretamente sobre esta construção em árvore, mapeando cada *tag* para um tipo estruturado específico chamado *node*, que faz parte da biblioteca Nokogiri, mantendo a formação da árvore original. As modificações portanto são feitas nesta árvore de *nodes*, dinamicamente. Mais detalhes serão discutidos no Capítulo 3 deste trabalho.

2.3 XHTML

Um documento XHTML (o X provém de *eXtensible*) nada mais é do que um arquivo HTML com algumas características particulares, que o tornam também um documento XML (*Extensible Markup Language*). XML também foi baseado na SGML e é uma recomendação da W3C. É uma linguagem de marcação de dados largamente utilizada no desenvolvimento de *software* atualmente.

O XML fornece um padrão para a descrição de dados de forma semelhante ao HTML. Sua principal diferença é que o XML é genérica, tendo um número infinito de

etiquetas que servem para delinear uma informação, e não para formatar um documento de forma específica. Cabe ao *software* que analisa o arquivo XML interpretar as *tags* e processar seu conteúdo, que pode mudar conforme o propósito pretendido.

Ao transformarmos um documento HTML em XHTML, estamos anunciando que tal documento é simultaneamente XML e HTML. Para tal, devemos tomar algumas alterações e cuidados, como descritos na Tabela 2.1[11], abaixo.

Tabela 2.1: Principais características e diferenças entre arquivos XHTML e HTML.

Característica	Observação	Exemplo

Documentos devem ser aninhados corretamente	Faz referência ao ordenamento das *tags*: uma etiqueta aberta antes de outra deve ser fechada depois.	`<div>`
		`  <p><b>Algum texto</b></p>`
		`</div>`

Todas as *tags* devem ser fechadas	Alguns elementos do HTML original podem ser utilizados sozinhos, como ` `.	` ` `<img src="my_image.jpg" />`

As *tags* devem ser escritas em letras minúsculas	XML é *case sensitive* (sensível ao tamanho de caixa), portanto os elementos devem ser escritos em minúsculas.	`<head>`
		`  <title>`
		`  Título da Página`
		`  </title>`
		`</head>`

O documento deve possuir um único elemento raiz	No HTML, esta característica não é uma exigência, mesmo que seja desaconselhável.	`<html>`
		`  <head>`
		`  <title>...</title>`
		`  </head>`
		`  <body>`
		`  ...`
		`  </body>`
		`</html>`

Além destas diferenças estruturais, arquivos XHTML devem, obrigatoriamente conter os elementos `<html>`, `<head>`, `<title>` e `<body>`. Existe ainda um elemento diferenciado, que é o `DOCTYPE`. O `DOCTYPE`, apesar de estar sempre presente no arquivo, não é parte da definição do XHTML. Ele indica ao *browser* qual linguagem de marcação está sendo empregada no documento em questão, além de uma série de atributos que definem entidades relacionadas ao emprego da linguagem (SILVA, 2008).

2.4 CSS

CSS (*Cascading Style Sheet*, ou, em português, folhas de estilo em cascata) surgiu em 1994 como resultado dos estudos de Håkon Wium Lie e Bert Bos na área de especificação de documentos para a *web*. Seu objetivo era ser um meio de definição de estilo ou apresentação de documentos HTML. Em 1996, foi lançada a versão *CSS Level 1* como uma recomendação da W3C (LIE; BOSS, 2005)

[11] http://www.w3schools.com/xhtml/xhtml_html.asp

A CSS define, através de uma série de regras, como um documento HTML deve se apresentar. Desta forma, separa-se o conteúdo e estrutura da página em documentos HTML e sua apresentação visual em CSS. Esta diferenciação traz alguns benefícios significativos, tornando a *interface* com o usuário mais poderosa e flexível.

A Figura 2.2 mostra a sintaxe de uma regra CSS. Cada regra é composta por um seletor e sua declaração. O seletor identifica os elementos no arquivo HTML aos quais serão aplicadas as propriedades relacionadas na parte de declaração. O funcionamento dos seletores CSS está explicado na seção 2.4.1[12].

Figura 2.2: Sintaxe da regra CSS (SILVA, 2008. p. 51)

A separação já mencionada entre HTML e CSS é um aspecto de grande importância no contexto deste trabalho. O sistema PrankDev proporciona a modificação de páginas HTML preservando seu aspecto visual, e isto pode ser feito porque o arquivo de estilo CSS é preservado. Ou seja, ao alterarmos o conteúdo HTML do site, este continuará sendo apresentado de acordo com a definição visual do arquivo CSS do *template*.

2.4.1 Seletores CSS

Seletores CSS são uma poderosa ferramenta para identificar elementos HTML de forma individual ou coletiva. Com eles, é possível restringir a aplicação de uma regra definida na declaração a somente alguns elementos da página alvo. A sintaxe dos seletores CSS possíveis é extensa, e existem muitos recursos específicos. Apresentaremos apenas os três tipos principais para ilustração: o seletor de etiqueta, o seletor de classe, e o seletor identificador.

O **seletor de etiqueta** casa todas as tags HTML do arquivo do tipo especificado, independente de sua classe ou identificador. Considere a seguinte regra:

```
p { color: blue; }
```

Esta regra aplica a cor azul a todos os parágrafos do documento. Ou seja, a todo o texto contido entre as *tags* `<p>` e `</p>`.

O **seletor de classe** casa todos os elementos do documento que contenham o atributo `class` especificado. Por exemplo, a regra a seguir:

```
.number { font-style: italic; }
```

[12] O aspecto técnico da linguagem e funcionamento da CSS não é abordado profundamente neste trabalho pois o objetivo aqui é apenas a forma como ela interage com o HTML. Para maiores detalhes sobre CSS, é sugerido o conceituado livro *Cascading Style Sheets: designing for web*, escrito pelos próprios criadores da tecnologia, Håkon Wium Lie e Bert Bos (LIE; BOSS, 2005).

Esta regra aplicará o estilo de fonte itálico a todas as *tags* que possuírem o atributo `class` igual a `number`, independente do nome da etiqueta. Observe que o seletor inicia com um ponto final. Desta forma, um elemento `<p class="number">1</p>` será afetado assim como `<div class="number">2</div>`. E em ambos os casos seus conteúdos serão formatados em itálico.

Por último, o **seletor identificador** casa todas as etiquetas que contiverem o atributo `id` especificado. Este tipo de seletor inicia sempre com o caractere "#". Por exemplo:

```
#subtitle { font-family: sans-serif; }
```

A regra acima irá aplicar a fonte *sans-serif* aos itens que contiverem o atributo `id` igual a `subtitle`. A ideia deste atributo é que ele seja único no documento, identificando singularmente um elemento (e isso é inclusive uma recomendação da W3C). Neste caso, por exemplo, um elemento como `<h3 id="subtitle">Pages</h3>` seria afetado.

Seletores CSS são utilizados pelo sistema proposto como meio de identificação dos elementos HTML para aplicação de regras. Uma determinada regra de uma página é definida com uma série de atributos, e o seletor CSS é utilizado para filtrar sua aplicação no *template* (este funcionamento será detalhado no Capítulo 4).

2.5 *Web Standards*

Web Standards (ou Padrões Web) é um termo genérico mas que se refere a uma série de normas para a construção de *websites* de qualidade. Estes padrões, especialmente os recomendados pela W3C[13], contribuem de maneira determinante para o bom funcionamento do modelo de transformação de documentos proposto neste trabalho. Na verdade, o próprio desenvolvimento do PrankDev foi baseado na premissa de que os *templates* utilizados pelos usuários do sistema seriam bem formados.

Um padrão muito importante diz respeito à separação do conteúdo das páginas em arquivos HTML da apresentação, em CSS. Assim toda a parte estrutural e o conteúdo propriamente dito fica isolado das definições visuais e de exibição. Isto facilita muito a manutenção de *sites*, já que as preocupações sobre o conteúdo da página se isolam das questões de *interface*.

Esta norma mencionada é crucial para o bom funcionamento do sistema proposto. Um usuário do PrankDev que cria sua página através de um *template* que não segue *web standards*, corre o risco de ter vários problemas durante sua transformação. Felizmente, existem na *web* inúmeros *sites* que oferecem modelos de qualidade, que seguem estas normas. Como o Free CSS[14] e o TemplateMo[15], dois bons repositórios de *templates* gratuitos para *download*.

Existem ainda ferramentas para validação de páginas, para verificar se as mesmas seguem os padrões *web standards*. Este é um bom meio quando se deseja criar um *website* do zero, analisando periodicamente os documentos à medida que estes são criados e incrementados. Isto fornece um maior controle e garante um bom nível de qualidade, já que estes meios de validação indicam o que deve ser modificado em caso

[13] http://www.w3.org/standards/webdesign/

[14] http://www.free-css.com

[15] http://www.templatemo.com

de erro. A principal ferramenta de validação utilizada atualmente é aquela oferecida pela própria W3C em seu site[16], mas existem outros disponíveis na *web*.

2.6 Serviços existentes

Existem alguns serviços de criação de *websites* disponíveis na Internet atualmente, e eles se dividem basicamente em dois tipos. O primeiro, permite a construção de *sites* mais completos, inclusive com opções de criação de formulários. A segunda categoria abrange os serviços de disponibilização simples de informação, como os *blogs*. Estes últimos são consideravelmente mais limitados, mas de simples utilização.

Dentre os serviços mais populares e gratuitos de criação de *websites* completos estão o Webnode[17] e o Moogo[18], que oferecem uma série de ferramentas para o auxílio na construção de páginas. As operações são realizadas através de um painel de controle que concentra as opções de edição. Estes serviços permitem a utilização de *templates* pré-definidos disponíveis internamente.

Uma desvantagem deste tipo de serviço é que, por abranger várias opções de edição, sua utilização é mais complexa, sendo necessário um tempo de aprendizado. Além disso, a inserção de conteúdo é possível somente nos *templates* disponíveis internamente, provavelmente para poder oferecer um número maior de possibilidades de edição. O problema neste caso é que, visualmente, estes modelos de *websites* são inferiores aos modelos genéricos disponíveis na *web*.

A outra categoria de *sites* são os *blogs*, onde podemos citar o Blogger[19], serviço oferecido pelo Google. Este tipo de serviço é um sistema simples de divulgação de informações na *web*, que segue uma ordem cronológica de publicação de artigos (também chamados de *posts*). Essa categoria de *site* é interessante para usuários comuns, mas não se adequa à empresas ou pessoas que pretendem oferecer algo mais profissional ou com um visual mais sofisticado (mesmo que várias empresas também mantenham *blogs*). Isso acontece tipicamente pelo fato dos *blogs* seguirem um padrão de estrutura simples, normalmente de uma página só. Sua principal vantagem é a facilidade de uso.

Alternativamente aos serviços disponibilizados na Internet para construção de *websites*, existem *softwares desktops* conhecidos que também visam a criação de páginas destinadas à rede como, por exemplo, o FrontPage, da Microsoft[20], e o Adobe Dreamweaver[21]. Estes programas são bastante completos e flexíveis, e permitem a criação de páginas muito complexas, sendo utilizados profissionalmente na criação de *sites*. Sua desvantagem está justamente no fato que são difíceis de serem utilizados. Existe um grande número de usuários que encontram muitas dificuldades no aprendizado de ferramentas completas como estas. E justamente por isso acabam contratando e pagando pessoal especializado para realizar estas tarefas.

---

[16] http://validator.w3.org/

[17] http://www.webnode.com/

[18] http://www.moogo.com/

[19] http://www.blogger.com/

[20] http://office.microsoft.com/en-us/frontpage-help/

[21] http://www.adobe.com/products/dreamweaver/

O modelo de serviço implementado neste trabalho pretende oferecer uma ferramenta que alia a facilidade de criação dos *blogs* com *templates* profissionais para criar *websites* completos e visualmente interessantes. Nos próximos capítulos será detalhado como esta construção é possível.

3 TECNOLOGIAS

Para o processo de seleção das tecnologias utilizadas no desenvolvimento do sistema proposto, alguns critérios prévios foram determinados. Primeiramente, como o código do PrankDev seria disponibilizado na Internet[22], e para incentivar o desenvolvimento de ferramentas livres, foi estabelecido que *website* também seria totalmente implementado com uso de tecnologias de código aberto. Assim, o sistema não estaria ligado de forma vital a nenhuma ferramenta proprietária, o que poderia atrapalhar o processo de cooperação desejável em projetos de *software* livre.

Com esta presunção, procurou-se por tecnologias livres que buscassem facilitar a criação de sistemas voltados para a *web* baseados em banco de dados. Nesses termos, a opção de se trabalhar com o *framework* de desenvolvimento *web* Ruby on Rails se mostrou bastante promissora. Este ambiente de produção de *software* para a *web* foi criado sobre a linguagem Ruby, que também é livre.

Uma vez determinada a linguagem e o modelo de desenvolvimento do sistema, observou-se a necessidade de uma biblioteca em Ruby que auxiliasse no processo de manipulação de arquivos HTML. Entre as propostas existentes hoje, destacou-se o *parser* Nokogiri, que fornece uma série de métodos para modificar arquivos HTML, XML, SAX e Reader[23], e que também possui código aberto. Mais informações estão detalhas no item 3.2, na página 24.

3.1 Ruby

Ruby é uma linguagem de programação livre desenvolvida por Yukihiro Matsumoto em 1995 com o objetivo inicial de ser usada como uma linguagem de *script*. Ruby é multiparadigma, já que permite programação orientada a objetos, imperativa e funcional. Duas características marcantes desta linguagem são a flexibilidade e simplicidade, já que provê soluções elegantes para os problemas mais recorrentes de programação, tornando-se bastante intuitiva.

Ruby foi implementada sobre a linguagem de programação C, e seu propósito era o de integrar o melhor que as linguagens Perl, Smalltalk, Eiffel, Ada, e Lisp ofereciam de forma natural, que a aproximasse da linguagem humana[24]. Nas palavras de seu criador,

[22] http://github.com/prankdev/ggtesta

[23] http://nokogiri.org/

[24] http://www.ruby-lang.org/en/about/

"*O Ruby é simples na aparência, mas muito complexo no interior, tal como o corpo humano*" [25].

A escolha desta linguagem para a criação do sistema PrankDev está ligada principalmente ao *framework* de desenvolvimento *web* Ruby on Rails, e no suporte que este oferece para aplicações baseadas em banco de dados. A propósito, a popularidade do Ruby deve-se muito pelo sucesso desta plataforma, lançada na sua primeira versão em 2005 (FITZGERALD, 2007).

3.1.1 Ruby on Rails

Ruby on Rails (ou somente Rails) é um *framework* de desenvolvimento de sistemas *web* baseados em banco de dados de código aberto, escrito na linguagem Ruby. Rails permite criar sistemas completos em um tempo reduzido, sem o uso de grandes ferramentas de auxílio à programação. Como o próprio *site* da plataforma cita[26]:

> Ruby on Rails é um framework de desenvolvimento web (gratuito e de código aberto) otimizado para a produtividade sustentável e a diversão do programador. Ele permite que você escreva código de forma elegante, favorecendo a convenção ao invés da configuração.

Este *framework* foi criado ao final de 2004 por David Heinemeier Hansson, e em pouquíssimo tempo conquistou um considerável espaço no mundo do desenvolvimento *web*. Os três princípios básicos mais importantes que impulsionaram este sucesso são (ORSINI, 2007. p. 1):

- **Convenção sobre configuração**: em aplicações Rails, utiliza-se uma série de convenções de nomenclatura que facilitam o entendimento e a conexão entre as diferentes partes do sistema, dispensando o uso de extensos arquivos de configuração.
- **Uso de geração de código**: o *framework* oferece um suporte para geração de código inicial funcional, auxiliando o programador na implementação das partes fundamentais do sistema.
- **Don't Repeat Yourself** (DRY, em português: não se repita): o código dentro do sistema não deve se repetir. Cada módulo possui um lugar bem definido e é importante o reuso de partes compartilhadas.

Rails está sendo empregado em diversos *websites* de reconhecimento mundial. Dentre muitos podemos citar o repositório GitHub, baseado no sistema de controle de versões Git, e o portal de compras Groupon, presente em inúmeros países[27].

Todo sistema criado utilizando esta tecnologia é baseado no padrão MVC (*Model*, *View*, *Controller*. Ou, modelo, visão e controlador), que segue algumas convenções de modularidade. Além disso, grande parte do código básico (que serve para a grande maioria das aplicações), é gerado automaticamente pela plataforma a partir dos dados do banco de dados.

Aplicações Rails são REST (Representational State Transfer), que se refere a um comportamento do sistema onde não há troca de mensagens a não ser as definitas pelo

[25] http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/2773

[26] http://www.rubyonrails.pro.br

[27] http://rubyonrails.org/

HTTP. Ou seja, um usuário do sistema consegue navegar pelo site escolhendo os links apropriados e recebendo uma nova página como resposta. Cada página corresponde a um estado específico na aplicação e pode ser acessado de forma independente.

Rails ainda oferece suporte a migrações. Uma migração é uma abstração de nível mais alto para o banco de dados que permite definir as entidades e atributos em Ruby que irão gerar as tabelas automaticamente. Desta forma, a aplicação torna-se independente do banco de dados, que pode ser substituído conforme a conveniência. Neste trabalho, foi utilizado o SQLite3, que é um SGBD (Sistema de Gerenciamento de Banco de Dados) simples escrito em C, de código aberto, utilizado como padrão nas aplicações Rails. O SQLite3 é parte integrante do sistema o qual está sendo utilizado, e não um processo separado.

3.1.1.1 A linguagem ERB

A linguagem ERB é uma linguagem de *template* definida em Ruby. Ela é muito utilizada em aplicações Rails conjuntamente com HTML para gerar informações dinâmicas de visualização, ou para controle de fluxo[28]. Tipicamente, após o processamento de uma requisição por parte do sistema, um documento HTML puro é gerado a partir da definição de uma *interface*, escrita com ERB e HTML.

Na verdade ela estabelece um meio de acesso para se utilizar Ruby dentro de arquivos HTML. Um documento com ERB recebe uma extensão adicional ".erb" ao seu nome, identificando que o arquivo utiliza a linguagem. Assim, por exemplo, um arquivo index.html.erb, ao ser processado pela aplicação Rails, terá seu conteúdo interpretado e convertido para HTML, e será gerado dinamicamente um arquivo index.html.

Todo o conteúdo entre os demarcadores especiais `<%` ou `<%=` e `%>` dentro deste arquivo faz parte da sintaxe Ruby e será compilado como tal. O código que inicia com o demarcador `<%` é interpretado sem gerar uma saída direta no arquivo HTML, e o código que segue o demarcador `<%=` é convertido para a saída, no caso, o arquivo HTML puro. O código a seguir ilustra este funcionamento:

```
<ul>
  <% @pages.each begin |page| %>
    <li><%= page.title %></li>
  <% end %>
</ul>
```

Neste caso, será criada uma lista com os títulos de um conjunto de páginas. Considerando que `@pages` seja um array de objetos `Page`, onde cada instância contém um atributo `title`, a saída deste exemplo hipotético para os títulos: "UFRGS", "Instituto de Informática" e "Ruby on Rails", seria:

- UFRGS
- Instituto de Informática
- Ruby on Rails

[28] Ela pode ser utilizada com outras linguagens além do HTML, mas tal assunto foge do escopo deste trabalho e não será expandido.

Todas as *views* do sistema PrankDev foram definidas com a linguagem ERB, assim como os *layouts* (*templates* gerais do *site*).

3.2 Nokogiri

Nokogiri é um *parser* para arquivos HTML e XML (entre outros), escrito em Ruby e de código aberto. Ele converte o documento sendo analisado para uma estrutura de dados que facilita sua modificação. Além disso, é possível procurar por elementos específicos através de seletores CSS e XPath.

3.2.1 Funcionamento

Ao analisar um documento HTML, Nokogiri converte o arquivo para XML e o percorre criando dinamicamente uma árvore que contém a composição do documento analisado. Cada nó desta árvore é uma instância da classe `Node`, que é uma estrutura de dados especial correspondente a cada elemento do documento HTML. Estes nós podem ser acessados de meneira similar a uma tabela *Hash* com múltiplas chaves.

Para ilustrar este mapeamento, considere uma *tag* em um documento HTML como a que segue:

```
<a href="#foo" id="link">link</a>
```
[29]

Neste exemplo, ao percorrer o documento, o *parser* irá criar um nó na árvore onde o atributo `node.name` é igual a `'a'`, e suas chaves podem ser `'href'` e `'link'` (atributo), assim como `'#foo'` e `'id'`. Ou seja, ao ser feita uma referência por qualquer uma destas chaves (que são os atributos da etiqueta), este nó será retornado. Este funcionamento é semelhante ao de uma tabela *hash*, onde é possível fazer referências através de chaves com nomes definidos.

3.2.2 Effigy

Effigy[30] é uma biblioteca escrita em Ruby com o uso do Nokogiri para criar *interfaces* dinâmicas sem a utilização da linguagem ERB a partir de modelos. É uma proposta para geração de HTML com Ruby puro. Como Effigy já oferece suporte para o uso do Nokogiri no ambiente Rails, esta ferramenta foi utilizada no sistema proposto como meio de comunicação entre o *template* e a *view* que faz sua transformação a partir das regras definidas pelo usuário.

3.3 SelectorGadget

SelectorGadget[31] é uma ferramenta de código aberto escrita em Javascript para a descoberta de seletores CSS através de uma *interface* visual, onde é possível ver os elementos selecionados destacados na página.

Seu funcionamento é bastante simples. Depois de ativá-lo, basta clicar no correspondente elemento da página que se deseja selecionar. SelectorGadget então irá evidenciá-lo em verde, e em amarelo todos outros elementos da página alvo que

[29] http://nokogiri.org/Nokogiri/XML/Node.html

[30] http://github.com/jferris/effigy/

[31] http://www.selectorgadget.com/

pertencerem a mesma classe. Para restringir a seleção, é só clicar nos elementos em amarelo que foram evidenciados indiretamente, que irão se tornar vermelhos. Basta continuar este processo até que somente os elementos desejados estejam em verde ou amarelo, assim o SelectorGadget irá indicar o seletor CSS (ou os seletores) que referenciam estas etiquetas.

Por utilizar seletores CSS para fazer as modificações nos *templates*, viu-se necessário uma ferramenta que auxiliasse este processo de identificação dos elementos via seletores CSS. O SelectorGadget mostrou-se uma interessante solução por ter uma interface visual clara e simples de se usar, e por ter seu código aberto. Futuramente, esta ferramenta poderá ser acoplada ao sistema, eliminando a necessidade do usuário ter que lidar com os seletores.

4 ARQUITETURA DO SISTEMA

Assim como toda aplicação desenvolvida em Ruby on Rails, PrankDev está estruturada no modelo MVC (*Model-View-Controller*), que é uma arquitetura bastante apropriada para sistemas *web*. Os modelos fornecem uma representação das informações da aplicação, onde os dados são tratados e validados antes da gravação efetiva no banco de dados. As *views* são responsáveis por apresentar o conteúdo de forma apropriada ao usuário. Por fim, os controladores executam a lógica da aplicação e fazem a conexão entre as partes do sistema. A Figura 4.1 ilustra os caminhos de comunicação da arquitetura MVC em uma aplicação Rails.

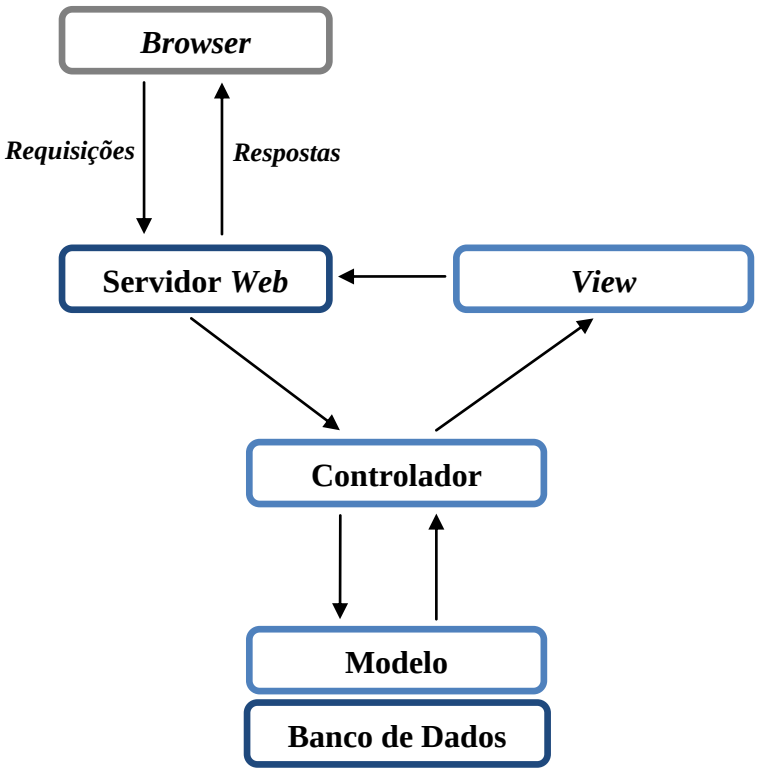

Figura 4.1: Fluxo de comunicação entre os módulos do sistema proposto.

Do ponto de vista prático, quando um usuário realiza uma operação no *browser* (navegador), este envia uma requisição para o servidor Rails do PrankDev que, por sua vez, repassa a ação desejada pelo usuário ao controlador. O controlador atende a requisição, processa e, se necessário, consulta ou modifica o banco de dados através do

modelo. Em seguida, a *view* que responde a requisição é construída a partir do controlador e repassada ao servidor, que a entrega ao *browser*. Nas próximas seções, o funcionamento de cada módulo desta arquitetura é descrito em detalhes.

4.1 Models

A camada de modelos do padrão MVC implementada pelo *framework* Ruby on Rails é responsável pela manipulação adicional ou adequação dos dados antes da gravação definitiva no banco de dados. Os modelos do PrankDev são: Rule, que descrevem as regras; Page, que define as estruturas dos arquivos e páginas; User, que modela os usuários do sistema; e Transform, que faz parte da estrutura de transformação dos arquivos HTML a partir das regras definidas pelo usuário.

Os modelos na verdade refletem as tabelas presentes no banco, com exceção do modelo Transform, que se faz necessário apenas para o complemento da comunicação entre o controlador e a *view* Transform. Esta presença é exigida pela ferramenta Effigy. Porém, como não existem informações no banco de dados relativas à transformação das páginas HTML, este modelo não executa nenhum tratamento de informações (que, tipicamente, é a função dos modelos em aplicações Rails). A transformação efetiva das páginas a partir das regras é realizada na *view* Transform, que está detalhada no Capítulo 5.

Desta maneira, podemos descrever os modelos em função de suas respectivas tabelas no banco de dados SQLite3, que são: User, Page e Rule. O diagrama entidade-relacionamento dos modelos do sistema PrankDev pode ser observado na Figura 4.2. Um usuário pode manter quantos arquivos desejar, e cada arquivo (HTML) pode possuir inúmeras regras.

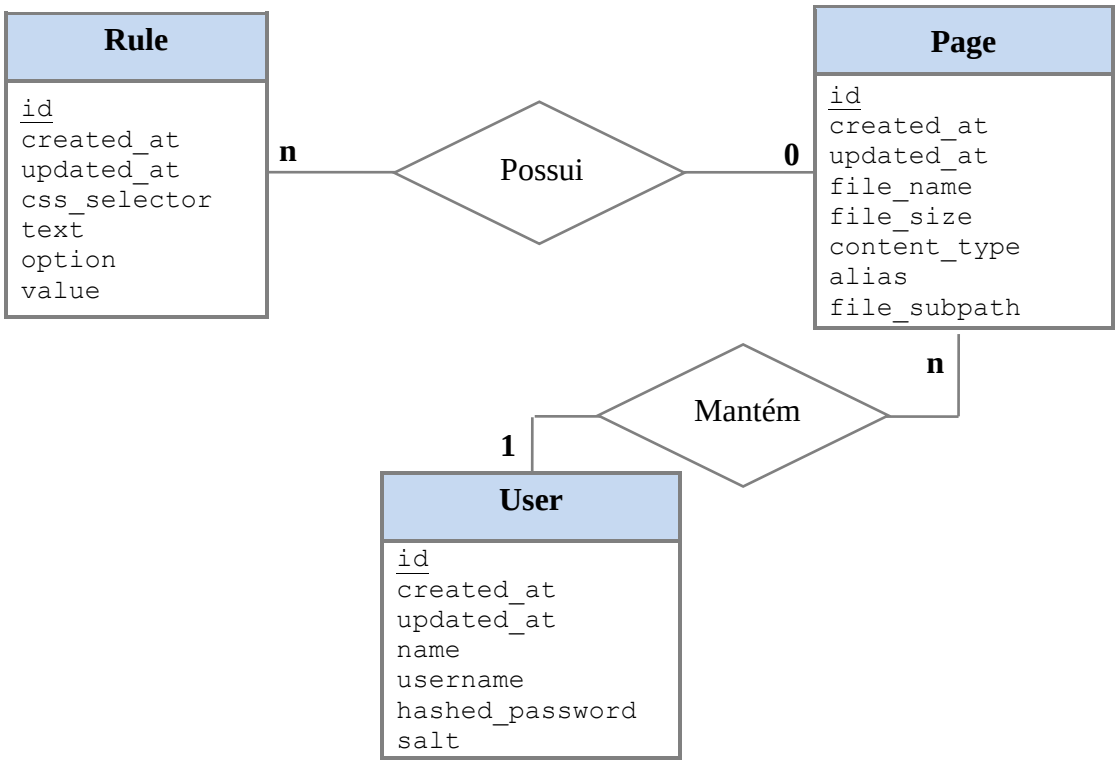

Figura 4.2: Diagrama entidade-relacionamento dos modelos do PrankDev

4.1.1 *User*

A tabela *User*, contém as informações acerca dos usuários registrados no sistema. A Tabela 4.1 descreve seus atributos. Os três primeiros itens, `id`, `created_at` e `updated_at` são campos adicionados automaticamente pela aplicação Rails. Estes dois últimos, que guardam o momento temporal de criação e atualização dos dados na tabela, são úteis para o maior controle da aplicação sobre o banco de dados. O atributo `name` refere-se ao nome completo do usuário e `username` ao nome de usuário para *login* no sistema.

O atributo `hashed_password` é a senha do usuário embaralhada pela `salt`. A `salt` é um número aleatório gerado para cada usuário. Isto é feito por questões de segurança. Em qualquer *software* computacional, não deve-se manter senhas de usuário de maneira clara no banco de dados, pois esta informação pode ser recuperada por usuários mal intencionados. Este embaralhamento dificulta consideravelmente a recuperação de senhas. Mais detalhes sobre o funcionamento deste tipo de criptografia utilizada no sistema pode ser encontrado em (RUBY, 2006).

Tabela 4.1: A tabela *User* do sistema PrankDev.

User	
id	(Integer)
created_at	(String)
updated_at	(String)
name	(String)
username	(String)
hashed_password	(String)
salt	(String)

O campo `id` é a chave primária da tabela *User*, e é um número inteiro atribuído de forma incremental a partir de zero. Rails utiliza este sistema de chave por padrão, para facilitar seu controle sobre o banco de dados.

4.1.2 *Page*

A tabela *Page* modela os arquivos que um determinado usuário possui. Como visto anteriormente, um usuário pode ter vários arquivos para manipular. A Tabela 4.2 detalha todos os atributos da tabela *Page*.

De forma análoga à tabela *User*, o campo `id` representa a chave primária, e `created_at` e `updated_at` guardam as datas de criação e última atualização dos dados no banco. O campo `user_id` é chave estrangeira que faz referência ao campo `id` da tabela *User*, e é utilizada para recuperar os arquivos de um determinado usuário. Já o atributo `file_subpath` mantém o sub-endereço do arquivo dentro do diretório raiz do usuário.

O atributo `alias` contém o endereço de diretório do arquivo modificado pelas regras no *template*. Apesar de este endereço ser fixo em relação ao arquivo no sistema, optou-se por manter esta informação no banco de dados para facilitar sua manipulação, uma

vez que este é referenciado frequentemente durante o uso do *site*. Desta forma evita-se calcular este endereço dinamicamente toda vez que o arquivo necessitar ser chamado.

Tabela 4.2: A tabela *Page* do sistema PrankDev.

Page	
id	(integer)
created_at	(string)
updated_at	(string)
file_name	(string)
file_size	(integer)
content_type	(string)
user_id	(integer)
alias	(string)
file_subpath	(string)

4.1.3 *Rule*

A tabela *Rule* mantém as informações das regras aplicadas a uma determinada página. Tipicamente, uma página HTML contém várias regras no sistema. A Tabela 4.3 descreve os atributos de *Rule*. Assim como as tabelas *User* e *Page*, possui os três campos da aplicação Rails: `id`, `created_at` e `updated_at`, onde `id` é a chave primária. O atributo `page_id` é chave estrangeira que faz referência ao campo `id` da tabela *Page*. Esta chave é necessária para recuperar as regras aplicadas a uma determinada página.

O campo `css_selector` contém o seletor CSS que representa os elementos alvo da regra no arquivo HTML. `text` guarda o texto que substituirá o conteúdo ao qual o seletor CSS aponta. O campo `option` apresenta as opções adicionais de regras, que são duplicar elemento e substituir por imagem. Por fim, o atributo `value` contém o endereço de *link* que, se presente, será aplicado ao elemento alvo.

Tabela 4.3: A tabela *Rule* do sistema PrankDev.

Rule	
id	(integer)
created_at	(string)
updated_at	(string)
page_id	(integer)
css_selector	(string)
text	(text)
option	(string)
value	(string)

4.2 Controllers

Os controladores são os responsáveis por tratar e processar as informações provenientes do banco de dados e destinadas ao usuário através das *views*, assim como o tratamento das informações fornecidas pelo usuário, antes da gravação das mesmas no banco de dados. É onde a maior parte da lógica da aplicação se encontra.

Nos controladores, cada ação é representada por um método, que corresponde a alguma funcionalidade do sistema. Existem também métodos auxiliares às ações, que normalmente são utilizados em conjunto com os primeiros. De maneira geral, cada ação possui uma *view* relacionada, tornando o sistema modular. Cada classe no modelo representa uma tabela no banco de dados que obrigatoriamente possui um controlador, porém o inverso nem sempre acontece: existem controladores que não necessitam de um modelo, como é o caso do `AdminController`, que implementa o *login* de usuários no sistema.

Os controladores do sistema PrankDev são:

- `AdminController`: realiza as ações de *login* e *logout* dos usuários no sistema.
- `ApplicationController`: é o controlador geral da aplicação, e classe pai de todos os outros controladores. Realiza ações sobre a aplicação, como proteger o acesso a usuários que não realizaram o *login*.
- `PagesController`: realiza a inserção, atualização e exclusão dos arquivos do usuário.
- `RulesController`: realiza a inserção, atualização e exclusão das regras de cada página HTML.
- `TransformController`: possui somente um método, `index`, que faz referência à página a ser modificada quando este é chamado. Seu funcionamento é diferente dos demais controladores e está explicado de maneira mais detalhada no Capítulo 0.
- `UsersController`: realiza o controle das informações dos usuários do sistema, como cadastro, atualização de conta e exclusão de conta.

4.3 Views

As *views* são as responsáveis por fazer a interação com o usuário. Elas recebem dados do sistema e os apresenta amigavelmente ao usuário final, assim como repassam os dados inseridos por ele para o controlador. As *views* no sistema PrankDev são escritas na linguagem híbrida ERB, portanto são renderizadas em HTML em tempo de execução à medida que são feitas solicitações ao servidor pelo usuário.

Tipicamente, cada controlador possui uma série de *views*, que representam ações chamadas através de métodos ao controlador. Assim, para cada *view* de um controlador temos ao menos um método associado a ele. PrankDev possui quatro conjuntos de *views* típicas: *admin*, *pages*, *rules* e *user*. A lista de controladores e suas respectivas *views* podem ser observada na Tabela 4.4 .

Tabela 4.4: *Views* e correspondentes controladores do sistema PrankDev.

Conjunto de views	Controlador	Arquivos	Métodos Relacionados
admin	`AdminController`	• login.html.erb	• `login`
pages	`PagesController`	• index.html.erb • new.html.erb	• `index` • `new` • `create` • `delete`
rules	`RulesController`	• edit.html.erb • index.html.erb • new.html.erb • show.html.erb	• `edit` • `index` • `new` • `create` • `show` • `update` • `delete`
users	`UsersController`	• edit.html.erb • new.html.erb • show.html.erb	• `edit` • `new` • `create` • `show` • `update` • `delete`

4.3.1 A *View* Transform

A *view* Transform se comporta de um modo particular, diferente das demais. Ela é a responsável pelo processamento e aplicação das regras em uma determinada página. Quando uma certa regra é criada, alterada ou excluída, o controlador `RuleController` invoca o método *index* do controlador `TransformController`, que então chama a *view* Transform, repassando as referências da página a ser modificada e as regras a serem aplicadas nela. Esta *view* então processa e aplica as regras no modelo. Ou seja, é a classe `TransformIndexView` que efetivamente aplica estas modificações. Este funcionamento está melhor descrito no Capítulo 5.

5 PRANKDEV EM DETALHES

PrankDev é uma proposta de hospedagem, criação e manutenção de *websites* para usuários comuns (Figura 5.1). Estes usuários, que são a maioria das pessoas que utilizam a Internet hoje, são tipicamente passívos, não atuando como fonte geradora de informação. E a ferramenta implementada neste trabalho objetiva fornecer um meio diferente para que este usuário produza informação.

Para o sucesso de uma ferramenta *web* que pretende atingir um grande número de usuários com perfis diferentes, a simplicidade é um fator muito importante. Apesar de ser uma ideia diferente da usual, a aplicação de regras para modificar um determinado *template* é um processo sitemático e simples. E podemos resumir o funcionamento do sistema proposto às aplicações de regras de transformação de páginas HTML. O restante do serviço apenas oferece uma estrutura para possibilitar o gerenciamento dos arquivos e regras em geral.

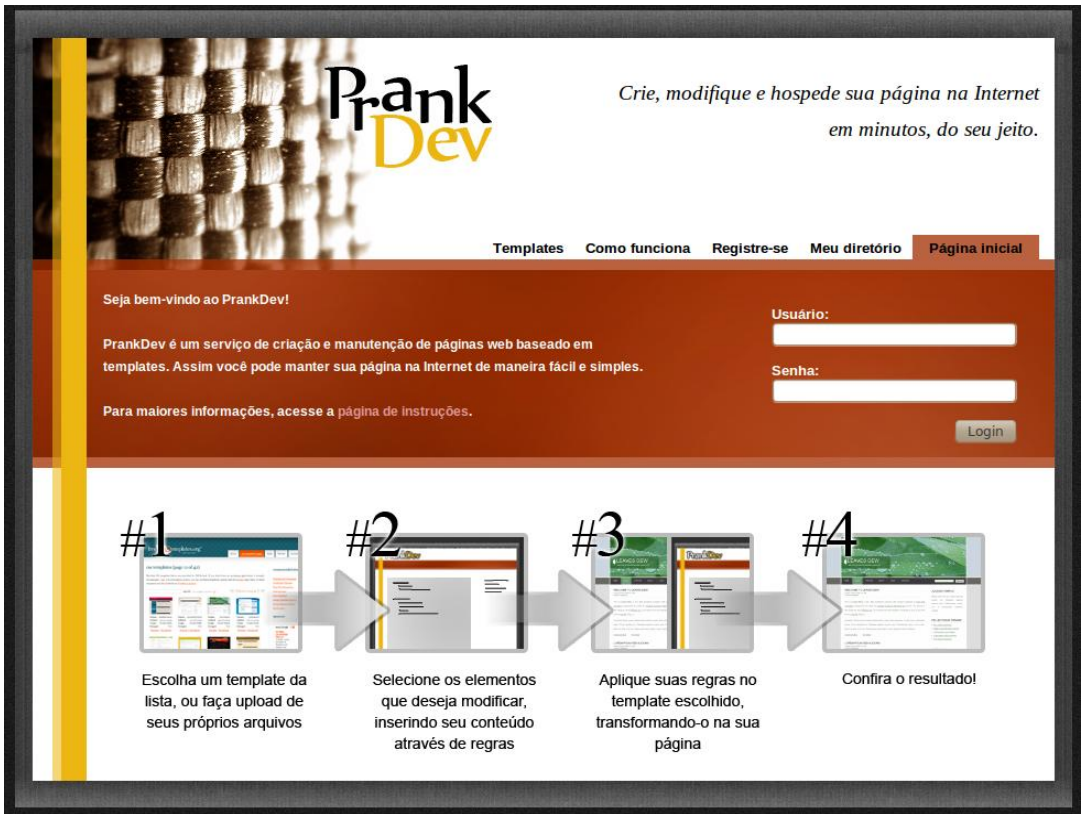

Figura 5.1: Tela inicial do sistema PrankDev.

Cada usuário do serviço possui um diretório próprio, onde é possível manter os arquivos de seu *website*. Neste diretório raiz, é possível realizar algumas operações como criar sub-diretórios, excluir, exibir e adicionar arquivos. Dentre estes arquivos, é possível realizar algumas ações adicionais àqueles que são documentos HTML, referentes à aplicação de regras de transformação. A Figura 5.2 exibe a tela do sistema que mostra o diretório do usuário, com sua respectiva lista de arquivos.

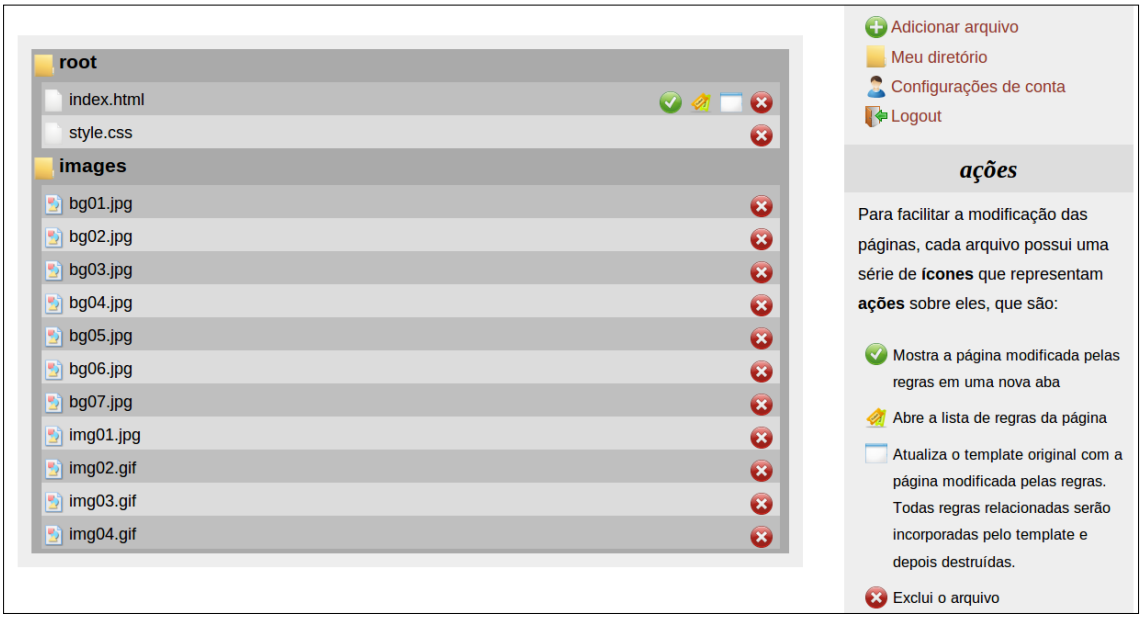

Figura 5.2: Tela do diretório do usuário, com sua lista de arquivos.

Neste Capítulo, iremos nos focar nos detalhes de como ocorre a transformação dos documentos HTML a partir das regras, já que este é motor da aplicação proposta. É possível consultar o código do PrankDev disponível livremente no repositório GitHub[32].

5.1 Aplicação de Regras

Cada documento HTML no diretório do usuário possui uma lista de regras de transformação que inicialmente está vazia, e não possui limite de número. Para cada arquivo, portanto, é possível inserir as regras que irão alterar o conteúdo do documento em questão. Uma regra é uma definição de modificação ou transformação de um elemento de uma página HTML. É através desta abstração que o usuário irá inserir o conteúdo que deseja no *template* escolhido.

Para inserir uma regra, é fornecido uma *interface* com quatro campos: “Seletor CSS”, “Novo texto”, “Link” e “Opções”. A partir da combinação dos valores possíveis nestes campos, diferentes tipos de regras são aplicados no documento a ser alterado.

A Figura 5.3 demonstra a inserção de uma regra a um arquivo HTML. No exemplo ilustrado, estamos definido que o texto contido no elemento da página alvo identificado pelo seletor CSS `'h1 a'`, será substituído pelo novo conteúdo “ufrgs”. A etiqueta `h1`

[32] http://github.com/ggtesta/prankdev

tipicamente define o título principal da página, ou seja, neste caso estamos dando um novo título (“ufrgs”) ao nosso *site*.

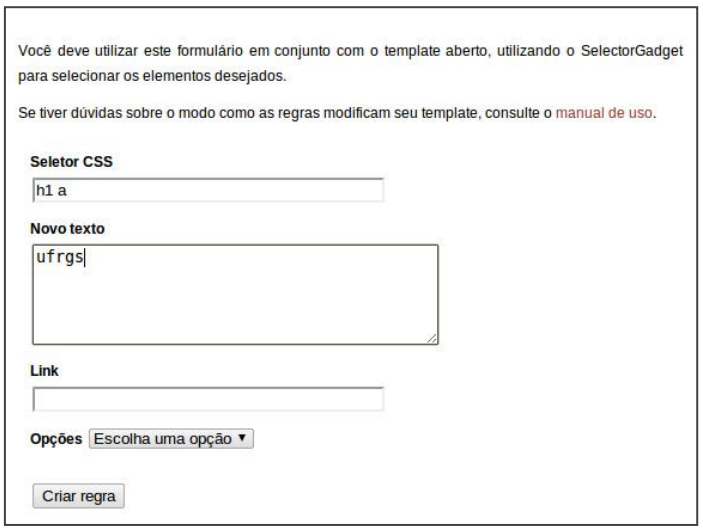

Figura 5.3: Inserindo uma nova regra a um documento.

Internamente, a aplicação das regras ao *template* é realizada na *TransformIndexView*, a *view* Transform, com o uso da biblioteca Nokogiri. Sempre que o usuário modificar o conjunto de regras relacionadas a um arquivo HTML de seu diretório, o sistema irá fazer uma chamada ao controlador *Transform*, o qual invocará o método *transform* da *view* de mesmo nome, repassando as informações sobre o documento envolvido e as regras relacionadas a ele (com as respectivas modificações). Neste ponto, o método fará uma cópia do arquivo original, que está localizado no diretório público do usuário, para dentro da aplicação, para que possa aplicar as regras a ele.

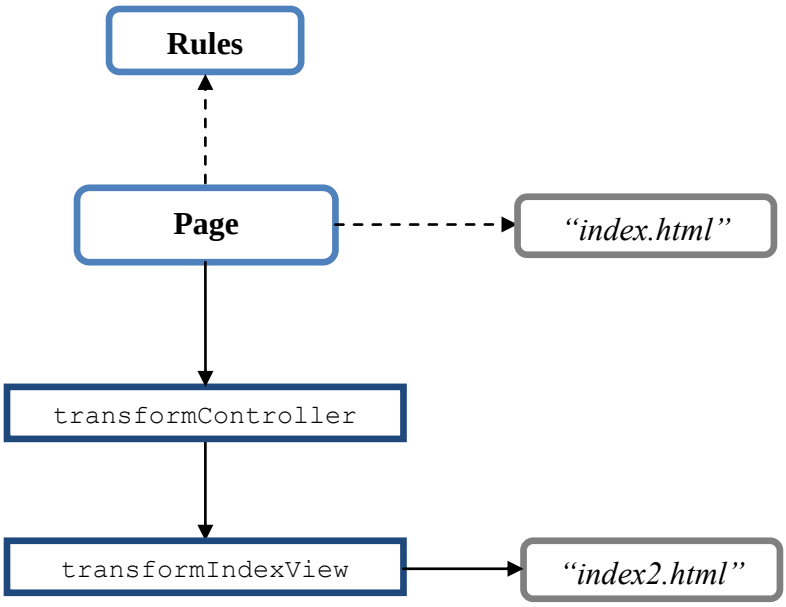

Figura 5.4: Caminho de aplicação das regras no arquivo hipotético *index.html*.

Neste momento, o método itera sob o conjunto de regras (que é um *array* de objetos da classe modelo *Rule*), tratando e aplicando cada regra individualmente, conforme sua categoria, ao *template*. De modo geral, existem quatro tipos de regras: de substituição de texto, de exclusão, de duplicação e de substituição por imagem (detalhadas nos próximos itens). Ao final da iteração, um arquivo de mesmo nome do original, com o sufixo adicional "2" é criado no diretório do usuário, com as regras aplicadas.

Criando um arquivo alternativo estático, a resposta a uma solicitação HTTP para visita ao *site* do usuário é mais rápida, já que não é necessário que o sistema crie dinamicamente o arquivo alterado com as regras a cada visita. O processamento e criação do documento HTML a partir das regras definidas pelo usuário é feito, portanto, somente quando o conjunto de regras é alterado.

5.1.1 Regras de substituição de texto

As regras de substituição de texto alteram o conteúdo textual do elemento selecionado pelo seletor CSS. Este procedimento é realizado recursivamente com a utilização do *parser* Nokogiri para selecionar e substituir somente o texto em questão, e não outras *tags* filhas do elemento selecionado. Por exemplo, considerando a seguinte etiqueta `h2` selecionada:

```
<h2 class="subtitle"><a href="link.html">Texto antigo</a></h2>
```

Ao alterar seu conteúdo, o sistema deve preservar a *tag* `a`, mantendo seu *link* original caso nenhum novo endereço tenha sido fornecido no momento da criação da regra. Assim, preservamos a estrutura geral do arquivo HTML e sua apresentação. O resultado da substituição neste exemplo seria:

```
<h2 class="subtitle"><a href="link.html">Novo texto</a></h2>
```

Normalmente, esta regra é a mais utilizada durante a criação de um *website* com o sistema, já que o *template* em geral oferece uma estrutura bem definida que não necessita muitas modificações. O maior tempo gasto, portanto, está na adição de conteúdo às páginas, que é feito com o uso desta regra.

5.1.2 Regras de exclusão

A regra de exclusão é um caso particular da substituição de texto onde não é fornecido um novo conteúdo para ser inserido no elemento selecionado. Assim, ao realizar a alteração, o sistema irá excluir por completo a etiqueta em questão. Este tipo de regra é útil para eliminar elementos indesejáveis do *template*, mesmo imagens, que não irão ser aproveitados na página que está sendo construída.

Este procedimento é realizado excluindo o nodo selecionado e todos os seus filhos da árvore de etiquetas construída pelo *parser* ao analisar o documento.

5.1.3 Regras de duplicação

Este tipo de construção tem como objetivo duplicar elementos já existentes no *template* para serem utilizados de forma adicional na página. Por exemplo, na inclusão de itens em listas, de *links* em um menu, etc. Ao aplicar esta regra, é possível ainda definir o novo conteúdo do item e um *link* para um novo endereço.

Basicamente, existem dois tipos de etiquetas HTML: estruturais e de formatação. Uma *tag* estrutural define uma organização da disposição dos nodos e suas definições no documento como, por exemplo, as etiquetas de lista: `<ul>` e `<li>`. Uma *tag* de formatação, ajusta o aspecto visual do elemento ou adiciona alguma propriedade. Por exemplo, as etiquetas para texto em negrito, `<b>`, e de *link*, `<a>`.

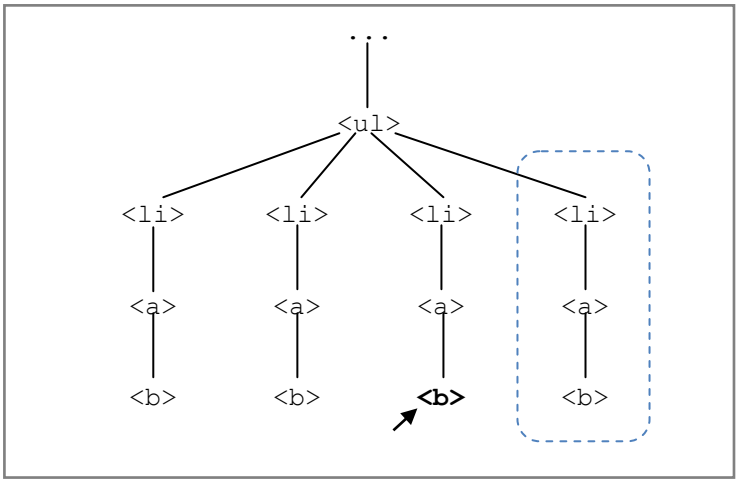

Figura 5.5: Exemplo de duplicação de um nodo na árvore gerada pelo *parser* Nokogiri.

Quando uma duplicação é executada, o sistema chama o método *clone* da *view Transform*, que adiciona um novo nodo estrutural na árvore construída pelo *parser*. Se a etiqueta selecionada pelo seletor CSS não for estrutural, seu nodo pai será selecionado. Recursivamente, este processo continua até que um nodo estrutural seja encontrado. A Figura 5.5 ilustra um exemplo de utilização. A seta indica o elemento selecionado, que é a etiqueta `<b>` do último item de uma lista. No momento da duplicação, o método irá procurar pelo primeiro elemento estrutural superior na hierarquia do documento, que neste caso, é a *tag* `<li>`, e criará um novo elemento, filho de `<ul>` (parte destacada da figura), com os atributos definidos na regra.

5.1.4 Regras de substituição por imagem

As regras de substituição por imagem têm como objetivo inserir imagens na página sendo construída. Este procedimento é aplicado em dois casos. Primeiro, se o elemento selecionado no *template* já contiver uma imagem, esta será substituída pela nova figura referenciada na regra. Segundo, se o seletor CSS apontar para um elemento textual, este conteúdo será substituído inteiramente pela imagem dada.

No primeiro caso, o método *transform* irá somente substituir o valor do atributo `src` da tag `<img>`, fazendo com que este contenha o endereço da nova imagem repassada na regra.

No segundo, todas as etiquetas de formatação serão excluídas do nodo pai selecionado, e um novo nodo `<img>` será criado, contendo o atributo `src` referenciando o endereço da imagem.

5.2 Algoritmo

A Tabela 5.1 contém o algoritmo simplificado do método *transform*, que realiza a aplicação das regras. Este código se encontra na *view Transform*.

Tabela 5.1: Algoritmo do método *transform*.

```
transform(page, rules)

   copy page to transform/index.html            // Copy the page to the
   open index.html                              // index.html file and
   create tree                                  // parse it with Nokogiri

   for each rule of rules begin
     case rule.option
       node = find_node(rule.css_selector)

       when 'clone' do

         new_node = node.duplicate              // Create a new node from
         new_node.content = rule.new_text       // the selected node and
         new_node.link = rule.value             // attach it to the tree
         node.parent.attach(new_node)           // of index.html

       when 'image' do

         if node.name = 'img' then              // Tests if the node is an
           node.attribute(src) = rule.new_text  // image. If not, create
         else                                   // a new image node. If it
           node.change_attributes(name = img,   // is, set properly the
           src = rule.new_text)                 // given attributes
         end

       else
         node.content = rule.new_text           // set the content and link
         node.link = rule.value
     end

  end
  copy index.html to page                       // recopy the file to the
  return                                        // page and return
end
```

6 ESTUDO DE CASO

Para demonstrar de forma prática a capacidade do PrankDev, foi criado um *website* alternativo a partir de um subconjunto das informações presentes no *site* do Instituto de Informática da UFRGS[33], utilizando um *template* livre extraído do repositório Free CSS Templates[34].

Foi escolhido um modelo que permitisse uma certa flexibilização nas informações, de modo que fosse possível apresentar o conteúdo da página do Instituto de Informática de maneira estruturalmente semelhante ao seu *site* original. Considerando tais fatos, optou-se pelo *template* Terrafirma2[35]. Este modelo apresenta uma certa estrutura de um *blog*, porém adapta-se bem para a experiência proposta. Ele é composto por um cabeçalho fixo horizontal na parte superior, um menu à direita com informações gerais com *links* diretos, e um espaço central à esquerda para o conteúdo principal. O *template* original Terrafirma2 pode ser observado na Figura 6.1.

Figura 6.1: O *template* original Terrafirma2 utilizado para o estudo de caso.

[33] http://www.inf.ufrgs.br. O conteúdo utilizado do *site* do Instituto de Informática da UFRGS foi consultado no mês de novembro de 2010.

[34] http://www.freecsstemplates.org/

[35] http://www.freecsstemplates.org/preview/terrafirma2

O *template* escolhido é composto somente por uma página HTML. Existem modelos que são compostos por mais páginas, de estilos diferentes, para serem usados conforme o conteúdo a ser exibido. Foi escolhido justamente um modelo de somente uma página para explorar a capacidade de modificação e adaptação do PrankDev.

6.1 Criando o *template* base

O primeiro passo para a criação de um *website* completo com o PrankDev consiste na criação do modelo básico para todas as páginas. Este procedimento evita a criação de regras repetidas para cada uma das páginas que compõe o *site*. Uma vez criado este *template* base, este é atualizado como modelo padrão através da opção "atualizar *template*". Assim, todas as regras definidas sobre ele serão incorporadas de forma definitiva.

No caso em estudo, esta etapa compreende criar as regras para o cabeçalho, o rodapé e do menu de contexto à direita do modelo. Para efeito de ilustração, a primeira regra aplicada substitui o título principal da página pelo correspondente título da página que desejamos construir, ou seja, "ufrgs". A regra utilizada para isto contém os atributos listados a seguir (campos marcados como "vazio" foram deixados em branco)[36]:

- **Seletor CSS**: "h1 a";
- **Novo texto**: "ufrgs";
- **Link**: vazio;
- **Outras opções**: vazio;

A Figura 6.2 mostra a aplicação desta primeira regra ao modelo.

Figura 6.2: Comparação do *template* Terrafirma2 antes e depois da aplicação da regra com o seletor CSS "`h1 a`".

[36] Consultar o Anexo – PrankDev: Guia de Utilização para informações complementares sobre a utilização do sistema.

Este processo de aplicação de regras prosseguiu até a construção do *template* base, que totalizou 18 regras aplicadas, sendo duas de substituição por imagem, três de duplicação de itens, quatro de exclusão de elementos e as restantes 9 de substituição simples de texto. Esta etapa consumiu cerca de 20 minutos para ser executada.

6.2 Criação das páginas intermediárias

A partir deste ponto, o *template* base criado foi utilizado para a construção das demais páginas que compõe o *site*. A Tabela 6.1 listas as páginas criadas, assim como o tempo gasto aproximado para construção de cada uma e as regras aplicadas sobre elas.

Tabela 6.1: Número de regras e tempo gasto para a construção de cada uma das páginas que compõe o estudo de caso.

Título da Página	**Número de Regras**	**Tempo** (em minutos)
Página inicial	14	23
Graduação	13	13
Pós-graduação	10	14
Pesquisa	8	9
Extensão	15	13
Institucional	14	13
INF completa 21 anos	7	6
INF realiza eleições	13	14
Inscrições no PPGC	11	9

As nove páginas HTML construídas mais a construção do *template* foram o resultado da aplicação de um total de 123 regras. Uma média de aproximadamente 14 regras por página. Esta etapa levou cerca de duas horas para ser completada[37], ou seja, 13 minutos por página. As regras aplicadas a cada uma das nove páginas estão listadas no Apêndice, na página 51.

6.3 Resultado

Ao final de todo o processo de criação do *site* alternativo do Instituto de Informática da UFRGS foram criadas 123 regras em 9 páginas diferentes. O tempo de construção total foi de aproximadamente duas horas e trinta minutos. A Figura 6.3 mostra a página inicial do site construído.

O *website* resultante mostrou-se plenamente funcional, com os *links* e imagens alterados apresentando-se de maneira coerente. A parte visual do estudo aplicado também foi satisfatória. As páginas resultantes preservaram a estrutura do *template* original, mantendo seu esquema gráfico.

[37] Não levando em consideração a criação e edição das imagens do *site*.

Figura 6.3: Página inicial do *website* criado com o PrankDev, a partir de informações da página do Instituto de Informática da UFRGS

Apesar de implementar somente as principais funções de manipulação HTML, a proposta de criação de páginas utilizando *templates* e seletores CSS descrita neste trabalho mostrou ser um interessante meio para alteração de páginas *web*. Algumas funções existentes ainda podem ser aprimoradas para melhorar o funcionamento, e outras opções de regras mais restritas podem ser implementadas.

6.4 Outras Aplicações

O algoritmo apresentado neste trabalho e as tecnologias envolvidas ainda podem ser utilizados para uma grande variedade de aplicações que ultrapassam a implementação proposta. Atualmente, existem inúmeros serviços *online* onde o usuário cadastrado pode personalizar seu ambiente dentro do respectivo *site* ou portal. Como exemplo, podemos citar as redes sociais de relacionamento na Internet, onde cada pessoa mantém um perfil público pessoal.

O esquema proposto neste trabalho pode ser utilizado para criar esta personalização, ao gosto do usuário. Neste tipo de serviço, as informações sobre cada pessoa encontram-se em um banco de dados e são recuperadas e apresentadas aos demais no momento em que existe uma requisição neste sentido. Para modificar o visual de seu

perfil (seja gráfica como estruturalmente), basta manter no banco de dados as informações sobre o *template* escolhido previamente e as regras estruturais que o modificam, que também são pré-definidas. Estas regras descrevem onde cada parte do conteúdo do perfil do usuário deve ser exibida. Assim, no momento de uma visita, o modelo apropriado é carregado e preenchido com as informações presentes no banco sobre o usuário.

Além deste exemplo, como dito antes, este sistema de personalização pode ser extendido para qualquer tipo de serviço que possua algum cadastro de usuários. Desta forma, mesmo em casos que não exista uma página pública, ele pode ser utilizado para satisfazer o gosto pessoal do cliente, como em serviços de E-mail, *internet banking*, portais de notícias dirigidas (que são escolhidas por tópicos de interesse), entre muitos outros.

7 CONCLUSÃO

Este trabalho foi desenvolvido com o objetivo de definir um serviço *online* de criação e hospedagem de *websites* de qualidade para usuários comuns. Esta tarefa não é trivial, já que elementos visualmente interessantes requerem um conhecimento teórico e tecnológico, e não são fáceis de construir. Isto se reflete nos serviços de criação de páginas *web* oferecidos na *Internet* atualmente, que se mostram limitados visualmente.

Neste sentido, a utilização de *templates*, proposta neste trabalho, como base para a construção de páginas *web* mostrou ser um interessante meio para a criação de *websites* simples mas visualmente interessantes. Assim, a partir da enorme quantidade de modelos HTML e CSS disponíveis gratuitamente, é possível construir uma página de maneira fácil e simples com um resultado interessante

O sistema protótipo, PrankDev, é uma proposta de utilização destes *templates* disponíveis gratuitamente para a construção de páginas *web*, empregando tecnologias de código aberto.

Apesar do sistema implementado ser um protótipo, os resultados obtidos na experiência prática utilizando a ferramenta desenvolvida indicaram que este é um interessante meio para a construção de *websites*. Além de ser relativamente flexível para propósitos gerais, consome pouco tempo de construção e produz páginas de qualidade.

O *parser* Nokogiri, escrito na linguagem Ruby, se revelou como um poderoso analisador de documentos HTML, e em conjunto com o *framework* Ruby on Rails molda uma plataforma de desenvolvimento *web* baseada em banco de dados eficiente e produtiva para o problema proposto.

7.1 Trabalhos Futuros

A solução proposta neste trabalho é um protótipo, e existem muitas melhorias e incrementos de funcionalidades que podem ser discutidas e implementadas como trabalhos futuros. Algumas delas são:

- Integrar a ferramenta de seleção dos elementos HTML com o sistema, evitando assim que o usuário necessite copiar o seletor CSS do elemento;
- Dar suporte para a modificação de parte do conteúdo de um determinado elemento;
- Melhorar o gerenciamento dos arquivos e diretórios do usuário, permitindo cópia e renomeação dos mesmos.
- Melhorar a ferramenta de substituição de imagens e possibilitar a mudança de seus atributos.

- Permitir importação automática de *templates* dos principais repositórios de modelos CSS livres da Internet.

Além destes aprimoramentos do sistema desenvolvido, um trabalho futuro alternativo é o de utilizar o algoritmo apresentado de aplicação de regras em *templates* para criar uma ferramenta de personalização de páginas de usuários em serviços *online*, conforme demonstrado no item 6.4. É possível implementar um mecanismo independente da plataforma utilizada no restante do serviço, apenas como uma ferramenta genérica que possa ser acoplada ao sistema principal.

REFERÊNCIAS

FITZGERALD, M. **Learning Ruby**. 1st ed. Beijing : O'Reilly, 2007. 238p.

CAMERON, A. et al. **A Arte e a Ciência da CSS**. Porto Alegre: Bookman, 2009.

ORSINI, R. **Rails Cookbook**. 1st ed. Beijing : O'Reilly, 2007. 514p.

FREEMAN, E. **Use a cabeça!: HTML com CSS e XHTML**. 2. ed. Rio de Janeiro: Alta Books, 2008. 580p.

SILVA, M S. **Construindo sites com CSS e (X)HTML: sites controlados por folhas de estilo em cascata.** 1. ed. São Paulo: Novatec Editora, 2008. 446p.

REGGETT D. et al. **Reggett on HTML 4**. 2nd ed. Addison-Wesley Longman, 1998. ISBN 0-201-17805-2. 437p.

NEGRINO, T; SMITH D. **JavaScript para a World Wide Web**. 4. ed. Rio de Janeiro: Editora Campus, 2001. ISBN 85-352-0841-0. 430p.

LIE, H W; BOS B. **Cascading Style Sheets: Designing for Web.** 3rd ed. Crawfordsville: Addison-Wesley, 2005.

CASTRO, E. **HTML for the World Wide Web: visual quickstart guide**. 5th ed. Berkeley: Peachpit, 2003. 480p.

CASTRO, E. **HTML, XHTML & CSS**. 6 th ed. Berkeley: Peachpit Press, 2007.

FULTON, H. **The Ruby Way: Solutions and Techniques in Ruby Programming**. 2nd ed. Indiana: Addison-Wesley. 2002.

CARLSON L.; RICHARDSON L. **Ruby Cookbook**. 1st ed. Sebastopol: O'Reilly Media. 2006.

RUBY s.; THOMAS, D.; HANSSON H. H. **Agile Web Development with Rails**. 3rd ed. Dallas: The Pragmatic Bookshelf, O'Reilly Media. 2009. 797p.

WILLIAMS, J. **Rails Solutions: Ruby on Rails Made Easy**. 2nd ed. Berkeley: Apress. 2007.

CASAL, D. P. **Advanced Software Development for Web Applications**. Centre for Excellence in Learning Technology, Goldsmiths College. University of London. 2005.

RUBY. *Website* oficial da linguagem Ruby, disponível em [http://www.ruby-lang.org]. Acesso em: set 2010.

RUBY ON RAILS. *Website* oficial do framework Ruby on Rails, disponível em [http://www.rubyonrails.org]. Acesso em: out 2010.

NOKOGIRI. *Website* oficial da biblioteca Nokogiri, disponível em [http://www.nokogiri.org]. Acesso em: out 2010.

W3C. *Website* da World Wide Web Consortium, disponível em [http://www.w3c.org]. Acesso em: set 2010.

XML. *Website* da linguagem XML mantida pela W3C, disponível em [http://www.w3c.org/XML]. Acesso em: nov 2010.

WEB 2.0. Enciclopédia Livre Wikipedia, disponível em [http://en.wikipedia.org/wiki/Web_2.0]. Acesso em: nov 2010.

ANEXO – PRANKDEV: GUIA DE UTILIZAÇÃO

Nesta seção, estão detalhados os procedimentos mais comuns para a utilização do sistema proposto, PrankDev.

Registro

Para utilizar o serviço oferecido pelo PrankDev, é preciso realizar um pequeno cadastro. Este cadastro é necessário para poder usufruir dos serviços de hospedagem do *site* e gerenciamento dos seus arquivos.

Os campos obrigatórios neste cadastro são:

- **Nome completo**;
- **Nome de usuário**, que será utilizado para fazer o *login* no sistema;
- **Senha**, que será utilizada em conjunto com seu nome de usuário no *login*;
- **Confirmar senha**.

Importar *template*

PrankDev oferece alguns *templates* interessantes para servirem de modelo a seu *website*, facilitando o desenvolvimento inicial. É útil também para testar e conferir as funcionalidades do *site*. É possível importá-los diretamente para seu diretório clicando no botão "Importar" de cada *template*.

Tenha cuidado ao importar um modelo, pois este irá substituir quaisquer arquivos existentes no seu diretório que tiverem o mesmo nome.

Adicionar arquivos

Se preferir, você pode fazer *upload* de seus próprios arquivos clicando em "Adicionar arquivo" no menu de contexto à direita da lista de arquivos de seu diretório (ver Figura 7.1). No formulário que se abre, basta clicar em "procurar" para chamar a caixa de diálogo de seleção de arquivos. Logo abaixo, você pode selecionar, em uma lista, a pasta destino dentre as existentes em seu diretório (inicialmente estará em branco pois só haverá a pasta raiz, "*root*", que não aparece listada).

Para criar um novo diretório, basta digitar o nome da nova pasta no campo "Novo diretório". Todos os arquivos adicionados aparecerão na lista inicial de seu diretório.

Figura 7.1: Adicionando arquivos no diretório do usuário

Meu diretório

Os arquivos que forem enviados ao sistema através da ferramenta "Adicionar arquivos", assim como os arquivos importados via "Importar *template*", serão listados em seu diretório. Aqui é possível gerenciar seus arquivos através de ações representadas pelos ícones localizados à direita de cada arquivo (ver Figura 7.2).

As ações possíveis sobre os arquivos HTML são:

- Mostrar a página modificada pelas regras em uma nova aba.
- Abrir a lista de regras da página.
- Atualizar o *template* original com a página modificada pelas regras. Todas regras relacionadas serão incorporadas pelo *template* e depois destruídas.

Clicando no nome do arquivo, é possível ainda visualizá-lo em uma nova aba. Além destas opções, todos os arquivos podem ser excluídos:

- Exclui o arquivo

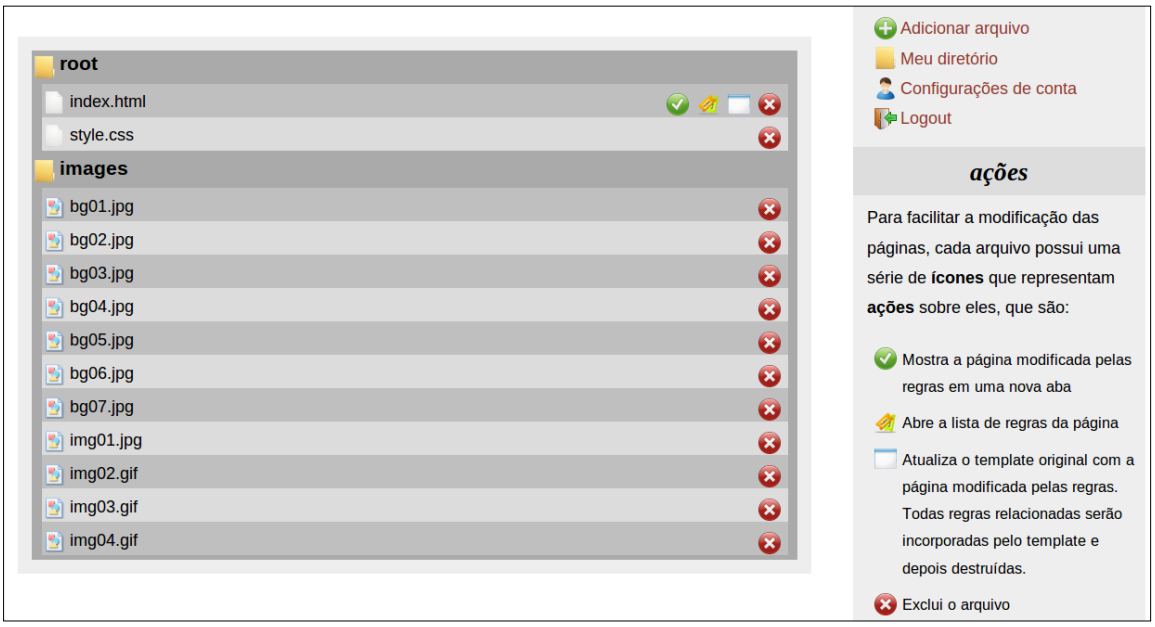

Figura 7.2: Tela do diretório do usuário

Adicionar Regra

O formulário para criação de uma nova regra contém:

- **Seletor CSS**: o seletor referente ao elemento HTML do *template* a ser modificado. Aconselha-se a utilização do SelectorGadget para tal fim.
- **Novo texto**: contém a informação que substituirá o texto do elemento selecionado. Ao ser utilizado em conjunto com a opção "Substituir por imagem", deverá conter o endereço da nova imagem. Se deixado em branco, o elemento selecionado será excluído da página.
- ***Link***: contém o endereço de *link*, caso necessário.
- **Opções:**
  - **Duplicação**: será feita uma cópia do elemento selecionado, com o texto informado no campo "Novo texto".
  - **Substituição por imagem**: substituirá o elemento selecionado (texto ou imagem), pela imagem referenciada no endereço dado.

Sempre que uma regra for criada, modificada ou excluída, o sistema irá processar o documento resultante novamente. Assim, após cada alteração, é possível verificar suas modificações atualizando a página.

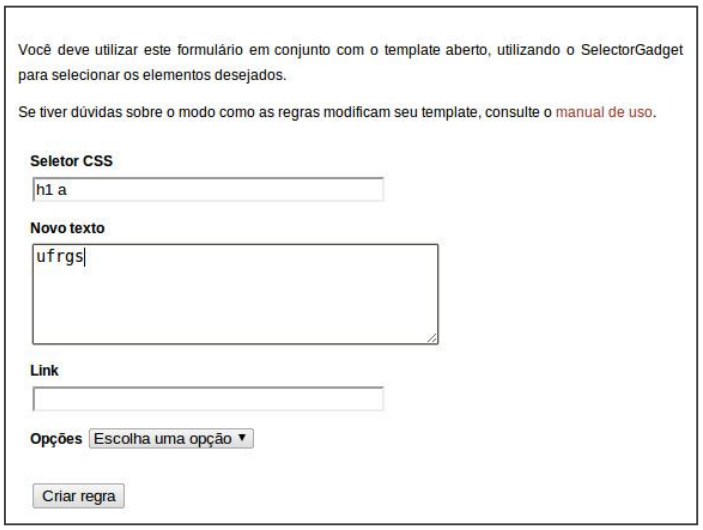

Figura 7.3: Adicionando uma regra.

Lista de Regras

Cada arquivo HTML possui uma lista de regras diferente, que inicialmente se encontra vazia. As ações sobre as regras estão representadas pelos ícones localizados à direita de cada regra na lista de regras. São elas:

 Exibe todas as informações sobre a regra.

 Edita a regra selecionada

 Exclui a regra

Além destas, é possível ainda criar uma regra nova clicando sobre "Adicionar regra", no menu de contexto à direita da página.

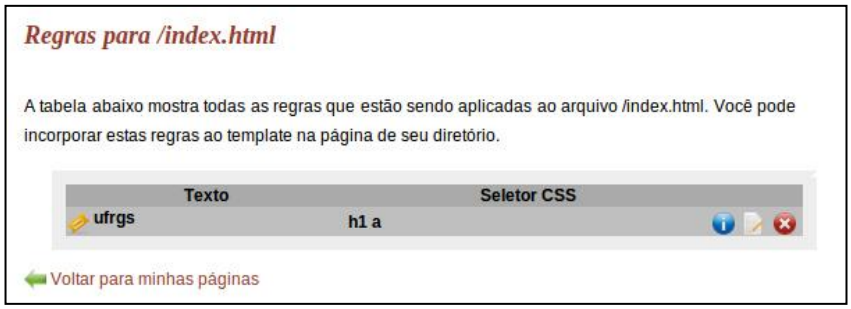

Figura 7.4: Lista de regras para o arquivo index.html

SelectorGadget

O funcionamento do SelectorGadget para seleção de elementos HTML é bastante simples. Depois de ativá-lo, basta clicar no correspondente elemento da página que se deseja selecionar. SelectorGadget então irá evidenciá-lo em verde, e em amarelo todos outros elementos da página alvo que pertencerem a mesma classe. Para restringir a seleção, é só clicar nos elementos em amarelos que foram evidenciados indiretamente, que irão se tornar vermelhos. Basta continuar este processo até que somente os elementos desejados estejam em verde ou amarelo, assim o SelectorGadget irá indicar o seletor CSS (ou os seletores) que referenciam estes elementos.

Para informações mais detalhadas, visite o site do SelectorGadget: http://www.selectorgadget.com/.

Atualizando o *template*

Ao atualizar o *template*, todas as regras aplicadas a ele serão incorporadas no arquivo original. E, portanto, estas regras relacionadas serão excluídas. Assim, é possível facilitar a manutenção da página, incorporando os elementos que normalmente são fixos ao modelo, e evitando um número muito grande de regras.

APÊNDICE – LISTA DE REGRAS DO ESTUDO DE CASO

A seguir, estão listadas todas as regras criadas com PrankDev para o estudo de caso detalhado no Capítulo 6. Em todas as tabelas, o campo "Opção", quando vazio, refere-se a uma regra de substituição de texto. Alternativamente, pode ser "clone", que indica que o elemento selecionado é duplicado, ou "image" que refere-se a uma substituição por imagem. Nas regras onde "Novo texto", "Link" e "Opção" estiverem vazios, os elementos selecionados serão excluídos da página alvo.

Tabela 7.1: Regras para a página "index.html" do estudo de caso.

Seletor CSS	Novo Texto	Link	Opção
#content :nth-child(1) .title a	INF realiza eleições para diversos cargos	eleicoes2.html	
#content :nth-child(2) .title a	Inscrições no PPGC até o dia 25/11	ppgc2.html	
#content :nth-child(4) .title a	INF completa 21 anos	21anos2.html	
#content :nth-child(4) .date	12 de Novembro de 2010		
#content :nth-child(2) .tags a:nth-child(1)	ppgc		
#content :nth-child(2) .tags a:nth-child(2)	inscricoes		
#content :nth-child(2) .tags a:nth-child(3)	inf		
#content :nth-child(4) .entry p	Na última terça-feira, dia 9, o INF completou 21 anos de existência, prestando um trabalho de excelência a serviço da sociedade no Estado e no País. O INF é líder na formação de profissionais altamente qualificados e diferenciados, que têm ajudado a transformar o país pela geração de riqueza, maior motor do crescimento social.		
#content :nth-child(4) .meta .tags a:nth-child(1)	Aniversario		
#content :nth-child(4) .meta .tags a:nth-child(2)	Inf		
#content :nth-child(4) .meta .tags a:nth-child(3)	excelência		
#content :nth-child(1) .more	ler mais...	eleicoes2.html	
#content :nth-child(2) .links .more	ler mais...	ppgc2.html	
#content :nth-child(4) .meta .more	ler mais...	21anos2.html	

Tabela 7.2: Regras para a página “pesquisa.html” do estudo de caso.

Seletor CSS	Novo Texto	Link	Opção
#content :nth-child(4) .date , #content :nth-child(4) a, #content p:nth-child(1), #content :nth-child(2) a, .meta, #content :nth-child(4) .title , .post:nth-child(2)			
.entry p	Grupos de Pesquisas	http://www.inf.ufrgs.br/index.php?option=com_content&view=category&layout=blog&id=38&Itemid=92	clone
.title a	Pesquisa		
.entry p:nth-child(2)	Pesquisa, inovação e desenvolvimento tecnológico possuem um papel de destaque entre as diversas atividades desenvolvidas no Instituto de Informática. Os grupos de pesquisa desenvolvem atividades nas principais áreas de Ciência e Engenharia da Computação e seus membros são ativos no ensino de graduação e pós-graduação em suas respectivas áreas.		
.alignleft, .date			
#content p:nth-child(1)	Projetos de pesquisa	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=830:projetos-de-pesquisa&catid=61&Itemid=93	clone
#content p:nth-child(1)	Seminários	http://www.inf.ufrgs.br/index.php?option=com_content&view=category&layout=blog&id=74&Itemid=119	clone
#content p:nth-child(1)	Comissão de Pesquisa	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=216&Itemid=60	clone

Tabela 7.3: Regras para a página “graduacao.html” do estudo de caso.

Seletor CSS	Novo Texto	Link	Opção
#content :nth-child(4) .date , #content :nth-child(4) a, .entry :nth-child(1), #content :nth-child(2) a, .meta, #content :nth-child(2) .title , #content :nth-child(2) .entry, #content :nth-child(4) .title, #content :nth-child(4) .entry			
#content :nth-child(1) .title a	Graduação		
.date			
#content p:nth-child(1)	O INF - UFRGS oferece dois cursos de graduação: Bacharelado em Ciência da Computação e Engenharia da Computação (este último oferecido em conjunto com a Escola de Engenharia). Ambos os programas estão entre os melhores do país, de acordo com o Ministério da Educação e avaliadores independentes. A cada ano são admitidos 100 novos estudantes em Ciência da Computação e 60 em Engenharia da Computação.		
#content p:nth-child(1)	Página do curso de Engenharia da Computação	http://www.ufrgs.br/ecp	clone
#content p:nth-child(1)	Página do curso de Ciência da Computação	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=205&Itemid=74	clone
#content p:nth-child(1)	O II/UFRGS é reconhecido internacionalmente como uma das instituições mais importantes da América Latina em Ciência da Computação e Engenharia de Computação. Seu corpo docente inclui doutores (PHDs) titulados por cerca de 20 instituições diferentes de 8 países. Esta diversidade possibilitou vários acordos bem sucedidos de cooperação internacional. Nossa equipe acadêmica tem forte participação como membros em fóruns internacionais – corpo editorial de publicações, comitês de programa de conferências e sociedades científicas.		clone
#content p:nth-child(1)	A sólida base fornecida pelo II/UFRGS, complementada com as oportunidades de estágio em instituições estrangeiras, resulta em amplas oportunidades de carreira profissional para os estudantes.		clone

#content p:nth-child(1)	O curso de Engenharia de Computação qualifica profissionais para trabalhar em áreas nas quais a eletrônica e a computação são complementares, tais como sistemas microprocessados, eletrônica embarcada, redes de comunicação, automação industrial e microeletrônica.	clone
#content p:nth-child(1)	Os graduados no Bacharelado em Ciência da Computação são qualificados em projeto, desenvolvimento, implementação e gerência de sistemas de computação, o que inclui fornecer soluções para computadores pessoais, jogos em computadores, dispositivos móveis e redes corporativas complexas.	clone
#content p:nth-child(1)		clone
#content p:nth-child(2)	images/picture.jpg	image

Tabela 7.4: Regras para a página “institucional.html” do estudo de caso.

Seletor CSS	Novo Texto	Link	Opção
.date , .post:nth-child(2), #content :nth-child(2) .title, #content :nth-child(2) a, #content :nth-child(2) .entry, .entry :nth-child(1), #content :nth-child(4)			
.title	Institucional		
#content p:nth-child(1)	Regimento	http://www.ufrgs.br/consun/leis/dec265-09.htm	clone
#content p:nth-child(1)	Semana Acadêmica	http://www.inf.ufrgs.br/index.php?option=com_content&view=section&id=18&Itemid=123	clone
#content p:nth-child(1)	Contato	http://www.inf.ufrgs.br/index.php?option=com_contact&view=contact&id=1&Itemid=79	clone
#content p:nth-child(1)	Fotos e Vídeos	http://www.inf.ufrgs.br/index.php?option=com_expose&Itemid=122	clone
#content p:nth-child(1)	Logotipo	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=173&Itemid=78	clone
#content p:nth-child(1)	Localização	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=74&Itemid=77	clone
#content p:nth-child(1)	Avaliação Institucional	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=200&Itemid=76	clone
#content p:nth-child(1)	Prêmios e distinções	http://www.inf.ufrgs.br/index.php?option=com_content&view=category&layout=blog&id=46&Itemid=75	clone
#content p:nth-child(1)	Infra-estrutura	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=199&Itemid=74	clone
#content p:nth-child(1)	Organização	http://www.inf.ufrgs.br/index.php?option=com_content&view=category&layout=blog&id=48&Itemid=66	clone
#content p:nth-child(1)	Notícias	http://www.inf.ufrgs.br/index.php?option=com_content&view=category&layout=blog&id=1&Itemid=107	clone
#content p:nth-child(1)	Apresentação	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=68&Itemid=57	

Tabela 7.5: Regras para a página “extensao.html” do estudo de caso.

Seletor CSS	Novo Texto	Link	Opção
#content :nth-child(4) , .entry :nth-child(1), #content :nth-child(4) .title, .post:nth-child(2), #content :nth-child(2) .title, .meta, .date			
#content a	Extensão		
a:nth-child(2)			
#content p:nth-child(1)	In Company	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=841:cursos-in-company&catid=93&Itemid=108	clone
#content p:nth-child(1)	Abertos ao Público	http://www.inf.ufrgs.br/index.php?option=com_content&view=category&id=93:cursos-de-especializacao&layout=blog&Itemid=108	clone
#content p:nth-child(1)	Cursos de Especialização:		clone
#content p:nth-child(1)	Cursos Realizados	http://www.inf.ufrgs.br/index.php?option=com_content&view=category&id=91:cursos-realizados&layout=blog&Itemid=61	clone
#content p:nth-child(1)	Cursos em Andamento	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=838:cursos-em-andamento&catid=66&Itemid=94	clone
#content p:nth-child(1)	Inscrições Abertas	http://www.inf.ufrgs.br/index.php?option=com_content&view=article&id=839:cursos-inscricoes-abertas&catid=66&Itemid=94	clone
#content p:nth-child(1)	Cursos de Extensão:		clone
#content p:nth-child(1)			clone
#content p:nth-child(1)	Com competência comprovada ao longo dos anos, o Instituto de Informática propicia condições de atuação em uma gama de atividades de desenvolvimento tecnológico, bem como de formação de Recursos Humanos. Isto é feito a partir de nossos cursos regulares (graduação, mestrado e doutorado) ou através de atividades específicas de especialização e extensão.		clone

#content p:nth-child(1)	O Instituto de Informática da UFRGS tem como missão formar profissionais de reconhecida qualificação em Computação, propiciando a geração de conhecimento e tecnologia e apoiando o desenvolvimento de produtos e empresas.	clone
#content p:nth-child(1)	Centro de excelência na formação em Computação	clone
#content p:nth-child(1)	O Instituto de Informática da UFRGS oferece capacitação especializada para profiissionais de informática	

Tabela 7.6: Regras para a página “21anos.html” do estudo de caso.

Seletor CSS	Novo Texto	Link	Opção
.post:nth-child(2) , #content :nth-child(2) .title, #content :nth-child(4), #sidebar			
.alignleft			
#content p:nth-child(1)	A comunidade do INF sente-se orgulhosa ao olhar para trás e ver a trajetória cumprida, resultado do esforço conjunto e harmonioso de diversas gerações de professores, funcionários, estudantes e colaboradores. E ao olhar para o futuro, reconhece que tem potencial para atingir resultados ainda mais expressivos.		
#content p:nth-child(1)	O INF é líder na formação de profissionais altamente qualificados e diferenciados, que têm ajudado a transformar o país pela geração de riqueza, maior motor do crescimento social. O INF tem sido um líder também na geração de conhecimento científico, na transformação deste conhecimento em tecnologias inovadoras e na transferência destas tecnologias para a sociedade, tanto para empresas públicas e privadas já estabelecidas como através da criação de start-ups, exercendo um papel essencial na consolidação do Rio Grande do Sul como um cluster de alta tecnologia.		clone
#content p:nth-child(1)	Na última terça-feira, dia 9, o INF completou 21 anos de existência, prestando um trabalho de excelência a serviço da sociedade no Estado e no País.		clone
.title a	INF completa 21 anos		
.date	12 de Novembro de 2010		

Tabela 7.7: Regras para a página “ppgc.html” do estudo de caso.

Seletor CSS	Novo Texto	Link	Opção
.date , #content :nth-child(4)			
#content :nth-child(1) .title a	Inscrições no PPGC até o dia 25/11		
#sidebar, .alignleft			
#content :nth-child(1) p	Informações sobre os procedimentos de inscrição e sobre o calendário do processo seletivo, bem como sobre as linhas de pesquisa e tópicos de interesse dos orientadores, podem ser encontrados no portal do Programa, em http://ppgc.inf.ufrgs.br, ou através do site do Instituto de Informática em http://www.inf.ufrgs.br/.		clone
#content :nth-child(1) p	http://inf.ufrgs.br/ppgc/applying	http://inf.ufrgs.br/ppgc/applying	clone
p:nth-child(4) a			
#content :nth-child(1) p:nth-child(1)	Estão abertas as inscrições para o mestrado e doutorado no Programa de Pós-Graduação em Computação do Informática da UFRGS - PPGC/UFRGS, para ingresso em março de 2011. Elas podem ser feitas até o dia 25 de novembro, pelo site		
#content :nth-child(2) .title a	Bolsas		
#content :nth-child(2) .entry p	O PPGC-UFRGS é um dos maiores, mais antigos e mais abrangentes programas de pós-graduação em Computação do país. Avaliado com conceito 6 pela CAPES (em uma escala que vai até 7), o PPGC possui a melhor avaliação da região Sul e está entre os melhores programas de pós-graduação em Computação do país. Com 51 professores no seu corpo docente (37 dos quais bolsistas de produtividade do CNPq, sendo 14 nível 1 e 23 nível 2) , o PPGC tem participação constante nos principais eventos internacionais e nacionais da área, projetos de cooperação com Universidades da Europa e Estados Unidos. O Programa é sediado no Instituto de Informática da UFRGS, reconhecido centro de excelência em pesquisa e ensino, dispondo de ótima infra-estrutura de laboratórios e de biblioteca. Mais informações em http://ppgc.inf.ufrgs.br		clone
#content :nth-child(2) .entry p:nth-child(1)	Sobre o PPGC- UFRGS		clone
#content :nth-child(2) .entry p:nth-child(1)	Nos últimos anos, todos os alunos de mestrado e doutorado do PPGC foram contemplados com bolsas de estudo de agencias de fomento e/ou empresas. Mais informações sobre disponibilidade e valores de bolsas podem ser obtidas através do e-mail ppgcapl@inf.ufrgs.br		

Tabela 7.8: Regras para a página “eleicoes.html” do estudo de caso.

Seletor CSS	Novo Texto	Link	Opção
.post:nth-child(2) , #content :nth-child(4), .meta, #sidebar .alignleft			
#content p:nth-child(1)	Divulgação dos resultados: 03/12/2010 (sexta)		clone
#content p:nth-child(1)	Eleições: 1º e 02/12/2010 (quarta e quinta)		clone
#content p:nth-child(1)	Divulgação dos inscritos: 26/11/2010 (sexta)		clone
#content p:nth-child(1)	Inscrições: 22/11 a 25/11/2010 (segunda a quinta)		clone
#content p:nth-child(1)	Segunda etapa: eleição dos membros do CONINF, dos membros do Colegiado do INA, da Comissão do PPGC e dos membros do Conselho Diretor do CEI.		clone
#content p:nth-child(1)	Divulgação dos resultados: 19/11/2010 (sexta)		clone
#content p:nth-child(1)	Eleições: 17 e 18/11/2010 (quarta e quinta)		clone
#content p:nth-child(1)	Divulgação dos inscritos: 12/11/2010 (sexta)		clone
#content p:nth-child(1)	Inscrições: 08 a 11/11/2010 (segunda a quinta)		clone
#content p:nth-child(1)	Primeira etapa: eleição para Chefe e Chefe Substituto do INA e do INT, Coordenador e Coordenador Substituto do PPGC e membros das Comissões CIC, ECP, Pesquisa e Extensão.		clone
#content p:nth-child(1)	CALENDÁRIO ELEITORAL:		Clone

Tabela 7.9: Regras para a página “pos_graduacao.html” do estudo de caso.

Seletor CSS	Novo Texto	Link	Opção
#content :nth-child(4) , #content :nth-child(4) .title, #content :nth-child(4) .entry, p:nth-child(1), .date, .post:nth-child(2), #content :nth-child(2) .title			
.alignleft			
#content p:nth-child(1)	Site da Especialização e Extensão do Instituto de Informática.	http://labcom.inf.ufrgs.br/especializacao	clone
#content p:nth-child(1)	Site do PGIE - Programa de Pós Graduação em Informática na Educação	http://www.pgie.ufrgs.br/	clone
#content p:nth-child(1)	Site do PGMICRO - Programa de Pós Graduação em Microeletrônica.	http://www.inf.ufrgs.br/pgmicro	clone
#content p:nth-child(1)	Site do PPGC - Programa de Pós Graduação em Computação.	http://ppgc.inf.ufrgs.br	clone
#content p:nth-child(1)	O programa de Pós-graduação em Informatica na Educação também conta com a participação de professores do Instituto de Informatica. Este programa promove a pesquisa sobre o desenvolvimento de métodos educacionais modernos e é um centro de referencia em educação à distância.		clone
#content p:nth-child(1)	Os professores e os estudantes participam regularmente dos principais eventos internacionais. Alguns professores do Instituto de Informatica também participam do programa de Pós-graduação em Microeletrônica (PGMicro). Como resultado desta cooperação, em 2008 foi criado o primeiro centro de treinamento brasileiro em projetos de circuitos integrados (VLSI).		clone
#content p:nth-child(1)	O PPGC - Programa de Pós-graduação em Computação sedia vários grupos de pesquisa. O programa conta com 43 professores e mais de 250 estudantes de mestrado e de doutorado. O PPGC prioriza os estudantes com dedicação integra. Todos os estudantes com dedicação integral recebem bolsas de estudo de agencias brasileiras. Desde 1973, o PPGC graduou mais de 1100 mestres e 150 doutores. Atualmente é um dos cinco programas brasileiros classificados como de classe internacional pelo Ministério da Educação do Brasil.		
.title	Pós-graduação		