UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ESPECIALIZAÇÃO EM SISTEMAS DE COMPUTAÇÃO PARA A WEB

SOFTWARE LIVRE PARA GESTÃO DE INFORMAÇÕES DOCUMENTAIS SOBRE LEGISLAÇÃO

MONOGRAFIA DE ESPECIALIZAÇÃO

EVERSON LUIS ROSA LUCION

Santa Maria, RS, Brasil

2007

SOFTWARE LIVRE PARA GESTÃO DE INFORMAÇÕES DOCUMENTAIS SOBRE LEGISLAÇÃO

por

Everson Luis Rosa Lucion

Monografia apresentada ao Curso de Especialização em Sistemas de Computação para a Web da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Especialista em Sistemas de Computação para a Web.

Orientador: Prof. João Carlos Damasceno Lima

Santa Maria, RS, Brasil

2007

Fone (0xx)55 91538472; End. Eletr.: elucion@gmail.com

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso De Especialização em Sistemas De Computação para a Web

A Comissão Examinadora, abaixo assinada,
aprova a Monografia de Especialização

SOFTWARE LIVRE PARA GESTÃO DE INFORMAÇÕES DOCUMENTAIS SOBRE LEGISLAÇÃO

elaborada por
Everson Luis Rosa Lucion

como requisito parcial para obtenção do grau de
Especialista em Sistemas de Computação para a Web

COMISÃO EXAMINADORA:

Prof. João Carlos Damasceno Lima
(Presidente/Orientador)

Andréa Schwertner Charão, Dr. (UFSM)

Célio Trois (UFSM)

Santa Maria, 31 de janeiro de 2007.

Agradecimentos

A Universidade Federal de Santa Maria, ao Centro de Tecnologia e a Coordenação do Curso de Ciência da Computação, pela oportunidade oferecida.

Aos professores do Curso de Sistemas de Computação para a Web, pelo comprometimento com a educação e determinação.

Ao Professor João Carlos Damasceno Lima, orientador, por apostar em novos desafios, propiciando ao aluno a idéia do "novo" como fator de crescimento profissional e intelectual.

RESUMO

Monografia de Especialização
Curso de Especialização em Sistemas de Computação para a Web
Universidade Federal de Santa Maria

SOFTWARE LIVRE PARA GESTÃO DE INFORMAÇÕES DOCUMENTAIS SOBRE LEGISLAÇÃO

AUTOR: EVERSON LUIS ROSA LUCION
ORIENTADOR: JOÃO CARLOS DAMASCENO LIMA
Data e Local da Defesa: Santa Maria, 31 de janeiro de 2006.

Este trabalho apresenta um estudo sobre Sistemas de Gerenciamento de Conteúdo na *Web*, usando software livre, propondo-se a pesquisar e desenvolver uma ferramenta de software livre, que auxilie na gestão de documentos sobre legislação educacional, integrando informações produzidas nas diferentes esferas governamentais de forma transparente, mantendo as características de autonomia e distribuição das informações. Explorando as características de distribuição e autonomia, este projeto tem como objetivo, desenvolver a ferramenta com tecnologias modernas e inovadoras para pesquisa de informações documentais sobre pareceres e legislação educacional. Pretende-se estabelecer os passos necessários para implementação desta abordagem e levantar um modelo de desenvolvimento de sistemas de gestão de conteúdos. Foi realizado um estudo de caso, tendo como escopo o domínio de conhecimento de um processo de desenvolvimento software. Como resultado, tem-se um processo de geração, composto das seguintes atividades: (1) Modelagem Independente de Plataforma, (2) Definição das Tecnologias e Ferramentas, (3) Implementação do Sistema, (4) Apresentação do Sistema e (5) Conclusões. Conclui-se que: Um dos pontos fortes deste trabalho são os produtos, com a criação dos Produtos Colegiado e Legislação, é possível gerenciar a documentação e agilizar a pesquisa. Outro aspecto importante dos frameworks de gerenciamento de conteúdo usados é a customização, trazendo um grau elevado de personificação dos ambientes. O sistema tenta mostrar as pessoas que gerenciar a informação pode ser um processo simples e que trará um alto grau de satisfação para aqueles que dependem dessas informações.

Palavras-chaves: *Web*, Gerenciamento de Conteúdo, Software Livre

ABSTRACT

Specialization Monograph
Course of Specialization in Computer Systems for the Web
Universidade Federal de Santa Maria, RS, Brazil

FREE SOFTWARE FOR MANAGEMENT OF DOCUMENTARY INFORMATION ON LEGISLATION

AUTHOR: EVERSON LUIS ROSA LUCION
SUPERVISOR: JOÃO CARLOS DAMASCENO LIMA
Date and Place of Defense: January 31th, 2007, Santa Maria.

This work presents a study on Web Content Management Systems by using free softwares. One intends to research and to develop a free software tool that could assist the management of educational legislation documents by integrating pieces of information produced transparently in different governmental spheres in order to keep the characteristics of autonomy and information distribution. By exploring the characteristics of distribution and autonomy, this project aims to develop a tool with both modern and innovative technologies to research documentary information about statutes and educational legislation. One intends to establish the steps needed to the implementation of this approach as well as to raise a development model of content management systems. A case study was carried out taking into account the domain of knowledge of a software development process. As a result, we have a process of generation that includes the following activities: (1) Platform-independent Model, (2) Definition of Technologies and Tools, (3) System Implementation, (4) System Presentation and (5) Conclusions. We concluded that products are one of the strongest points of this work. In addition, it is possible to manage the documentation and to speed up research through the creation of Collegiate and Legislation Products. Another important aspect of the frameworks of the used content management is customization that brings a high degree of environment personification. The system try to show to people that managing information can be a simple process which will proportionate a high degree of satisfaction for those who depend on these pieces of information.

Keywords: Web, Content Management, Free Software

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

BSD	*Berkeley* Software *Distribuition*
CFE	Conselho Federal de Educação
CMF	*Content Management Framework*
CMS	*Content Management Systems*
CNE	Conselho Nacional de Educação
DTML	*Document Template Markup Language*
GED	Gestão Eletrônica de Documentos
GPL	*GNU General Public License*
HTML	*HiperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
INEP	Instituto Nacional e Estudos e Pesquisas em Educação Anísio Teixeira
JSP	*JavaServer Pages*
MEC	Ministério da Educação e Cultura
PERL	*Practical Extraction and Report Language*
PHP	*Hypertext Preprocessor*
ProLEI	Programa de Legislação Educacional Integrado
TI	Tecnologia de Informação
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
WWW	*World Wide Web*
XML	*Extensible Markup Language*
ZMI	*Zope Manage Interface*
ZODB	*Zope Object DataBase*
ZOPE	*Z Object Publishing Environment*
ZPT	*Zope Page Tamplates*

SUMÁRIO

INTRODUÇÃO

A disponibilização de informações governamentais na Internet possibilitou aos cidadãos o acompanhamento das decisões e dos atos normativos que regulam as atividades da sociedade. Esse aumento de transparência das atividades governamentais produz uma grande quantidade de documentos que devem ser gerenciados, catalogados, indexados e controlados quanto a sua periodicidade (LOPES, 2000), (BALASUBRAMANIAN, 1998).

A gestão documental de informações sobre educação é um bom exemplo da aplicação dos conceitos citados anteriormente, mas muito há para ser realizado neste setor que possui gestão parcial sobre legislação. De fato, atualmente, só na esfera federal, existem diversos locais que produzem informações sobre legislação educacional, sendo o Ministério da Educação (MEC) um dos mais importantes. Essa legislação produzida no MEC e as demais produzidas no Governo Federal e Congresso Nacional são catalogadas e indexadas no Sistema ProLEI, que está disponível na Internet e é mantido pelo Instituto Nacional e Estudos e Pesquisas em Educação Anísio Teixeira (INEP, 2006).

Já as informações documentais de legislação e de pareceres do Conselho Nacional de Educação (CNE) estão parcialmente disponíveis, uma vez que não existe nenhum mecanismo de classificação e indexação. O processo de localização ou busca propriamente dito, normalmente, é efetuado através de ligações telefônicas ao Conselho. Essa realidade não é exclusividade do CNE, ocorrendo também nos demais conselhos de educação, nas esferas estaduais e municipais.

Outro setor que também padece com problemas semelhantes são os conselhos das universidades, que decidem o dia-a-dia das novas diretrizes da educação das mais importantes instituições educacionais do país.

A falta de um mecanismo eficiente e de baixo custo torna as informações produzidas por esses conselhos difíceis de serem localizadas e consultadas, inviabilizando a efetividade e a transparência das decisões tomadas nas câmaras dos conselhos.

Nesse contexto até aqui delineado, fica evidente a necessidade de se aprimorar e modernizar os instrumentos de gestão de documentos sobre legislação educacional. Em particular, é importante agilizar a manipulação desses documentos em todo o seu ciclo de vida, ou seja, desde a sua criação até a sua utilização nas diversas esferas de governo e nos demais setores envolvidos.

Existe, atualmente, uma ampla oferta de softwares para Gestão Eletrônica de Documentos (GED), ou seja, de ferramentas que auxiliam no armazenamento, localização e recuperação de informação em documentos digitalizados ou criados em meio digital (BALDAM, 2002). No entanto, a maior parte das soluções disponíveis é sujeita a licenças de software proprietárias, que restringem as liberdades de uso e adaptação das ferramentas e, geralmente, têm custos de aquisição e manutenção proibitivos para muitos órgãos governamentais.

A este projeto tem como referência o ProLEI - Programa de Legislação Educacional Integrado – que, atualmente, é mantido pelo INEP. Esse programa tem uma das mais altas taxas de utilização dentro do sítio *Web* do INEP, sendo realizados aproximadamente 50.000 acessos mensais às informações sobre legislação educacional na esfera federal. Com essa experiência em desenvolvimento de sistemas com alta disponibilidade e sobre legislação, esta equipe procura, agora, desenvolver uma solução de baixo custo e bom desempenho para completar o elenco de informações documentais sobre legislação educacional.

Este projeto possui outras características inovadoras:

[1] utilização extensiva de ferramentas de software livre;

[2] utilização de padrões de software aberto para integração e intercomunicação.

A relevância deste projeto no contexto nacional ficará evidenciada através da utilização desta ferramenta nos demais setores da sociedade e nos conselhos de educação a nível estadual e municipal.

Este projeto tem como objetivos a pesquisa e o desenvolvimento de uma ferramenta de software livre para gestão de informações documentais distribuídas, utilizando tecnologias modernas e inovadoras, integrando informações das diferentes esferas governamentais de forma transparente. O foco principal desta ferramenta é o conjunto de informações inter-relacionadas que normatizam o sistema educacional brasileiro em suas diversas esferas. Com este projeto, espera-se aprimorar o processo da disseminação dessas informações e, ao mesmo tempo, oferecer uma solução tecnológica baseada nos princípios de autonomia e distribuição das informações.

Os objetivos específicos deste projeto são:

- Disponibilizar as informações documentais dos pareceres e legislação educacional via software *Web*;
- Desenvolver *o* software sob licença de software livre e
- Publicar os resultados obtidos.

Assim, propõe o desenvolvimento de uma ferramenta que integre as informações sobre legislação de forma transparente e com características de autonomia e distribuição. Isso permitirá que a desejada efetividade e transparência dos conselhos chegue aos seus maiores interessados, os cidadãos. Para a efetivação do projeto, prevê-se a implantação deste sistema no colegiado do Curso de Ciência da Computação da UFSM.

1 REVISÃO DA LITERATURA

1.1 *Software* Livre

O Software Livre como movimento organizado teve início em 1983, quando Richard Stallman deu início ao Projeto *GNU* e, posteriormente, à *Free* Software *Foundation* (CAMPOS, 2006).

Software livre caracteriza-se pela liberdade aos usuários, para não somente utilizar seus produtos, mas também para executar, copiar, estudar e modificar. É importante compreender que livre não quer dizer necessariamente que o software é gratuito.

Tendo como base a definição da Fundação para o Software Livre: entende-se este como qualquer programa de computador que pode ser usado, copiado, estudado, modificado e redistribuído sem restrições. Liberdade de tais restrições é central ao conceito, o qual se opõe ao conceito de software proprietário, mas não ao software que é vendido almejando lucro (software comercial).

A maneira usual de distribuição de software livre é anexar a este uma licença de software livre e tornar o código fonte do programa disponível (WIKIPÉDIA, 2006a).

Um software é tido como livre quando atende aos quatro tipos de liberdade para os usuários do software definidas pela *Free* Software *Foundation:*

- Liberdade de executar o programa, para qualquer propósito (liberdade nº 0);
- Liberdade de estudar como o programa funciona, e adaptá-lo às suas necessidades (liberdade nº 1). Acesso ao código-fonte é um pré-requisito para essa liberdade;
- Liberdade de redistribuir cópias de modo que se possa ajudar ao seu próximo (liberdade nº 2);
- Liberdade de aperfeiçoar o programa e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie (liberdade nº 3). Acesso ao código-fonte é um pré-requisito para esta liberdade.

Um programa é software livre se os usuários tem todas essas liberdades (CAMPOS, 2006).

O usuário deve ser livre para redistribuir cópias, com ou sem modificações, de graça ou cobrando uma taxa pela distribuição, para qualquer um em qualquer lugar. Ser livre significa (entre outras coisas) que não há necessidade de pedir ou pagar pela permissão, uma vez que esteja com o programa.

O usuário deve também ter a liberdade de fazer modificações e usá-las para uso pessoal no seu trabalho ou lazer, sem nem mesmo mencionar que elas existem. Se for publicadas as modificações, o usuário não deve ser obrigado a avisar a ninguém em particular, ou de nenhum modo em especial (CAMPOS, 2006).

A liberdade de usar um programa significa a liberdade para qualquer tipo de pessoa física ou jurídica utilizar o software em todo o tipo de sistema computacional, para qualquer tipo de uso, sem que seja necessário dar ciência ao desenvolvedor ou a qualquer outra entidade em especial. A liberdade de redistribuir cópias deve incluir formas binárias ou executáveis do programa, assim como o código-fonte, tanto para as versões originais quanto para as modificadas. De modo que a liberdade de fazer modificações e de publicar versões aperfeiçoadas tenha algum significado, deve-se ter acesso ao código-fonte do programa. Portanto, acesso ao código-fonte é uma condição necessária ao software livre (CAMPOS, 2006).

Para que essas liberdades sejam reais, elas têm que ser irrevogáveis desde que não se faça nada errado; caso o desenvolvedor do software tenha o poder de revogar a licença, mesmo que não se tenha dado motivo, o software não é livre (CAMPOS, 2006).

1.2 Copyleft

Segundo o *site* da *Free* Software *Foundation – www.psf.org* - "O *copyleft* diz que qualquer um que distribui o *software*, com ou sem modificações, tem que passar adiante a liberdade de copiar e modificar o programa. *Copyleft* é uma extensão das 4 liberdades básicas e ocorre na forma de uma obrigação. O *copyleft* garante que todos os usuários tem liberdade". Quem recebeu um software com uma licença livre que inclua cláusulas de *copyleft*, e se optar por redistribui-lo (modificado ou não), terá que mantê-lo com a mesma licença com que o recebeu. Nem todas as licenças de software livre incluem a característica de *copyleft*. A licença *GNU GPL* (*kernel Linux*) é o maior exemplo de uma licença *copyleft*. Outras licenças livres, como a licença *BSD* ou a licença *ASL* (*Apache Software License*) não incluem a característica de *copyleft*. A figura 1 do *copyleft*, palavra que é um trocadilho com *copyright*, e cuja tradução aproximada seria "cópia permitida" (CAMPOS, 2006).

Figura 1 – *Copyleft*.

1.3 Software livre x Código aberto

No ano de 1998, um grupo de pessoas da comunidade e do mercado que gira em torno do software livre, não satisfeitos com a postura filosófica do movimento existente e acreditando que a condenação do uso de software proprietário é um instrumento que retarda, ao invés de acelerar, a adoção e o apoio ao software livre no ambiente corporativo, criou a *Open Source Initiative*, que adota o termo *Open Source* (Código Aberto) para se referir aos *softwares* livres. Tem uma posição voltada ao pragmatismo visando à adoção do software de código aberto como uma solução viável, com menos viés ideológico que a *Free* Software *Foundation (*CAMPOS, 2006).

Código Aberto não quer dizer simplesmente ter acesso ao código-fonte dos softwares (e não necessariamente acompanhado das "4 liberdades" do software livre). Para uma licença ou software ser considerado como Código Aberto pela *Open Source Initiative*, eles devem atender aos 10 critérios da "Definição de Código Aberto", que incluem itens como livre redistribuição, permissão de trabalhos derivados, não discriminação, distribuição da licença e outros. Os termos de distribuição do **software de código aberto** devem estar de acordo com os seguintes critérios (OpenIP, 2006):

- 1. Redistribuição livre: A licença não deve restringir as partes de vender ou integrar o software como componente de uma distribuição de software agregada, contendo programas oriundos de diversas fontes. A licença não deve exigir *royalties*[1] ou qualquer outro tipo de custo para venda;

[1] *Royalty* (Palavra inglesa) - Importância cobrada pelo proprietário de uma patente de produto, processo de produção, marca, entre outros, ou pelo autor de uma obra, para permitir seu uso ou comercialização. Plural: royalties.

• 2. Código fonte: O programa deve incluir o código fonte e precisa permitir a distribuição na forma de código fonte bem como compilada. Quando alguma forma do produto não é distribuída com o código fonte, precisa dispor de meios reconhecidos de obtenção do código por não mais que um preço de custo razoável para a reprodução ou *download* pela Internet sem custos. O código fonte deve ser a forma privilegiada na qual um programador altera o programa. Código fonte deliberadamente obscurecido não é aceito. Formas intermediárias, como a saída de um processador ou tradutor não são permitidas;

• 3. Trabalhos derivados: A licença deve permitir modificações e trabalhos derivados, e precisa garantir a estes a distribuição sob os mesmos termos como a licença do software original;

• 4. Integridade do autor do código fonte: A licença deve restringir a distribuição do código fonte sob forma alterada somente se permitir a distribuição de "*patch files*" com código para o propósito de modificar o programa quando de sua compilação. A licença deve permitir explicitamente a distribuição do software compilado a partir de um código modificado. A licença pode exigir que trabalhos derivados tragam um nome ou versão distintos da original;

• 5. Sem discriminações quanto a pessoas ou grupos: A licença não deve discriminar qualquer pessoa ou grupo de pessoas;

• 6. Sem discriminação quanto a ramos de empreendimento: A licença não deve restringir nenhum uso a ramos de empreendimento específicos. Por exemplo, ele não pode ser vedado para uso comercial ou para usos em pesquisas genéticas;

• 7. Distribuição da licença: Os direitos associados ao programa devem ser aplicáveis para todos aos quais o programa é redistribuído sem a necessidade de licenças adicionais a estas partes para sua execução;

• 8. A Licença não deve ser específica ao produto: Os direitos associados ao programa não devem depender deste ser parte de uma distribuição particular de software. Caso o programa seja extraído desta distribuição e usado ou distribuído através dos termos de sua licença, todas as partes para as quais o programa é redistribuído devem gozar dos mesmos direitos garantidos na conjunção original da distribuição do software;

• 9. A licença não deve restringir outros softwares: A licença não deve aplicar restrições sob outros softwares que sejam distribuídos com softwares objetos de seu licenciamento. Por exemplo, a licença não deve insistir que outros programas distribuídos no mesmo meio sejam software de código aberto;

• 10. A licença deve ser neutra às tecnologias: Nenhuma aplicação da licença deve preterir uma tecnologia específica ou estilo de interface.

De maneira geral, as licenças que atendem à já mencionada Definição de Software Livre (da *Free* Software *Foundation*) também atendem à "Definição de Código Aberto" (da *Open Source Initiative*). Pode-se dizer (na ampla maioria dos casos, ao menos) que se um determinado *software* é livre, ele também é de código aberto e vice-versa.

A diferença prática entre a *Free* Software *Foundation* e a *Open Source Initiative* está em seus objetivos, filosofia e modo de agir e não nos *softwares* ou licenças.

Para o Movimento do Software Livre, que é um Movimento Social, não é ético aprisionar conhecimento científico, que deve estar disponível sempre, para permitir assim a evolução da humanidade. Já o Movimento pelo Código Aberto, que não é um Movimento Social, mas voltado ao Mercado, prega que o Software desse tipo traz diversas vantagens técnicas e econômicas. Este segundo movimento surgiu para levar as empresas a adotarem o modelo de desenvolvimento de Software Livre (CAMPOS, 2006).

1.4 Licenças de *software* livre

Há muitas licenças de software livre. Nada impede (embora não seja recomendado) que cada interessado(a) crie sua própria licença que atenda às 4 liberdades básicas, agregando ou não uma cláusula de *copyleft.*

A *Free* Software *Foundation* mantém um site (*http://www.gnu.org/licenses/license-list.pt.html*), classificando-as entre livres (compatíveis ou não com a *GPL*) e não-livres, incluindo comentários sobre elas.

Algumas das licenças livres mais populares são:

• *GPL* ou *GNU General Public License* (ver também a *GPL* em português e a *CC GPL* no site do Governo Brasileiro); Licença BSD ;

• *MPL* ou *Mozilla Public License*;

• *Apache License.*

Alguns softwares livres excelentes são o *Linux*, o ambiente gráfico *KDE*, o compilador *GCC*, o servidor *web Apache*, o *OpenOffice.org* e o navegador *web Firefox*, entre outros (CAMPOS, 2006).

1.5 Gerenciamento de Conteúdo na Internet

Diariamente grandes e pequenas empresas, organizações governamentais e não governamentais buscam a utilização de sistemas para facilitar a atualização de conteúdo centralizando em um departamento de pessoal, por exemplo. Uma das razões disso é remover um dos passos no processo de atualização do *site*, reduzindo custos com terceiros e deixando a responsabilidade nas mãos do responsável pelo próprio conteúdo. Tornando este processo simples e eficiente.

O que se busca com o Gerenciamento de Conteúdo na Internet é permitir que o conteúdo de *website* possa ser modificado de forma rápida e segura de qualquer computador conectado à Internet, reduzindo custos e ajudando a suplantar barreiras potenciais à comunicação *web*, reduzindo o custo da criação, contribuição e manutenção de conteúdo, agregando alto valor ao controle do gerenciamento e controle das informações da organização em questão.

No início da Internet, o conteúdo dos *sites* era na maioria das vezes inteiramente desenvolvido por código *Hypertext Transfer Protocol* (*HTML)* estático e documentos de texto. Um documento estático é qualquer página que seja salva em unidade de armazenamento permanente de dados e disponibilizada através de um navegador *web* sem sofrer nenhuma mudanças.

Num *site* estático, todo o conteúdo das páginas é alimentado manualmente por colaboradores e construtores de *sites*, podendo ser pessoas da própria organização ou terceiros. Esse processo é também conhecido como construção de página *web* em tempo de projeto, pois as páginas são construídas na sua totalidade enquanto o *site* está sendo desenvolvido. *Sites* como estes são desenvolvidos e mantidos por profissionais experientes com conhecimento em linguagens para a *web*. De uma maneira geral, o custo inicial é menor, todavia todas as futuras mudanças e atualizações têm que ser realizadas por profissionais *Web*. Diante de tudo isso, um *site* estático poderá ser mais oneroso para manter, especialmente quando são necessárias alterações freqüentes nas páginas (COSTA, 2004).

Em um *site* dinâmico, tudo é construído no momento em que o mesmo é requisitado por um navegador. *Sites* dinâmicos também são desenvolvidos por colaboradores e construtores de *sites*, mas podem ser mantidos diretamente pelo usuário-cliente. O seu custo inicial é mais alto, bem como o tempo de desenvolvimento, mas não é necessário pagar um profissional para fazer as freqüentes alterações necessárias nas páginas. Uma página *Web* dinâmica tem seu conteúdo interpretado pelo servidor no momento em que for requisitada.

Linguagens de programação normalmente utilizadas para a construção de *sites* dinâmicos são o *PHP, JSP, PYTHON* entre outras (COSTA, 2004)..

Um exemplo de *sites* dinâmicos são aqueles em que o conteúdo do mesmo está armazenado em um banco de dados e no momento em que é requisitado pelo navegador do cliente, o servidor monta a página e envia para o cliente. Normalmente, *sites* desse tipo visam reduzir o trabalho de manutenção das páginas, como na maioria das vezes os usuários que não conhecem a linguagem *HTML*, os mesmos alimentam o banco de dados que por sua vez é lido pelo servidor e devolvido o seu conteúdo ao cliente.

Um *site* pode conter conteúdo estático e dinâmico, onde a informação que não muda freqüentemente é mais bem manipulada estaticamente, contudo a informação que sofre alterações freqüentes é mais bem manipulada dinamicamente. Esse modo de desenvolvimento misto permite que os desenvolvedores concentrem a maior parte de seu tempo em alimentar o site com conteúdo e menos tempo em manutenção. Um *site* ou sistema criado e gerenciado por um Sistema de Gerenciamento de Conteúdo na *Web* é caracterizado como um *site* dinâmico.

1.6 Sistemas de Gerenciamento de Conteúdo *Web*

CMS é a expressão utilizada para descrever ferramentas que promovem meios de gerenciamento, publicação e manutenção de informações. Atualmente, essas ferramentas são utilizadas na construção de portais *Web*, vindo às características dos sistemas *CMS* ao encontro das capacidades desejadas para tal sistema (RIZZETTI ET ALL, 2005).

Um *Content Management System* (*CMS*), tem por objetivo específico o de estruturar e facilitar a criação, administração, distribuição, publicação e disponibilidade da informação (ZANIRATO, 2006).

Os sistemas *CMS* promovem um aumento de produtividade referente à implementação de portais *Web*, disponibilizando ferramentas que satisfazem as necessidades gerais de um portal; além disso, em função de ser um sistema de gerência de conteúdo, pode ser aplicado aos mais diversos fins. Os principais objetivos do *CMS* são permitir fácil criação, publicação e retorno de conteúdo que se ajuste às necessidades requeridas.

As ferramentas *CMS* possibilitam uma forma fácil e padronizada de publicar conteúdo na *Web* e distribuí-lo. Além disso, promove uma abstração do conteúdo a ser publicado, da forma como ele é publicado. Ou seja, para publicar informações através do portal, o usuário não necessita ter conhecimentos sobre qualquer linguagem de programação, basta que ele

utilize o conjunto de ferramentas que acompanham o sistema *CMS*. Dessa forma, por exemplo, membros que atuam em um projeto podem disponibilizar informações a respeito do seu trabalho, independente dos seus conhecimentos sobre linguagens de programação.

Outras características dos sistemas *CMS* são:

- estrutura para *workflow;*

- instâncias públicas e privadas, para dessa forma efetuar o controle de publicação de conteúdo;

- sistema de fórum;

- listas de discussões;

- carga de arquivos;

- modelos para construção de páginas.

Para exemplificar como o *CMS* é usado deve-se imaginar um portal de notícias onde artigos são diariamente publicados a cada instante, onde um ***webmaster*** é encarregado de estar programando e publicando no *website* todos artigos recém criados.

O editor passa para o aprovador que por sua vez repassa para o *webmaster*, após esse processo finalmente o artigo é inserido no *website*. A publicação dos artigos no site pode demorar várias horas ou até dias, tornando-se bastante trabalhosa e demorada, pois apenas o *webmaster* está fazendo a publicando os artigos.

Através do uso do *CMS*, o **editor** poderá economizar rotinas de trabalho, publicar seus artigos em tempo real e sem necessidade de estar em seu local de trabalho. O ***editor*** por sua vez, não precisa ter conhecimentos para trabalhar com ferramentas de criação de *sites* e linguagens de programação, apenas precisa saber como publicar seus artigos em um gerenciador de conteúdo. O **editor** publica o conteúdo diretamente, o sistema já auto implementa data/hora e nome do autor.

Nas publicações, além de textos o **editor** também pode adicionar figuras nos seus artigos. O mesmo não precisa ter conhecimento de como editar imagem ou códigos para inserir uma imagem na sua publicação, ele apenas indica a imagem de algum endereço da máquina local para serem publicadas em seu artigo.

Baseando-se na publicação de conteúdo, aprovação e administração, o processo de utilização do ***CMS*** é bem prática, a topologia abrange desde simples usuário até administradores experientes conforme se pode observar na figura 2 (ZANIRATO, 2006).

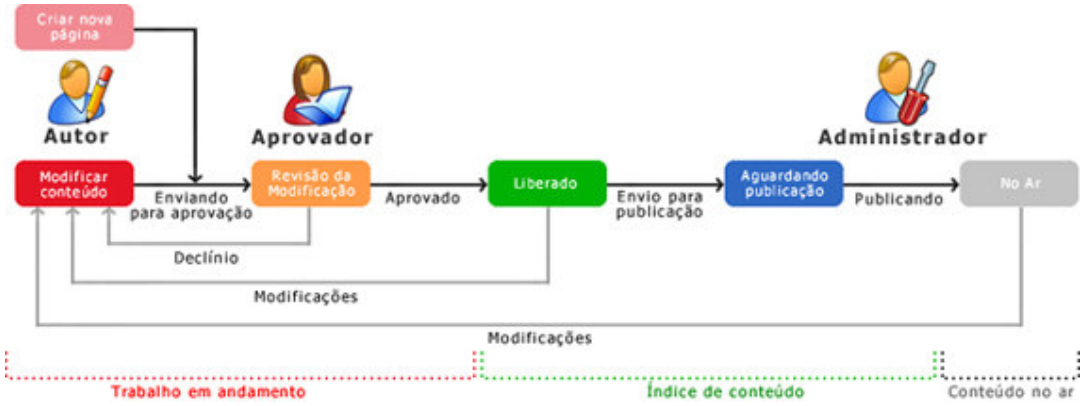

Figura 2 – Topologia do *CMS*.

Existem sistemas *CMS* baseados nas mais variadas linguagens de programação, sendo freqüentemente encontrados nas linguagens *Java, PHP, Python e Perl* . Um dos Sistemas Gerenciadores de Conteúdo livre e de código aberto mais usado atualmente é o *Plone* (WIKIPÉDIA, 2006b). O *Plone* é um *CMS*, baseado em (*Framework* de Gerenciamento de Conteúdo) *Content Management Framework* (*CMF*), que roda no *Zope* (servidor de aplicação para *Web* de código aberto, escrito em *Phyton*), mas com seus próprios conjuntos de *templates* e tipos de arquivos.

1.7 Servidores de Aplicações

Application servers, ou servidores de aplicação, são softwares que fornecem a infra-estrutura de serviços para a execução de aplicações distribuídas. Os servidores de aplicação são executados em servidores e são acessados pelos clientes através de uma conexão de rede. As vantagens dos servidores de aplicação em relação ao modelo cliente/servidor residem nos serviços implementados por eles e disponíveis aos desenvolveres, fazendo com que eles possam concentrar a maior parte do tempo no desenvolvimento da lógica de negócio. Em geral estes serviços diminuem a complexidade do desenvolvimento, controlam o fluxo de dados, incrementam a performance, gerenciam a segurança. O servidor de aplicação predispõe a utilização da arquitetura chamada de 3-camadas ou n-camadas, que permite um melhor aproveitamento das características de cada componente (servidor de banco de dados, servidor de aplicação e cliente). A primeira camada, chamada Front-End, usualmente são *browsers*, que servem para apresentação e algumas validações. A segunda camada, é a aplicação sendo executada no servidor de aplicação. A terceira camada é o servidor de banco de dados. Os

servidores de aplicação priorizam o compartilhamento de componentes e aplicações, fazendo assim com que seja mais fácil o desenvolvimento, manutenção e gerenciamento de sistemas complexos.

Além das características já citadas, outros serviços também estão disponíveis nos servidores de aplicação:

- Tolerância à falhas: através de políticas para recuperação e distribuição de componentes em clones dos servidores;
- Balanceamento de carga: com a análise da carga nos servidores permite a distribuição de clientes de forma maximizar a utilização dos recursos disponíveis;
- Gerenciamento dos componentes: através de ferramentas para a manipulação de componentes e serviços, tais como gerenciamento de sessão, notificação, distribuição da lógica de negócios;
- Gerenciamento de transações: garante a integridade da transação em ambientes distribuídos;
- Console de gerenciamento: permite o gerenciamento de vários servidores de aplicação através de um único sistema gráfico;
- Segurança: garante a segurança da aplicação.

Existem várias implementações de servidores de aplicação, em sua maioria implementados na plataforma Java, como exemplos podemos citar *IBM WebSphere Application Server, Oracle Oracle9i Application Server, BEA WebLogic, SUN iPlanet*. Outras implementações existem em outras plataformas, como o *Apple WebObjects* que roda em *MacOS* e o *Zope Application Server* que roda sobre a linguagem *Python*. Em geral, os servidores de aplicação rodam em vários sistemas operacionais, como *Solaris*, *Linux* e *Windows*, o que permite que seja possível o desenvolvimento em uma plataforma e sua publicação para produção em outra. Assim a máquina utilizada no desenvolvimento pode ter um custo bem inferior à de produção (I-WEB, 2007).

2 METODOLOGIA

Este projeto se propõe a pesquisar e desenvolver uma ferramenta de software livre que auxilie na gestão de documentos sobre legislação educacional, mantendo as características de autonomia e distribuição das informações, explorando as características de distribuição e autonomia de pareceres e legislação educacional, em esferas educacionais como o CNE e CFE e Conselhos dos Colegiados dos Cursos de Graduação das Universidades de Ensino Superior. Para atender a diferentes órgãos com procedimentos muito similares, mas com particularidades específicas, é preciso criar Produtos *CMF Plone* para cada caso.

Para atender os CNE e CFE, definiu-se para criação de um produto CMF *Plone* denominado "Legislação".

E para atender os Conselhos dos Colegiados dos Cursos de Graduação, definiu-se pela criação de um outro produto CMF *Plone* com o nome de "Colegiado".

2.1 Modelagem Independente de Plataforma

Para execução deste projeto, inicia-se com a definição do que cada produto precisa conter. Para modelar esse sistema será usado a *UML* (*Unified Modeling Language*).

A *UML* é uma linguagem para especificação, documentação, visualização e desenvolvimento de sistemas orientados a objetos. Sintetiza os principais métodos existentes, sendo considerada uma das linguagens mais expressivas para modelagem de sistemas orientados a objetos. Por meio de seus diagramas é possível representar sistemas de softwares sob diversas perspectivas de visualização facilitando a comunicação de todas as pessoas envolvidas no processo de desenvolvimento de um sistema - gerentes, coordenadores, analistas, desenvolvedores - por apresentar um vocabulário de fácil entendimento.

Será usada a Classe como elemento abstrato que representa um conjunto de objetos. Em programação, um diagrama de classes é uma representação da estrutura e relações das classes que servem de modelo para objetos.

Os Diagramas de Classes são compostos por: Atributos, operações e associações. A figura 3 apresenta o diagrama *UML* do sistema Legislação.

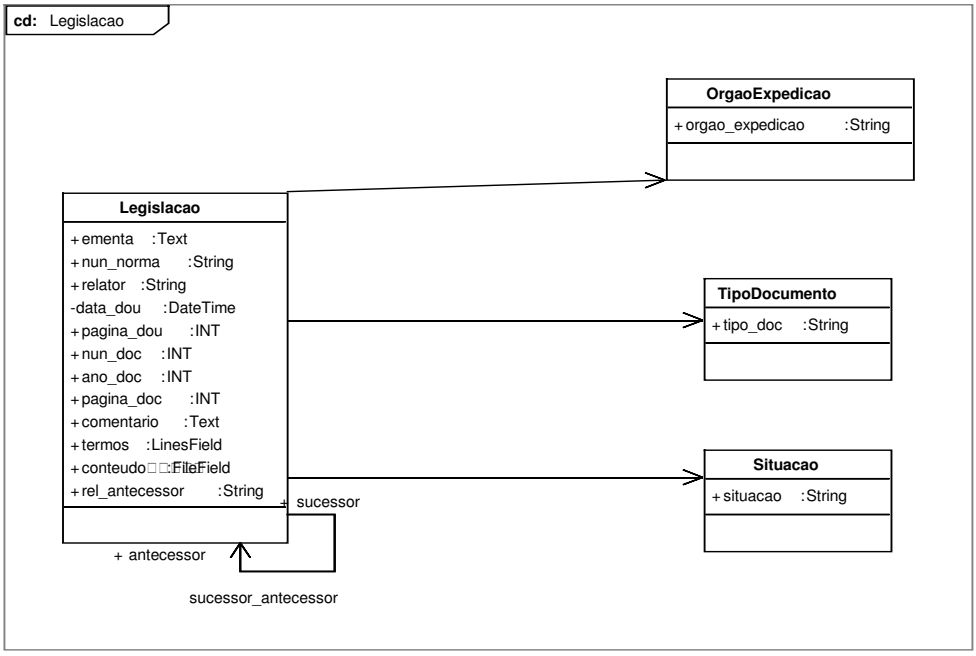

Figura 3 - Diagrama do sistema Legislação

Para buscar a idéia de Produto do *Plone*, desenvolveu-se um sistema minimalista a partir do diagrama de classes Legislação. Com isso, obtém-se um Produto que pode ser instalado/desinstalado no *Plone* com as funcionalidades embutidas. A figura 4 traz o novo diagrama de classe.

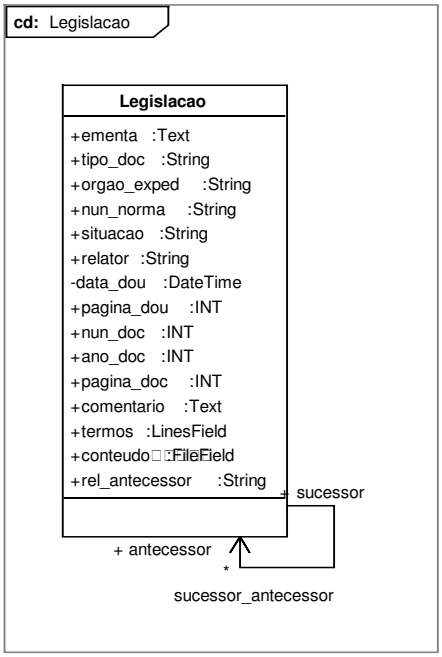

Figura 4 – Novo diagrama de classe Legislação.

A partir do modelo acima pode-se criar um Produto Legislação padrão e customizável para o *Plone*, onde o sistema incorporou funcionalidades e dados comuns ao CNE e ao CFE.

As classe OrgaoExpedicao, TipoDocumento e Situacao foram minimizadas para dentro da classe Legislação, onde cada uma tornou-se um atributo da mesma. Isso incorpora funcionalidades ao Produto, trazendo consigo Órgãos de Expedição, Tipos e Situação dos Documentos já pré-definidos, onde o usuário quando da digitação do conteúdo selecionará esses campos através de um menu *select* do tipo *top-dow*.

Seguindo o mesmo modelo de representação, foi desenvolvido o modelo do Produto Colegiado padrão, apresentado na figura 5.

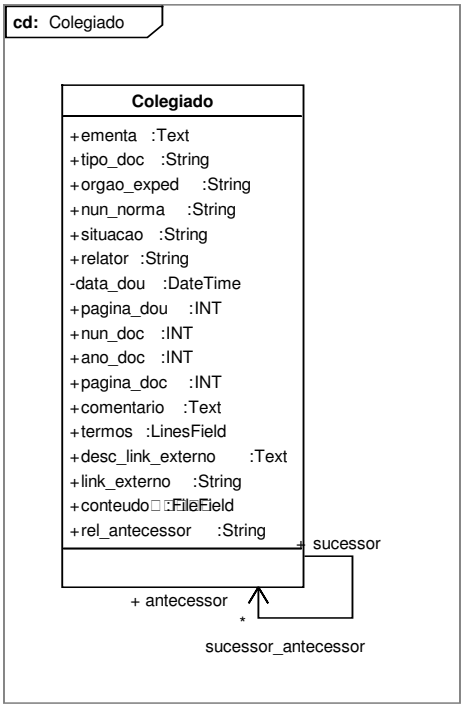

Figura 5 – Diagrama de classe Colegiado.

2.2 Definição das Tecnologias e Ferramentas

Após a definição da modelagem dos produtos, inicia-se a construção dos mesmos. Este capítulo aborda quais tecnologias e ferramentas estarão envolvidas para realização do desenvolvimento dos produtos Legislação e Colegiado.

O sistema é baseado em softwares de código aberto. Para execução deste projeto buscaram-se ferramentas ou sistemas que apoiassem às atividades necessárias à realização do processo. O sistema utilizou como principais atividades necessárias à realização do processo um conjunto de ferramentas descritas abaixo.

- Sistema operacional: *Kurumin Linux* 6.1 com *kernel* 2.6.17.6;
- Linguagem de programação: *Python* 2.4.4c0;
- Servidor de Aplicações: *Zope* 2.9.5-final;
- Ferramenta de modelagem UML: *Poseidon for UML Community Edition* 4.2.1;
- Gerador de código: *ArchGenXML* 1.5.0;
- Construtor de tipos: *Archetypes* 1.4.1-final;
- Ferramenta para remover *tags HTML* de *Strings: Striprogram* 1.5.0;
- Criação de *websites* multilíngües: *I18ndude* 2.1;
- Criar referências inversas: *ATBackRef* 0.1.

2.3 Implementação do Sistema

2.3.1 O Sistema Operacional

Inicialmente para a implementação do sistema foi adotado a plataforma *Linux Kurumin 6.1* com *kernel* 2.6.17.6. O Kurumin é a distribuição *Linux* desenvolvida pela equipe do Guia do *Hardware* e colaboradores, que se tornou rapidamente uma das distribuições *Linux* mais usadas no país. Todos os componentes do sistema são abertos, permitindo que além de usar, você possa redistribuí-lo, ver e modificar os scripts de configuração e desenvolver versões modificadas do sistema.

O Kurumin *Linux* é uma distribuição *Linux* baseada no *Knoppix*. O *Knoppix* é uma distribuição *GNU/Linux* baseado no *Debian* e gravado em *CD* bootável, dotado de um sistema de detecção automática de *hardware*, suporte para vários dispositivos gráficos, de som, *SCSI*, *USB* e outros periféricos. Ao dar o *boot* do *CD* do *Knoppix* o usuário não precisa instalar nada em seu disco rígido. Devido à sua descompressão, que acontece a partir do *CD*, estão disponíveis mais de 2 *GB* de aplicativos, desde aplicações de escritório eletrônico a ferramentas de sistema, o que permite usar um *CD* de *Knoppix* como uma demonstração do *GNU/Linux*, um *CD* educacional, um sistema de recuperação ou adaptado e usado como plataforma para demonstração de demos de softwares.

Debian é simultaneamente o nome de uma distribuição não comercial livre (gratuita e de código fonte aberto) de *GNU/Linux* (amplamente utilizada) e de um grupo de voluntários que o mantêm à volta do mundo. Uma vez que o *Debian* se baseia fortemente no projeto *GNU* (e a distribuição oficial do Projeto *GNU é Debian*), é usualmente chamado *Debian GNU/Linux*. O *Debian* é especialmente conhecido pelo seu sistema de gestão de pacotes, chamado *APT*, que permite: atualizações relativamente fáceis a partir de versões realmente antigas; instalações quase sem esforço de novos pacotes e remoções limpas dos pacotes antigos.

O ciclo de desenvolvimento das versões do *Debian* passa por três fases, todas desenvolvidas simultaneamente:

- "*Unstable*" - instável
- "*Testing*" - teste
- "*Stable*" - estável

Nesse sistema foi instalado a versão "*Testing*".

O *Kurumin* é baseado no *Knoppix*, no *Kanotix* e no *Debian*, portanto mantém compatibilidade com os pacotes *.deb* que podem ser encontrados nos *CDs* do *Debian* ou em vários outros lugares. Também é possível instalar programas automaticamente pela *Internet* usando *o apt-get.* Por exemplo: para instalar o *firefox*, basta executar o comando:

 # apt-get update (para atualizar as listas de instalação de programas do apt-get)

 # apt-get install mozilla-firefox (para instalar o programa em si) O "#" significa que o comando deve ser executado como *root* (o usuário com privilégios administrativos, como o "administrador" do *Windows*).

A página oficial do projeto Kurumin é *http://www.guiadohardware.net/gdhpress/kurumin/.*

2.3.2 Linguagem de Programação

Python é uma linguagem de programação simples e poderosa possuindo mecanismos eficientes e com um bom nível de abstração para manipulação de estrutura de dados. É uma linguagem interativa, interpretada[2] e orientada a objetos.

Características da linguagem:

[2] Programas escritos em linguagens interpretadas não são convertidos em um arquivo executável. Eles são executados utilizando um outro programa, o interpretador, que lê o código-fonte e o interpreta diretamente, durante a sua execução.

- Tipos dinâmicos de variáveis, retornos de funções e parâmetros, que são inferidos pelo interpretador. A tipagem também é forte, pois os valores e objetos têm tipos bem definidos e não sofrem coerções como em linguagem *C* ou *Perl;*
- Orientada a objetos ainda que suporte outros paradigmas, como a programação funcional e modular;
- Coletor de lixo automático, gerência de memória pelo interpretador;
- Portável, disponível para praticamente qualquer sistema operacional;
- Extensível para implementação em conjunto com outras linguagens, como *C*.

Para a implementação deste sistema foi usada a versão 2.4.4c0 da linguagem de programação *Python*. Abaixo segue a figura 6 com o comando *apt-get* de instalação:

```
root@infoway:/var# apt-get install python2.4
root@infoway:/var# python
Python 2.4.4 (#2, Jun 13 2006, 23:12:55)
(GCC 4.1.2 20060613 (prerelease) (Debian 4.1.1-4)) on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figura 6 - comando *apt-get*

2.3.3 O Servidor de Aplicações: Zope

O Zope consiste de vários componentes diferentes que trabalham juntos para ajudar o usuário a construir aplicações para a *Web,* é só instalar e sair configurando e programando, constituindo uma das melhores opções para desenvolvimento e manutenção de Aplicações Web e Gerenciamento de Conteúdos. *ZOPE* é um acrônimo para "*Z Object Publishing Environment*" (Ambiente *Z* para Publicação de Objetos). Trata-se de um Ambiente para Desenvolvimento e Gerenciamento de Aplicações *Web* ou um Servidor de Aplicações.

Foi desenvolvido pela *Zope Corporation*, antiga *Digital Creations*, é escrito em linguagem *Python*. É registrado sob uma licença "*Open Source*" compatível com *a GPL*, ou seja, é gratuito e tem seu código fonte disponível.

O *Zope* é Orientado a Objetos e possui um Banco de Objetos embutido, o *Zope Object DataBase* (*ZODB*). Possui também um Servidor *Web* embutido, o *ZServer* e duas linguagens de *templates*, a *DTML* e a *Template Attribute Language* (*TAL*).

O *ZOBD* armazena os objetos em uma representação simples, que é formada pela classe do objeto e uma estrutura de dicionário[3] contendo nas chaves as propriedades do objeto e nos valores associados às chaves que cada propriedade possui. Essa é uma representação muito eficiente e que na prática permite, através da implementação de uma interface definida no *ZOBD*. As transações são tratadas de forma que quando ocorre um erro ou alguma operação invalida, todas as alterações são desfeitas por meio de um "*rolled back*" para aquela transação. A arquitetura do *Zope* é construída em um único bloco modular, conforme a figura 7.

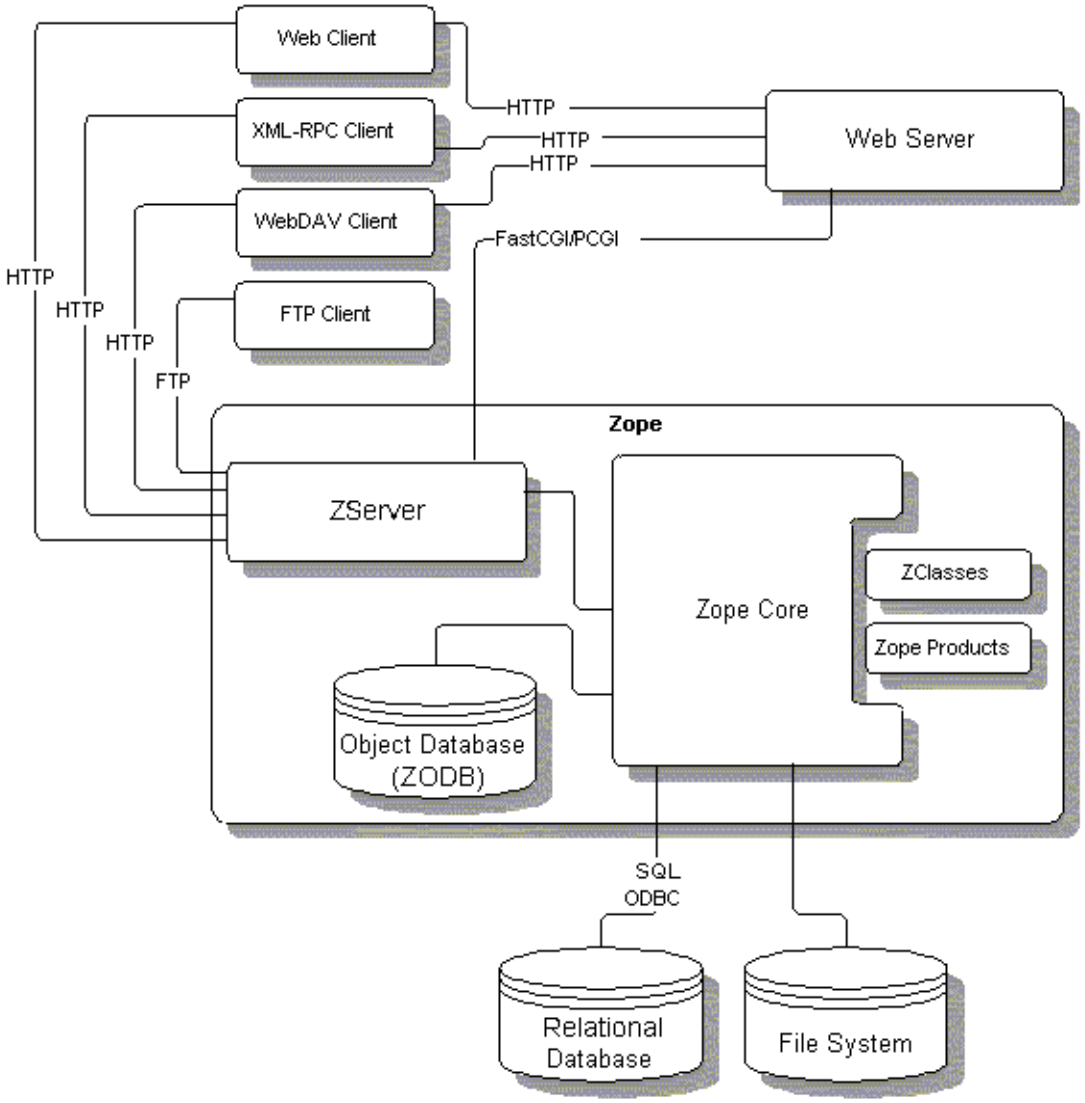

Figura 7 – Arquitetra do *Zope*.

[3] Os dicionários são estruturas da linguagem *Python* que que associam um objeto (chamado de chave) a outro objeto (chamado de valor). Um exemplo: {'Marta':8542,'Joao':1554}

O *Zope* tem como principal objetivo servir páginas *Web*. A manutenção e o gerenciamento de um *site* feito em *ZOPE* são realizadas quase que na sua totalidade pela própria *Web*, através de uma interface similar a um gerenciador de arquivos. Esta interface é chamada de *Zope Management Interface* (*ZMI)*, conforme figura 8.

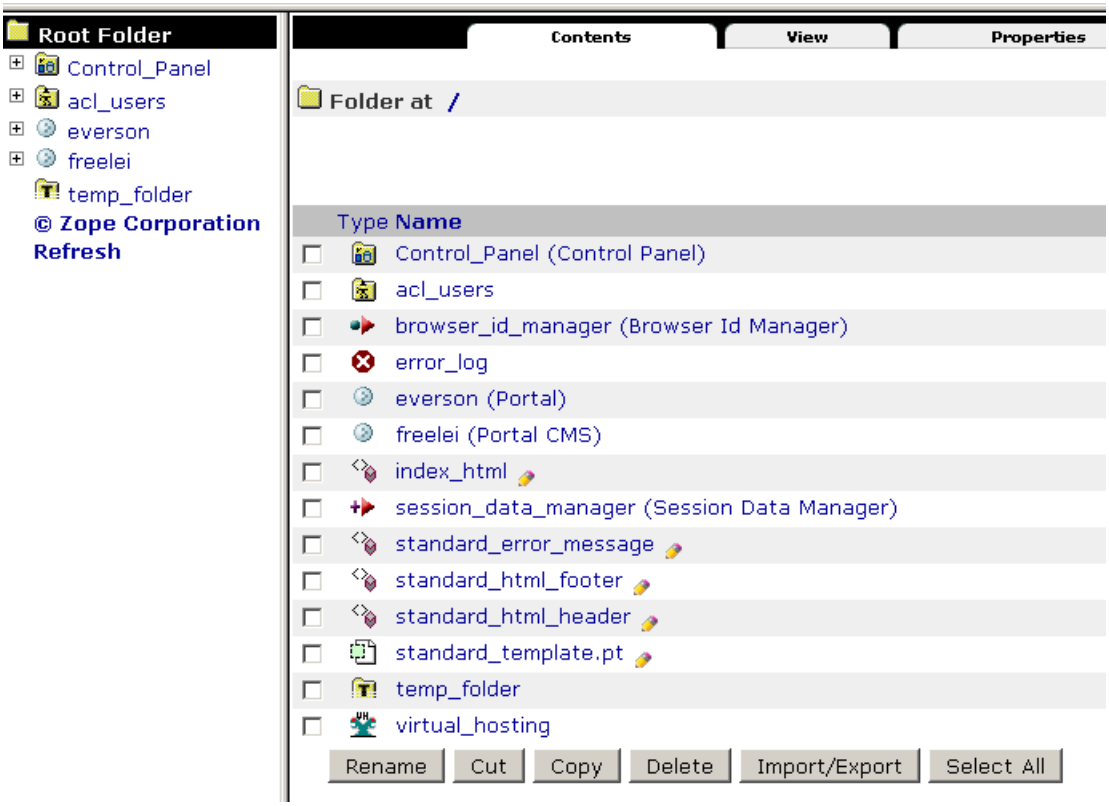

Figura 8 - *Zope Management Interface.*

O acesso ao *ZMI* do *Zope* é feito através do navegador *web* digitando-se o endereço a onde o servidor está instalado, como por exemplo *http*://<endereço>/*manage* sendo solicitado o usuário e senha do administrador *Zope*, definidos na instalação.

O *zope* pode ser instalado com o comando *apt-get*:

- *root@infoway:/# apt-get install plone-site*

Visando simplificar a instalação do *Zope*, optou-se por logo após a instalação do *Zope*, instalar-se também o *Plone*, pois através do comando *apt-get plone-site* é possível instalar todas as dependências que o *Zope* e *o Plone* vão precisar.

Na instalação do *Zope* é definido, além do usuário e senha do administrador do sistema, a porta *http* em que o servidor irá trabalhar, sendo a *default* a porta 8080, mas no caso deste sistema foi usada a porta 80, pois é um padrão *http*. Na figura 9 é apresentado o Painel de Controle do *Zope*:

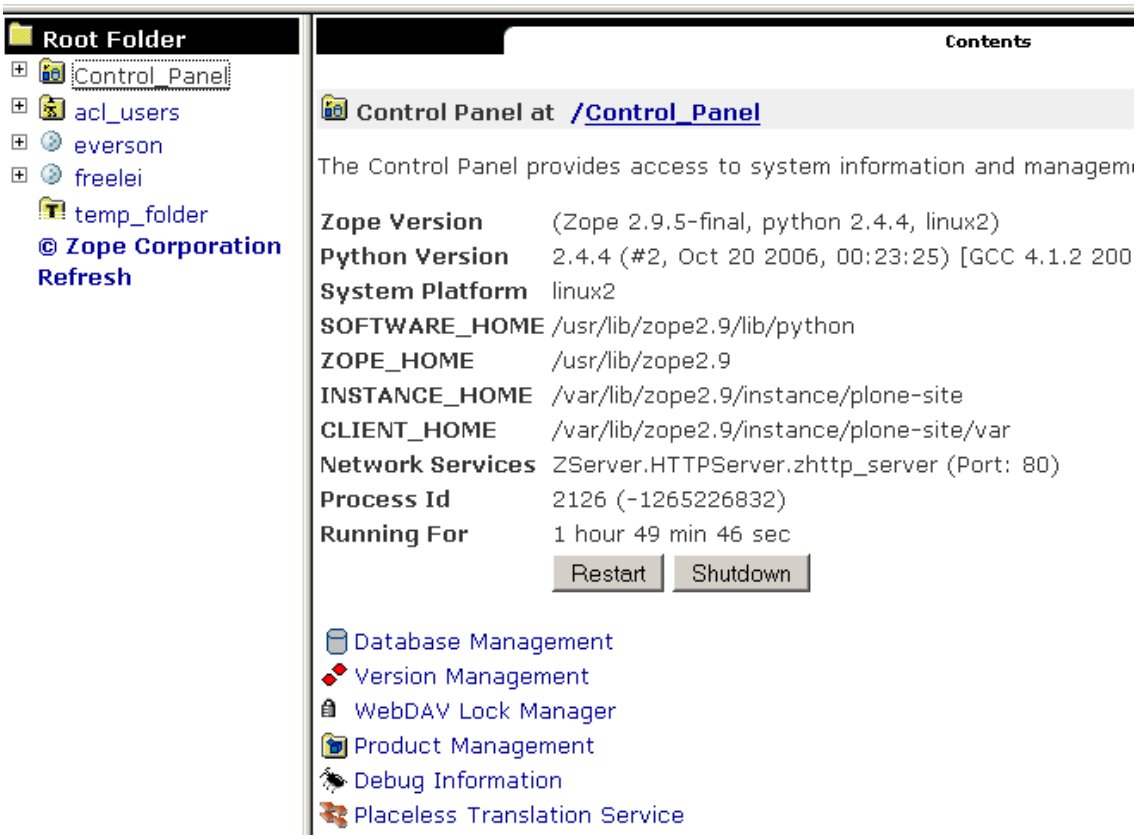

Figura 9 - Painel de Controle do *Zope*

O *Zope* instala seus arquivos de configuração no diretório */etc/zope2.9/plone-site* onde através do arquivo *zope.conf* pode-se definir suas configurações. Abaixo é apresentado um trecho do arquivo de configuração visualizado na figura 10:

```
%define INSTANCE /var/lib/zope2.9/instance/plone-site
%define ZOPE /usr/lib/zope2.9
%define ZOPE_USER zope
%define HTTPPORT 80
```

Figura 10 – Parte do código de configuração do *Zope*

Uma particularidade do *Zope* é seu Banco de objetos embutido, o *ZODB*, que apresenta a sua interface apresentada na figura 11 abaixo. Ele guarda todos os dados em um único arquivo chamada *Data.fs* no diretório */var/lib/zope2.9/plone-site/var/Data.fs.*

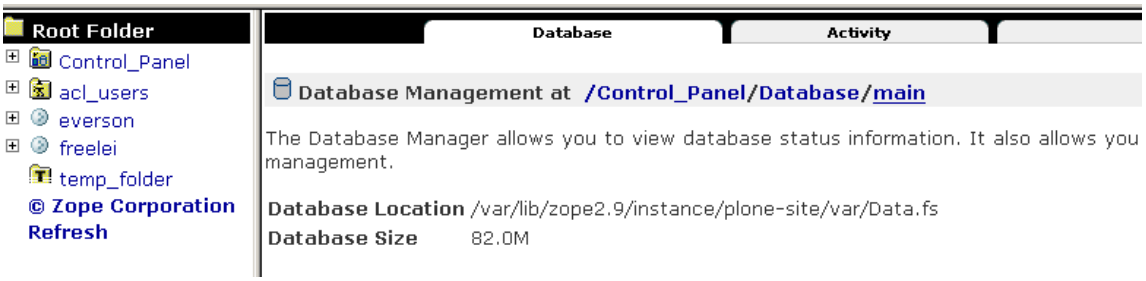

Figura 11 - Inferface do Banco de Objetos do *Zope*.

Para iniciar o *Zope*, utiliza-se o seguinte comando:

- *root@infoway:~# /etc/init.d/zope2.9 start*

Zope2.9: starting plone-site instance.

Também é possível parar e reiniciar o servidor *Zope* no Painel de Controle do mesmo.

Após a instalação e configuração do *Zope*, é necessário criar um novo *Plone Site* através do *ZMI.*

2.3.4 O Gerenciador de Conteúdo: Plone

O *Plone* é um Sistema Gerenciador de Conteúdo livre e de código aberto que roda sobre o servidor *ZOPE*. Ele advém de uma evolução do *CMF Zope*. O *Plone* vem com um sistema de *workflow*, segurança e funções pré-configuradas, um conjunto de tipos de conteúdo e suporte a vários idiomas. Existem vários desenvolvedores e usuários que testam o *Plone* de todas as partes do mundo, contribuindo diariamente com o seu desenvolvimento.

Como dito anteriormente, nesse caso específico deste sistema, o *Plone* foi instalado juntamente com o *Zope* através do comando *apt-get install plone-site* onde são instalados todas as dependências necessárias tanto para o *Plone* como para o *Zope*. Também pode-se fazer a instalação em separado, um por vez, mas não é recomendado.

Para poder criar um novo *Plone Site* é necessário dentro do *CMF* do *Zope* criar um novo tipo, conforme a figura 12.

Figura 12 - Criando um novo *Plone Site*.

Logo após esta etapa é necessário definir o campo "*ID*", onde será o nome do site a ser criado. Esse ID atribuído na criação nada, mais é que o endereço *URL*[4] que o novo *Plone Site* terá.

O *Plone* acrescenta ao *Zope* uma característica muito útil, ele traz um bom e estruturado *framework* para navegação, baseado em pastas e visualização de seu conteúdo ao invés de *links* em documentos *html* (que tem que ser atualizados) e adicionados e avisos (que mostram objetos criados ou modificados nos últimos dias, eliminando assim a necessidade de se navegar pelo site para achá-los). Esta característica faz do site baseado em *Plone* ser fácil e rápido de usar.

O *Plone* é uma tecnologia altamente personalizável, e muito expansível. Para criar melhorias, são desenvolvidos produtos para trabalhar com o *Plone*. Muitos produtos do *Plone* são tão utilizados que muitas pessoas nem sabe que aquela aplicação não é padrão do *Plone* e foi instalada a parte.

Boa parte das novas funcionalidades do *Plone*, são oferecidas através de Produtos. Estes produtos devem ser instalados no *Plone*, como, por exemplo, o *Plone* sendo instalado no *Zope*. Para instalar um produto, deve-se mover a pasta com o Produto para dentro da sua instância do *Zope*, que contém o *Plone*. Reiniciar o servidor *Zope* e acessar a opção Adicionar/Remover Produtos do *Plone* .

[4] Uma *URL* (de *Universal Resource Locator*) em português significa (Localizador Universal de Recursos) é o endereço de um recurso (um arquivo, uma impressora etc.), disponível em uma rede; seja a *Internet*, ou uma rede corporativa, uma *intranet*. Uma *URL* tem a seguinte estrutura: protocolo://máquina/caminho/recurso

Na figura 13 pode-se visualizar como ficou o *site Freelei* dentro *CMI* do *Zope*.

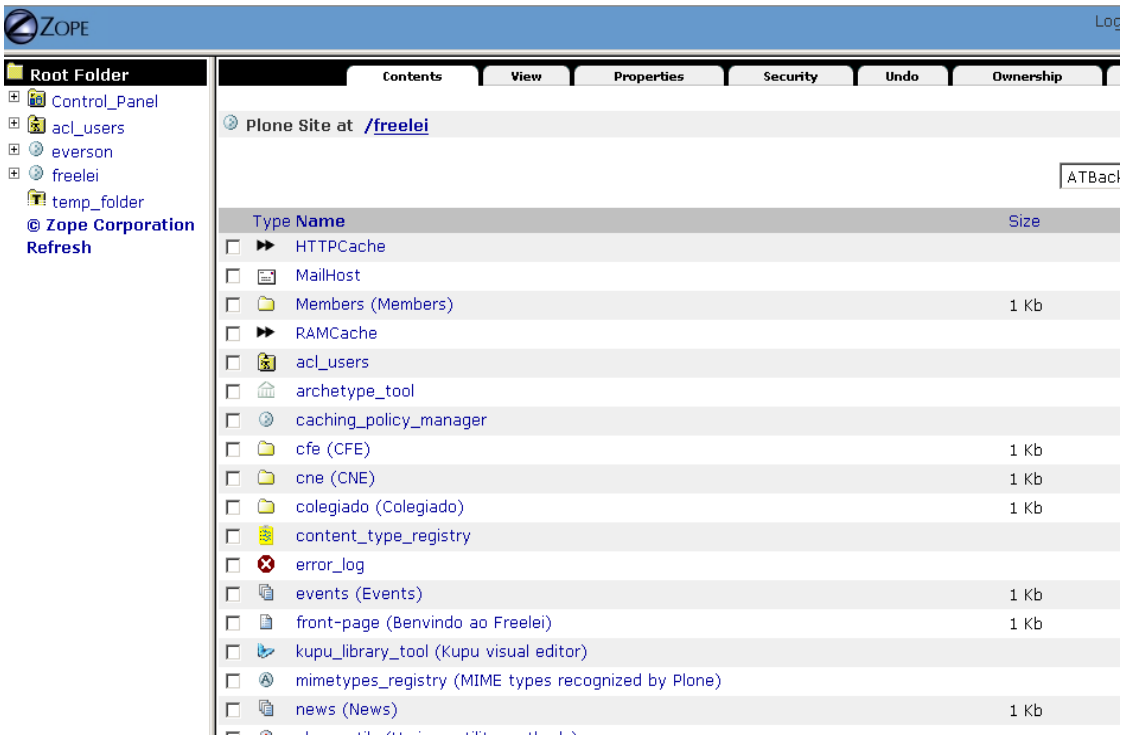

Figura 13 - Site Freelei dentro do *CMI* do *Zope*

2.3.5 A Internacionalização do *Plone*

Há um esforço constante em manter uma alta qualidade nas traduções da interface para os usuários do *Plone*. O fato é que o *Plone* provêm o acesso a sua interface em mais de 30 linguagens. É possível também criar novas traduções e inserí-la em um site *Plone*, possibilitando assim a criação de *websites* multilíngües.

Tal recurso é possível com a instalação de um módulo *Python* denominado *i18ndude*. Para isso deve-se:

- Baixar o pacote *i18ndude-2.1-py2.4.egg* de

  http://plone.org/products/i18ndude

- Salvar no diretório

  /var/lib/zope2.9/instance/plone-site/Products/ArchGenXML/

- Extrair os dados, com isso é criado os diretórios

  i18ndude e EGG-INFO

• Remover o pacote *i18ndude-2.1-py2.4.egg*:

rm -f i18ndude-2.1-py2.4.egg

A partir deste momento o *Site Plone* poderá ter suporte a Intercionalização de idiomas, podendo os produtos que forem criados para o mesmo terem a mesma característica. A figura 14 pode-se visualizar como isto pode ser feito, mostrando em quais idiomas o site poderá ser visualizado.

Figura 14 - Definindo quais idiomas o s*ite* poderá ser visualizado.

O Administrador do *site Plone* pode definir os idiomas em que o usuário do site poderá visualizar o conteúdo. Do ponto de vista do usuário o modo de alteração do idioma fica conforme a figura 15.

Figura 15 - O idioma do s*ite* foi definido para "*Italiano*".

2.3.6 O Conversor *HTML*

O *stripogram* é uma Biblioteca para converter *HTML* para texto puro e separar *tags* específicas do *HTML*.

A instalação pode ser feita como módulo *Python* ou como produto do *Plone*. Optou-se por instalar como produto *Plone* pelo fato de uma atualização deste produto ser mais simples do que uma instalação como módulo do *Python*. Abaixo pode-se ver os passos da instalação:

- Baixar o pacote *stripogram-1-4.tgz* de

  http://sourceforge.net/project/showfiles.php?group_id=1083&package_id=34645
- Salvar o pacote *stripogram-1-4.tgz* no diretório

  /var/lib/zope2.9/instance/plone-site/Products
- Descompactar o pacote

  tar -zxvf stripogram-1-4.tar.gz
- Será criado o diretório *stripogram*
- Remover o pacote stripogram-1-4.tgz

  rm -f stripogram-1-4.tgz

2.3.7 A Ferramenta de Modelagem *UML*

O *Poseidon* é uma Ferramenta de modelagem *OO* desenvolvida em *JAVA*, sendo *Cross-Plataform*[5] desde que tenha a versão do *Java* compatível instalada. Possui uma versão que é gratuita, que é a *Community Edition*, a qual foi utilizada neste sistema.

O *Poseidon* requer versão 1.5 para *Java JRE* ou *JDK* para executar. Caso não haja nenhuma *JRE* ou *JDK* instalado, deve-se baixar a versão com *Poseidon for UML 4.2.1 – com JRE.*

A *Community Edition* é uma versão gratuita e tem as seguintes funcionalidades:

- Executa em qualquer plataforma por ser totalmente implementada em *Java;*
- Engenharia reversa e geração automática de código *Java*;
- Todos os 9 diagramas de *UML* estão disponíveis;
- Compatível com o padrão *UML* 1.4;
- O formato *XMI* como o padrão dos arquivos de diagramas ;

---

[5] *Cross-platform* é um termo referente a programas, sistemas operacionais, linguagens de computador, linguagens de programação ou outros *softwares* de computador e implementações que as implementações deles/delas podem ser feitas para trabalhar em plataformas de computador múltiplas.

- Exportação de diagrama como *GIF, PS, EPS, SVG, JPEG e PNG*;

- *copy/cut/paste, drag-and-drop e undo/redo* dentro da ferramenta;

- *Zoom in e out* de diagramas;

- *Auto-layout* de diagramas gerados por engenharia reversa;

- Crítica cognitiva de diagramas, fornecendo sugestões de correção;

- Pode ser usado via *web* (*Java Web Start*).

Através da figura 16 pode-se visualizar a ferramenta de modelagem *Poseidon.* Convém salientar que o *Poseidon* salva seus arquivos no formato *".zuml"*.

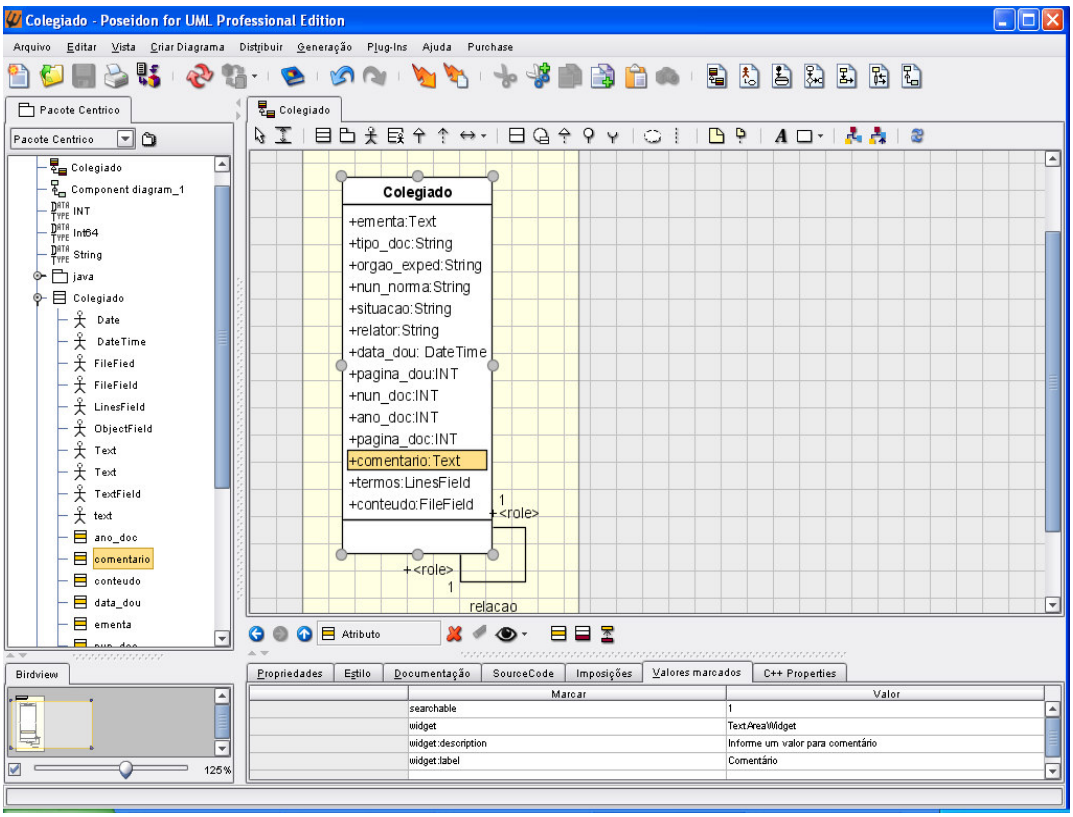

Figura 16 – A ferramenta de modelagem *Poseidon.*

Através do endereço *http://www.gentleware.com/* pode-se obter o *Poseidon* e informações relacionadas.

2.3.8 Archetypes

Archetypes é um *framework* projetado para facilitar a construção de aplicações para o *Plone*. Seu principal propósito é prover um método comum para construção de tipos de conteúdo baseados em definições de *schema*. Campos podem ser agrupados para edição, tornando muito simples a criação de formulários.

Archetypes facilita o desenvolvimento ao fazer o trabalho comum, maçante e pesado, permitindo ao desenvolvedor pensar em coisas mais importantes, tais como regras de negócio e planejamento. Fornece as seguintes características:

- Auto-geração de edição e apresentação;
- Fácil instalação dos tipos de objetos gerados;
- *References,* entre outros...

O princípio do *archetypes* está na definição de *schemas*. Um *schema* é o conjunto de campos (atributos/*fields*), é a definição de um tipo. Quando se define um *schema*, define-se os campos que um objeto vai ter, como se estivesse criando uma tabela, conforme figura 17 abaixo:

```
schema = Schema
((
   TextField(name='ementa') ,
   StringField(name='tipo_doc'),
  StringField(name='orgao_exped'),
))
```

Figura 17 - Definição de *schemas*

Em cada campo pode-se definir seu tipo, permissão de escrita/leitura, se é requerido, se deve ser indexado, *validadores*, etc. O mais interessante é que, além disto, pode-se definir como cada campo será visualizado/editado. Cada campo possui um *widget* que é responsável por renderizar o campo na tela. Na figura 18 é apresentado um pequeno trecho de código onde é apresentado um *widget*.

```
        IntegerField(
                name='ano_doc',
                widget=IntegerWidget
                (
                   description="Informe um valor para o Ano do Documento",
                   label="Ano do Documento",
                   label_msgid='Legislacao_label_ano_doc',
                   description_msgid='Legislacao_help_ano_doc',
                   i18n_domain='Legislacao',
                ),
                validators=('IsInt',)
    ),
```

Figura 18 – Representação de um *widget*

Na figura 19 é apresentado como o *Archetypes* estrutura *Schema, field e widget*.

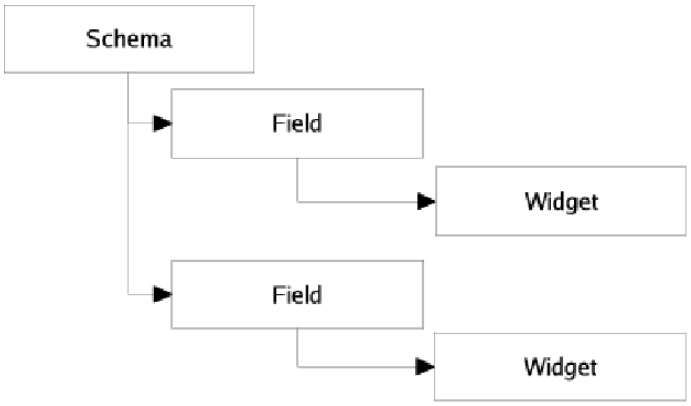

Figura 19 – *Archetypes Schema, Field e Widget*

Apartir do *Plone* 2.5 o *Archetypes* está disponível para instalação, necessitando ir até a seção de instalação de novos produtos para adicioná-lo ao *Plone Site*.

2.3.9 ArchGenXML

O *ArchGenXML* é um gerador de código para aplicações (Produtos) do *CMF/Plone* baseados no *framework Archetypes*. Ele executa um *parser*[6] sobre um modelo *UML* no formato *XMI (.xmi, .zargo, .zuml*), criado com aplicações como o *Poseidon*.

Ele não vem junto com o *Plone*, necessita ser obtido da Central de Produtos em http://plone.org/products/ e descompactado na pasta "*Products*" do *Plone Site*.

O *ArchGenXML* é um gerador de códigos baseado em linha de comando, que permite criá-los sem escrever diretamente na linguagem. O seu uso tem sido articulado com o *Archetypes*, que é usado para produzir novos tipos de conteúdo para o *Plone*.

Na figura 20 é apresentado como o *ArchGenXML* através de um diagrama *UML* gera código *Python*.

Figura 20 - *ArchGenXML* gerando código *Python*.

Como dito anteriormente, o *ArchGenXML* é um gerador de códigos baseados em linha de comando, através de um arquivo gerado com o *Poseidon*, salvo com a extenção (*.zuml*), é possível gerar produtos para o *Plone* com o auxílio do *framework Archetypes*. No código da figura 21, o *ArcheGenXML.py* através do parâmetro "-o Legislacao" e com base no arquivo Legislacao.zuml gera o Produto Legislacao, sendo que mesmo aparecerá como uma pasta que poderá ser movida para a pasta Products da instância do Plone.

[6] Procesa os arquivos do código fonte a procura de informações relevantes para a construção do modelo e as coloca em memória, ainda sem a organização necessária; funciona como uma espécie de analisador sintático, utilizando um termo mais conhecido da área de compiladores.

```
root@infoway:/var/lib/zope2.9/instance/plone-site/Products# ArchGenXML.py
-o Legislacao Legislacao.zuml
ArchGenXML Version 1.5.0
(c) 2003-2006 BlueDynamics, Austria, GNU General Public License 2.0 or
later
INFO  Parsing...
INFO  Directory in which we're generating the files: 'Legislacao'.
INFO  Generating...
INFO  Starting new Product: 'Legislacao'.
INFO     Generating class 'Legislacao'.
```

Figrura 21 – ArchGenXML gerando o Produto Legislacao

A pasta Legislacao é agora um novo produto *Plone*, podendo ser instalado em qualquer *Plone Site*. Analisado o conteúdo do Produto Legislação, tem-se como principais arquivos:

- *__init__.py*

Este arquivo será executado quando o *Zope* carregar o produto. Ele possui código para inicializar o produto.

- *config.py*

Arquivo que possui as constantes de configuração do produto: o nome do produto, as permissões dos tipos de conteúdos criados, etc.

- */Extensions/Install.py*

É nesta pasta onde está o método externo de instalação do produto. O arquivo *Install.py* é lido pelo *QuickInstaller* no *Plone* e seus métodos *install() e uninstall()* são executados.

- */Skins/Legislacao*

Pasta que será registrada com o *portal_skins*, fazendo com que *page templates* e *scripts* sejam usados em todo o portal.

O funcionamento do *ArchGenXML* é representado basicamente pela figura 22.

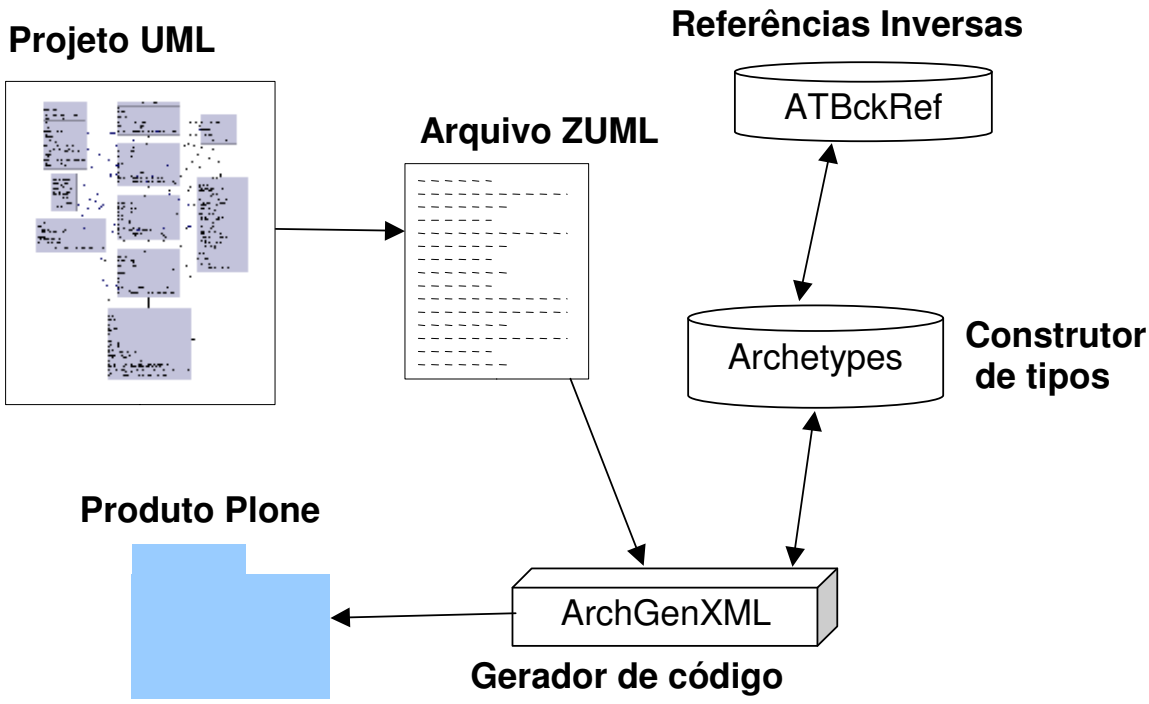

Figura 22 – Funcionamento do *ArchGenXMl*

2.3.10 ATBackRef

O Produto *ATBackRef* ou *back references* como é comumente chamado cria um campo/*widget* do tipo *BackReference* novo para o *Archetypes*. Ele cria uma "referência para trás" em um relacionamento. Por exemplo, se um objeto do tipo Artista tiver uma referência a um grupo o qual pertence, quando forem visualizados seus dados aparecerá um campo do tipo *link* para o grupo o qual pertence.

Para criar campos do tipo *BackReference* é necessário criar o campo no diagrama *UML*, conforme a figura 23. Para poder criar Produtos com essa propriedade é necessário a seguinte linha de comando quando da geração do Produto com o ArchGenXML:

- *# ArchGenXML.py --backreferences-support=YES Colegiado.zuml*

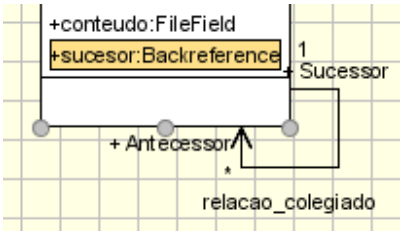

Figura 23 – Criando um novo tipo *BackReference* no diagrama *UML*

2.4 Mudando o *Layout* de Apresentação dos Produtos

As imagens, cores e scripts que se pode mudar no *Plone* estão na pasta "*portal_skins*". Quando esta pasta é acessada, obtém um conjunto de pastas (*custom, plone_3rdParty, plone_content*) que são responsáveis pela estrutura do *Plone*. Então quando se quer mudar algo, é aqui que deve-se trabalhar.

Quando um produto é instalado ele registra sua pasta "*skins*" no *portal_skins*, fazendo com que *page templates* e *scripts* customizados sejam usados em todo o portal.

O *Plone* procura primeiramente a pasta "*skins*" do Produto, se não encontrar nada ele procura ma pasta "*custom*" em *portal_skins*, caso não ache o que procura novamente e segue para a pasta "*gruf*" e assim por diante.

Para mudar o modo de apresentação dos conteúdos Legislação e Colegiado, os arquivos Legislação_*view*.pt e Colegiado_*view*.pt foram customizados através da linguagem *ZPT,* ela se incorpora à linguagem *HTML*, isto é, ela reaproveita todos os comandos *HTML* e adiciona um novo atributo aos comandos. Este atributo é o "*tal*" que pode ser inserido em todos os comandos *HTML* (*b, i, table, body, href, etc*...).

Abaixo a figura 24 apresenta um trecho do código *ZPT* do arquivo colegiado_*view*.pt.

```
<div class="field" tal:define="widget_view python:here.widget('termos')"
tal:condition="here/termos">
        <label tal:content="python:widget.Label(here) + ':'"/><br>
        <metal:block tal:repeat="item accessor">
 <a href="#" tal:attributes="href string:${here/portal_url
}/search?SearchableText=${item}" tal:content="item"/><br>
        </metal:block>
 </div>
```

Figura 24 - Trecho do código *ZPT* do arquivo colegiado_*view*.pt.

Analisado o código tem-se:

- *tal:define*

O comando *tal:define* é usado para definir variáveis que são referenciadas por um caminho.

- *tal:condition*

Usado para incluir uma condição no contexto do caminho usado, por exemplo: caso o campo estiver em branco ele não será exibido.

- *tal:content*

Muito usado para substituir o valor que está no comando *HTML* por outro conteúdo mais significativo. Por exemplo: o valor do *label* será substituído pelo *label* do *widget* associado a este campo.

- *tal:repeat*

É formado por uma variável "item" que será usada nas linhas para identificar as propriedades do campo. O termo “acessor” são os dados do campo em questão. A cada interação do loop (a cada palavra encontrada), item será um objeto do tipo “acessor”.

- *tal:attributes*

O comando *tal:attributes* adiciona atributos aos comandos *HTML*. Por exemplo, quando é feito um link no *HTML*, a sintaxe básica é: <a *href*="local de destino"> local . Com o *tal:attributes* consegue-se alterar o valor do atributo "*href*".

3 APRESENTAÇAO DO SISTEMA

Para a visualização do *Freelei,* é necessário ter acesso a um *browser* que possa trabalhar bem com *Cascading Style Sheets* (*CSS*), para que a sua identidade visual não seja comprometida.

A tela inicial do *Freelei* pode ser visualizada na figura 25 A visão apresentada nesta figura é a visão pública do sistema.

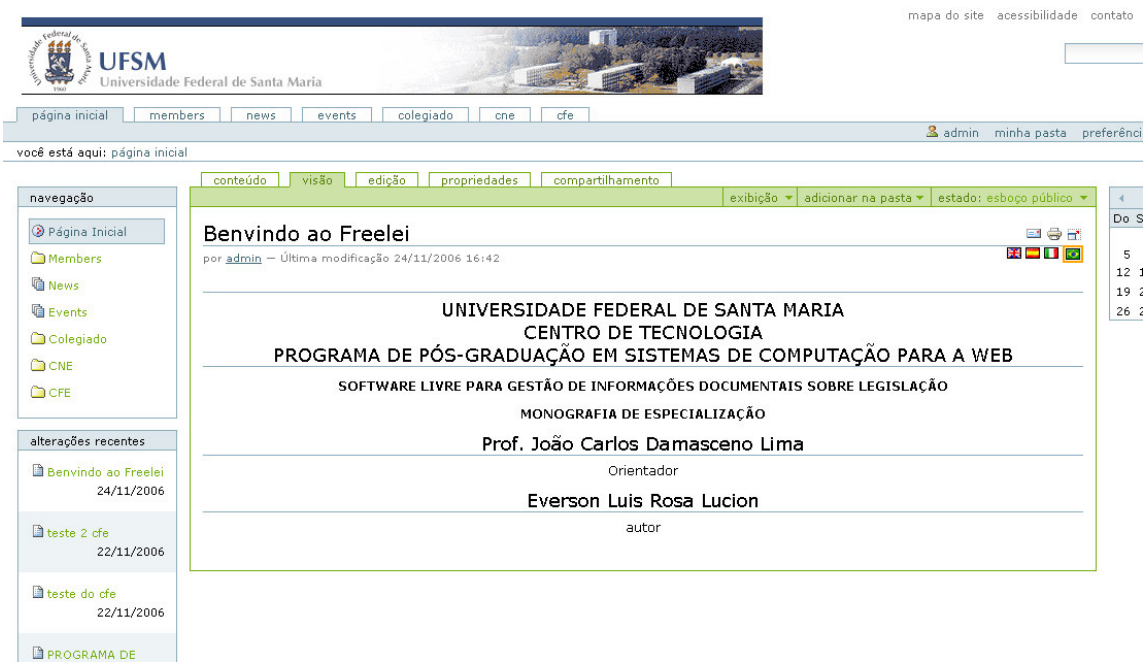

Figura 25 – Tela Inicial do *Freelei*

O *Plone* possui um sistema de *Workflow* para que o processo de publicação de conteúdo seja coerente, qualquer documento do *website* deve ser passado por aprovação de alguém que tem mais direitos do que um simples usuário. *O workflow* padrão do *Plone* é esse, uma pessoa escreve e a outra publica". Para o sistema Freelei, foi criado o grupo de usuários "Usuários do Freelei", onde os mesmos podem adicionar novos documentos do tipo Colegiado ou Legislação. Os usuários do grupo "Revisores" podem adicionar novos documentos e também publicar os mesmos, a figura 26 mostra os papéis de cada grupo de usuários e a figura 27 mostra aonde aplicar regras avançadas para controle de usuários e grupos.

Administração via site com privilégios de administrador

Figura 26 – Papéis dos grupos de usuários do Sistema Freelei.

Administração através do *CMI* do *Zope*

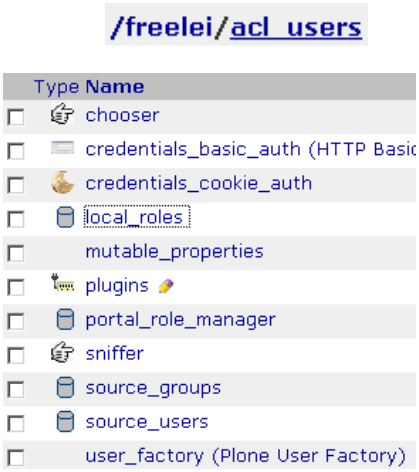

Figura 27 – Controle avançado de grupos e usuários

Após a autenticação é possível usar as funcionalidades do sistema, onde se pode "adicionar um novo item" conforme a figura 28.

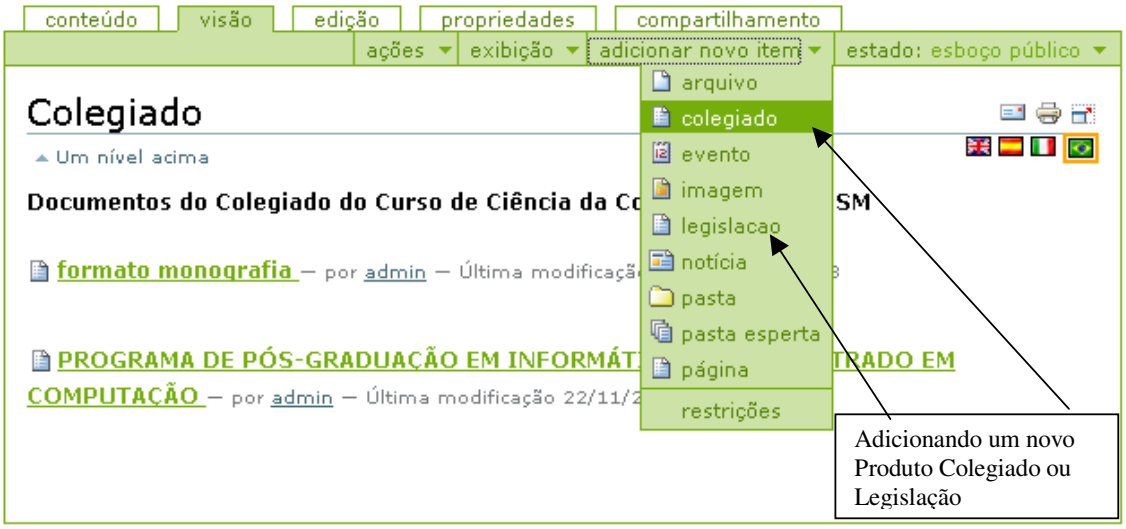

Figura 28 – Adicionando um novo ítem

O Produto Colegiado e Legislação possuem um formulário padrão para adição de novo conteúdo, a figura 29 mostra a parte final do produto Colegiado.

Figura 29 – Alguns campos do novo item Colegiado.

O *Plone* disponibiliza um mecanismo de busca muito eficiente no conteúdo do portal como um todo, a cada caractere digitado ele mostra a relevância de precisão com o termo que está sendo digitado, a figura 30 mostra um exemplo disto.

Figura 30 – Efetuado uma busca de conteúdo no site.

Um dos aspectos importantes no preenchimento do formulário é a relação com seu antecessor, caso haja. A figura 31 mostra como é referenciado seu antecessor.

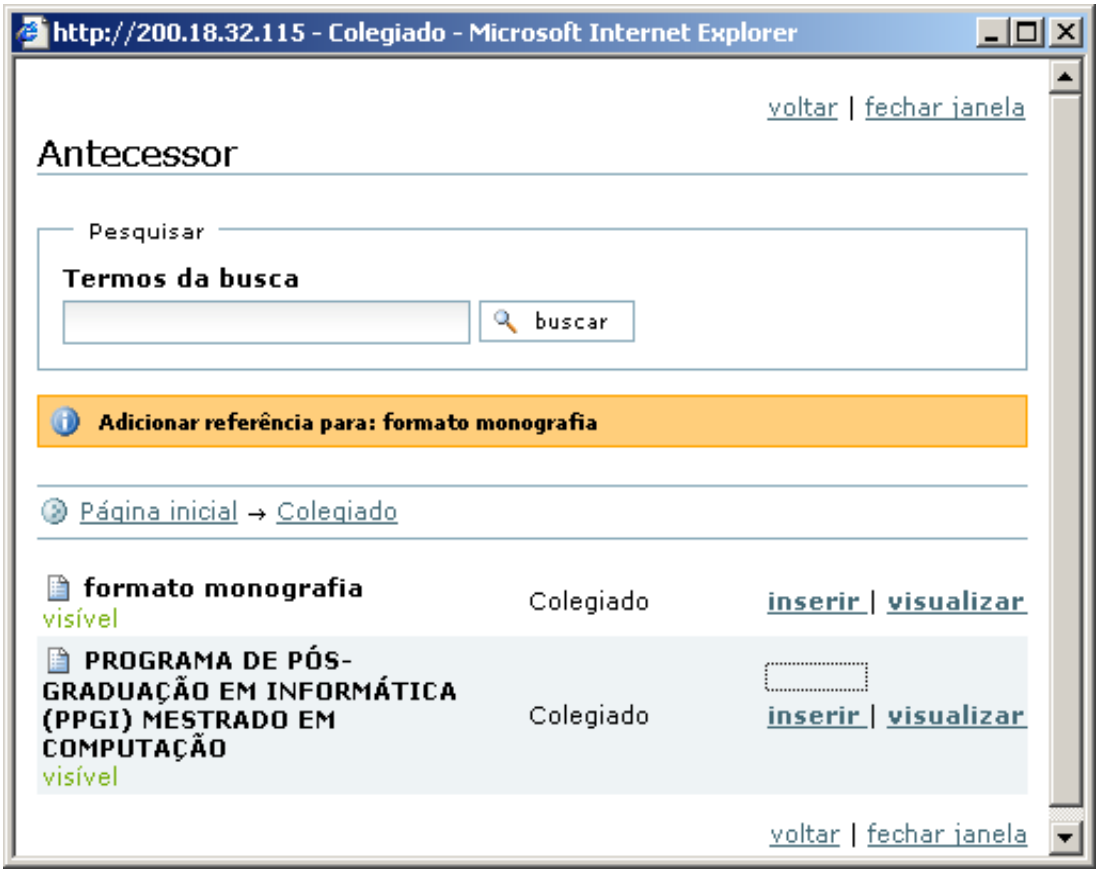

Figura 31 – Selecionando e referenciado o documento antecessor.

Como resultado final deste trabalho, chega-se à tela representada pela figura 32 onde o usuário terá como resultado da busca anterior. Os dados são meramente fictícios e ilustrativos.

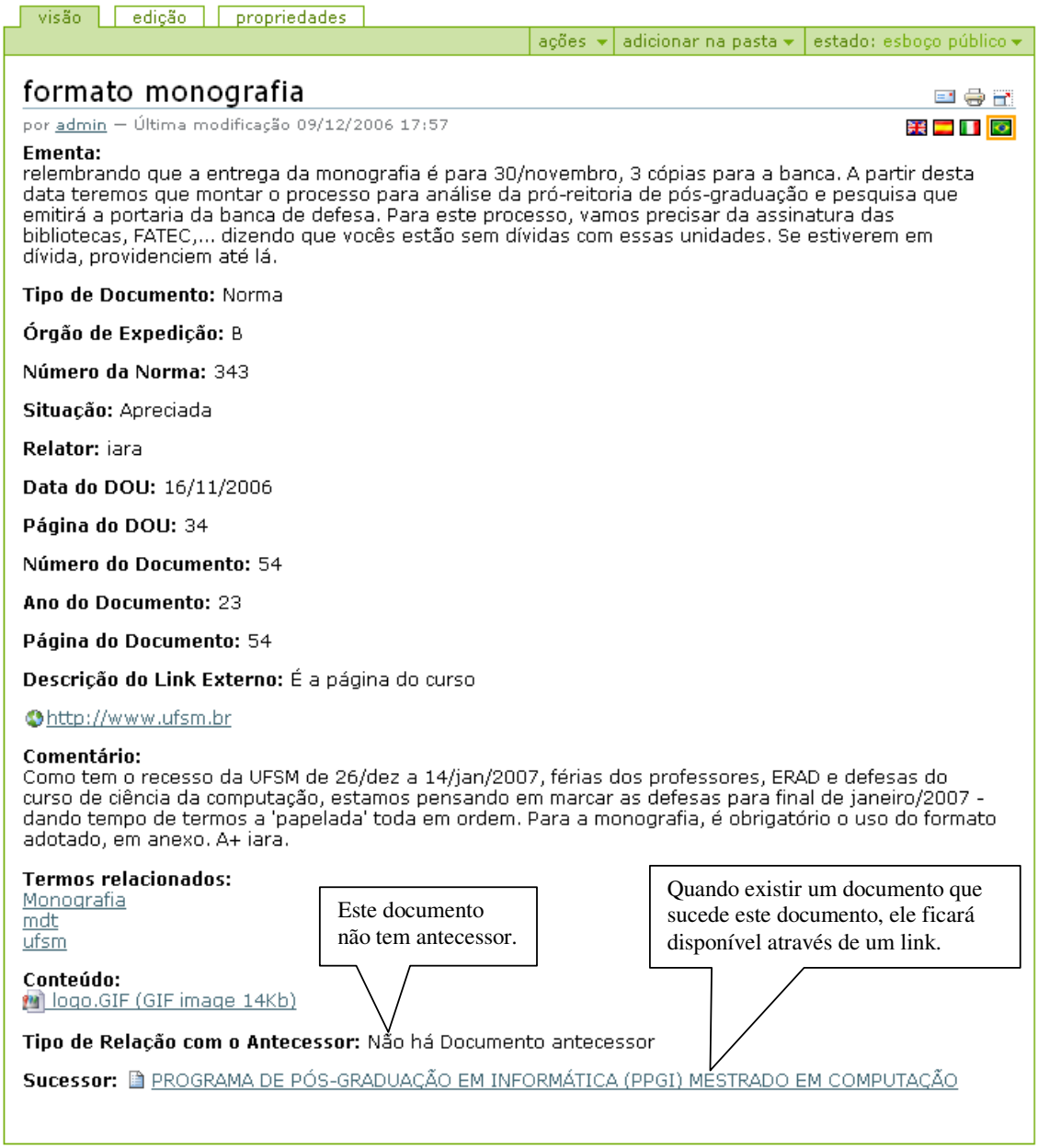

Figura 32 – O resultado de uma pesquisa por documentação específica.

Clicando-se no link “Sucessor” da figura 32 pode-se visualizar o documento que o sucede, o qual é visualizado na figura 33.

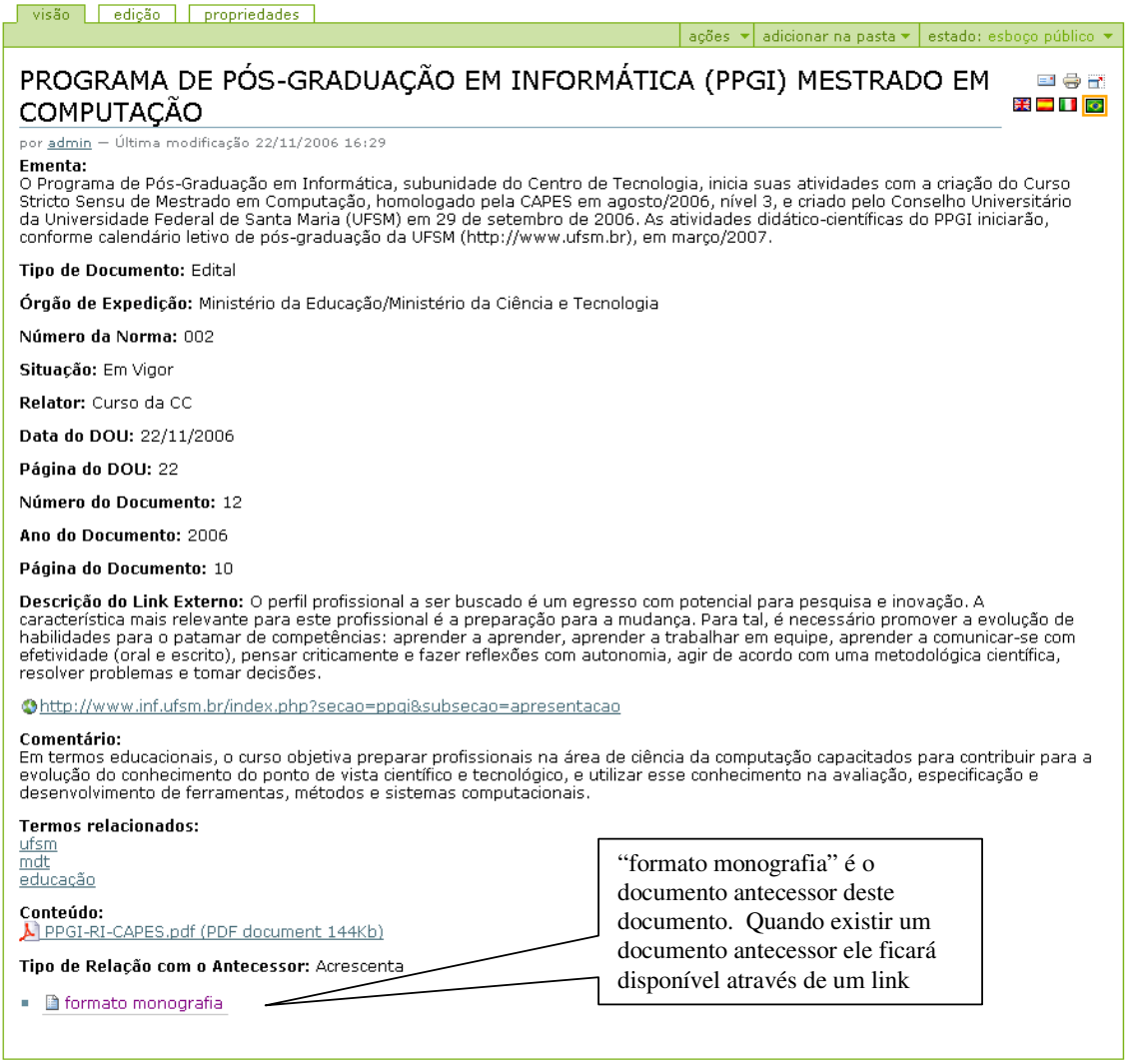

Figura 33 – Documento sucessor.

Convêm salientar que:

1 - Um documento pode ter zero, um ou vários sucessores;

2 - Um documento pode possuir zero, um ou vários antecessores.

CONCLUSÕES

A proposta deste trabalho é um estudo sobre Sistemas de Gerenciamento de Conteúdo na *Web*, com o uso de softwares livres, pesquisando e desenvolvendo uma ferramenta de software para auxiliar na gestão de documentos sobre legislação educacional. Teve como foco principal o conjunto de informações inter-relacionadas que normatizam o sistema educacional brasileiro em suas diversas esferas.

Usou-se como base documental e estrutural as informações documentais de legislação e de pareceres do CNE e CFE. Num segundo aspecto, procurou abranger outro setor que também padece com problemas semelhantes, os conselhos das universidades, que decidem o dia-a-dia das novas diretrizes da educação das mais importantes instituições educacionais do país.

O propósito é buscar formas de Gerenciamento de Conteúdo na Internet, buscar ferramentas que promovem meios de gerenciamento, publicação e manutenção de informações, denominadas ferramentas *CMS*.

Este trabalho propôs um sistemas de gestão de conteúdo, em especial ao *CMF Plone*, o qual acrescenta muitas características ao *Zope*.

A utilização do *CMS Plone* fez com que o desenvolvimento do portal tivesse o tempo de implementação reduzido, no que diz a respeito a *layout* e estruturação do site, pois muita coisa já vem desenvolvido. Outro benefício encontrado, foi a disponibilidade de diversos produtos para o *Plone* os quais são desenvolvidos por uma comunidade mundial de especialistas sendo uma característica típica do desenvolvimento de softwares livres.

Um aspecto que dificultou foi a documentação do *Plone*, pois 95% da mesma faz referências ao Plone 2.1, quase inexistindo documentação do Plone 2.5. Outro fator limitante é o fato de que no Brasil existam poucos provedores de serviço *ZOPE*, o qual é requisito para uso do *Plone*.

Os Produtos são um dos pontos fortes do *Plone* e com a criação dos Produtos Colegiado e Legislação, a comunidade acadêmica, através do Produto Colegiado e os muitos Conselhos, em suas diversas esferas, Nacional, Estadual e Municipal, através do Produto Legislação, terão com a criação dos mesmos uma forma de gerenciar a sua documentação, agilizar a pesquisa, estruturar o conteúdo através de pastas, referenciar os documentos, apontando o seu sucessor e o seu antecessor, incluir referências externas através de *URL* com documentos de outros órgãos ou universidades, sugerir e apontar ao leitor termos relacionados, ver o documento no seu estado original (*.doc*, *.pdf*, ...), ver dados relacionados a

publicação no Diário Oficial, podendo inserir um *link* direto com a publicação no mesmo, informar ao leitor da origem do documento.

Ressalta-se a "customização" como outro aspecto importante e ao mesmo tempo um termo muito usado dentro do ambiente *Zope* e *Plone*. Os Produtos poderão estar disponíveis de duas maneiras, a primeira, será através dos arquivos Legislação.zuml e Colegiado.zuml, trazendo ao responsável que venha implantar os Produtos, a possibilidade de modelar os mesmos conforme as peculiaridades de cada Órgão acrescentado e retirando campos ou funcionalidades que julgar necessário. A segunda, os Produtos vão estar dentro de uma pasta chamada Legislação ou Colegiado, bastando fazer a customização já no produto instalado ou pré-instalado, trazendo um personificação aos Produtos.

O gerenciamento de conteúdo na Internet é um tema que envolve muita discussão sobre a melhor maneira de realizá-lo, como é a melhor maneira de coletar, armazenar e disponibilizar esses dados. Se existe alguma certeza em relação ao gerenciamento de conteúdo, é a necessidade das organizações de gerenciarem as informações geradas de maneira simples e eficaz, permitindo a disseminação da informação, obedecendo a critérios específicos no acesso e formato de exibição do conteúdo. O *Zope* e o *Plone* tentam mostrar as pessoas que gerenciar a informação pode ser um processo simples e que trará um alto grau de satisfação as que dependem dessas informações.

Como trabalhos futuros pretende-se personificar mais ainda cada produto, desenvolvendo produtos específicos para a comunidade acadêmica e para os conselhos nas suas diversas esferas, Nacional, Estadual e Municipal.

REFERÊNCIAS BIBLIOGRÁFICAS

BALASUBRAMANIAN, V. et al. Document management and Web technologies: **Alice marries the Mad Hatter**. Communications of the ACM. V41, Issue 7, 1998.

BALDAM, Roquemar, VALLE, Rogério, CAVALCANTI, Marcos. **GED: Gerenciamento Eletrônico de Documentos**. São Paulo: Érica, 2002.

CAMPOS, Augusto. **O que é software livre**. BR-Linux. Florianópolis, março de 2006. Disponível em <http://br-linux.org/linux/faq-softwarelivre>. Acesso em: 2 set. 2006.

COSTA, H. S. D. **Desenvolvimento de um Sistema de Gerenciamento de Conteúdo na Web. 2004**. 70f. Monografia (Graduação em Ciência da Computação) – Universidade Federal do Maranhão, São Luís, 2004.

I-WEB, **Artigos e Informativos Técnicos**. Disponível em: < http://www.iweb.com.br/iweb/pdfs/20031008-appservers-01.pdf >. Acesso em: 14 jan. 2007.

INEP, **ProLEI – Programa de Legislação Educacional Integrado**. Disponível em: <http://www.inep.gov.br/> . Acesso em 29 jun. 2006

LOPES, L. C. A. **Nova Arquivística na Modernização Administrativa**. Rio de Janeiro: Arquivo Público do RJ, 2000.

OpenIP – **A Definição de Código Aberto**. Disponível em < http://www.openit.com.br/?module=displaystory&story_id=641&format=html>. Acesso em: 14 nov. 2006.

RIZZETTI ET ALL. **SIRC - Simpósio de Informática da Região Centro do Estado**. 2005. Artigo (Análise de Sistemas de Gerenciamento de Conteúdo para o Projeto PDSCE) – Universidade Federal de Santa Maria, Santa Maria, 2005.

WIKIPÉDIA – **A enciclopédia livre**. Disponível em: < http://pt.wikipedia.org/wiki/Software_livre >. Acesso em: 29 jun. 2006a.

ZANIRATO, **Consultoria Web**. São Paulo, 2006. Disponível em: <http://consultoria.zanirato.com/?p=48>. Acesso em: 14 set. 2006.