UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Arquitetura de Software para Ambiente de Colaboração Educacional, Baseado em CMS.

DANIEL CHAOUI SANTOS

FLORIANÓPOLIS
Setembro de 2008

DANIEL CHAOUI SANTOS

Arquitetura de Software para Ambiente de Colaboração Educacional, Baseado em CMS.

Monografia aprovada em ___ / ___ / 2009,
como requisito para a obtenção do grau de bacharel em Ciências da Computação.

Banca Examinadora

_____
Lúcia Helena Martins Pacheco, Dra.
Orientadora

_____
Juliano Soares dos Santos, Me.
Membro

_____
Leonardo Pereira Demilis.
Membro

DANIEL CHAOUI SANTOS

Arquitetura de Software para Ambiente de Colaboração Educacional, Baseado em CMS.

Trabalho de conclusão de curso apresentado à Universidade Federal de Santa Catarina, como parte dos requisitos para a obtenção do grau de bacharel em Ciências da Computação.

Professora Lúcia Helena Martins Pacheco

FLORIANÓPOLIS
Setembro de 2008

AGRADECIMENTOS

Primeiramente a Deus, que sempre me deu forças para vencer meus obstáculos, a minha mãe (a melhor do mundo), meu pai e meus irmãos, por todo o apoio e carinho. A minha professora e orientadora, Lúcia Helena Martins Pacheco pela paciência e correções, ao meu chefe Juliano Soares dos Santos pela ajuda e por me permitir apresentar esse projeto como trabalho de conclusão de curso e a Leonardo Pereira Demilis que aceitou prontamente o convite para participação da banca. Agradeço a todos meus amigos, colegas, e professores que de forma direta ou indireta me fizeram chegar ate aqui. Por fim, um agradecimento especial a minha namorada Katia Milrad que há quatro anos me faz uma pessoa melhor, mais feliz e que foi fundamental para minha formatura no curso de Ciências da Computação UFSC.

RESUMO

Este trabalho aborda aspectos práticos na elaboração de uma arquitetura de *software* para *Web* tendo por objetivo a criação de um ambiente de colaboração para instituições de ensino. Estabelecida uma fundamentação teórica, passou-se ao levantamento das necessidades institucionais por meio de entrevistas, questionários, e reuniões envolvendo alunos e professores. Para facilitar a implementação da plataforma, decidiu-se utilizar um Sistema Gerenciador de Conteúdo (*CMS – Content Management System*). Toda parte de acesso, perfis de usuários, questões de segurança, e alguns métodos já vêm pré-configurados pela plataforma.

A arquitetura é baseada em conceitos já existentes (como chat, fórum, blog), com pequenas adaptações feitas para facilitar a utilização das ferramentas do sistema por parte do usuário. A navegação foi elaborada para ser intuitiva, com uma visão de hierarquia entre o ambiente como um todo e cada agente do sistema, tais como comunidades, alunos e professores. Estudos comparativos entre CMSs foram realizados, sendo levados em conta apenas CMSs de código livre, e escrito em PHP.

Por fim, uma ferramenta foi implementada, para cuja elaboração foram realizados estudos aprofundados sobre o Drupal. O código é estruturado, porem vários conceitos de orientação a objetos são utilizadas pela plataforma como encapsulamento, herança, polimorfismo, etc.

ABSTRACT

This paper deals with practical questions concerning the building of software architecture for the web having as purpose the creation of a collaborative environment for educational institutions. A theoretical foundation was first established and then took place the survey of institutional needs by means of interviews, questions form and meetings wherein students and professor took part. In order to facilitate the platform implementation a CMS (Content Management System) has been employed. All access procedures, user profile, safety devices and some methods are preconfigured by the platform.

The architecture is based upon existing concepts (such as chat, forum, blog) and some adaptations have been introduced to facilitate the system tools utilization by the user. The navigation has been worked out to be intuitive with a view of hierarchy between the environment as a whole and each system agent such as communities, students and professors. Comparative studies between CMSs were carried out and only open source CMSs in PHP were taken into account.

Finally a tool has been implemented after extensive studies on Drupal. The code is structured but several object oriented concepts have been employed in the platform, such as encapsulation, hierarchy, polymorphism, on so on.

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

CONSIDERAÇÕES INICIAIS

Este trabalho tem seu foco na definição da arquitetura de um ambiente de colaboração educacional. O propósito é a produção de uma ferramenta voltada para criação de ambiente virtual em contexto de ensino presencial.

Experiências obtidas no desenvolvimento de *softwares* vêm trazendo a instituições e empresas preocupações com as questões de qualidade na construção de sistemas. Cada vez mais se tem a necessidade de definir processos internos de desenvolvimento, incluindo técnicas de modelagem e ferramentas de teste e ressaltando a importância da documentação como registro das decisões e definições adotadas.

A arquitetura de um sistema busca uma solução técnica para um problema. Uma arquitetura é definida de forma a satisfazer todos os requisitos (requisitos funcionais e requisitos não funcionais), além dos subsistemas e componentes que compõem este sistema. Com isso notamos que a arquitetura não engloba apenas informações tecnológicas e de infra-estrutura, outros fatores também devem ser levados em consideração.

Com o amadurecimento da tecnologia *Web*, muitos cenários são trazidos para facilitar a aprendizagem e a troca de informações entre estudante e professor. Atualmente, materiais como trabalhos, aulas, artigos e livros são facilmente disponibilizados em algum endereço *Web.* Com isso, surgiram inúmeros endereços, ambientes e plataformas para o estudante acessar e pesquisar conteúdos.

Assim, surgiu a idéia de se juntar todos esses recursos em um único ambiente. Os dois principais conceitos, comunidades e contatos (alunos e professores), trazem uma maior flexibilidade entre a troca de informação. Esta pode ser feita entre alunos<->alunos, alunos<->professores, professores<->professores, e não só entre Professor->Aluno como em muitos casos anteriores.

> *"O computador apresenta várias virtudes, entre elas a de possibilitar as diversas formas de relação, enriquecendo as experiências dos indivíduos, colaborando, portanto, em seu*

desenvolvimento e possibilitando também a construção do conhecimento pelo próprio sujeito, por meio de sua exploração autônoma e independente (MARTINS, 1999).*"*

Este trabalho baseia-se em um projeto de pesquisa, chamado "Espaço Conhecimento", implantado no Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento da Universidade Federal de Santa Catarina (PPEGC/UFSC) e na Engenharia de Produção e Sistemas da Universidade do Estado de Santa Catarina (EPS/UDESC). Seu desenvolvimento teve apoio da empresa Número Soluções e Sistemas Ltda, cuja equipe pertence o autor deste trabalho de conclusão de curso.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

Explorar a viabilidade da implementação de ferramentas para um ambiente colaborativo por meio do uso de CMS[1].

1.2.2 OBJETIVOS ESPECÍFICOS

Levantar requisitos para o ambiente de colaboração atendendo as necessidades das universidades públicas.

Estabelecer uma arquitetura de sistema *Web* para plataforma de apoio ao ensino presencial que atenda os requisitos levantados permitindo uma integração maior entre os principais atores que integram um meio acadêmico (professor, aluno).

Implementar parte dessa arquitetura demonstrando a viabilidade da construção de uma ferramenta básica que atenda as necessidades das universidades públicas por meio de um CMS.

1.3 JUSTIFICATIVA

Este trabalho visa atender as necessidades específicas das universidades públicas, no que se refere à modernização dos métodos de ensino e incremento nos instrumentos pedagógicos.

[1] CMS, Content Management System, ou em português Sistema de Gerenciamento de Conteúdos. É uma ferramenta que permite a um editor criar, classificar e publicar qualquer tipo de conteúdo em páginas web.

Destaca-se também pelo seu vasto referencial teórico que possui estudos realizados com foco principal para aplicações *Web*. Para que possa servir como material de apoio, para futuras pesquisas voltadas ao tema de ambientes de colaboração, que é de forte tendência em sistemas educacionais atuais.

1.4 ESTRUTURA

Para atender aos objetivos propostos, este documento foi estruturado da seguinte forma: O Capítulo um apresenta os objetivos e as justificativas para o estudo e desenvolvimento do trabalho em questão. O Capítulo dois disserta a fundamentação teórica do sistema, necessários para a compreensão deste como um todo. O Capítulo três aborda o *CMS* Drupal, escolhido para o desenvolvimento da plataforma. No Capítulo quatro temos os requisitos funcionais do sistema, resultado da etapa de levantamento de requisitos. No Capítulo cinco temos a arquitetura proposta, onde são detalhadas as especificações dos requisitos. O Capítulo seis demonstra o passo a passo do desenvolvimento da ferramenta de recados para o ambiente de colaboração. E por fim temos o Capítulo de conclusão deste trabalho, onde são abordados resultados alcançados e trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Desenvolver uma aplicação *Web* difere do desenvolvimento de uma aplicação *Desktop*. Quando tratamos de aplicações para *Desktop*, existe na memória do computador um programa executável sendo rodado, em conjunto com a sua interface seus comandos e variáveis. Nas aplicações *Web* isso não ocorre, em um ambiente *Web* não existe um programa na memória, mas sim requisições http (CONALLEN, 1999). Essas requisições que são entradas do lado do cliente (Browsers de internet) são enviadas a um servidor *Web* onde são processados para gerar uma resposta ao cliente. Assim temos uma típica arquitetura *Web* chamada Cliente-Servidor (BERNERS-LEE, 1989). Tecnologias diversas podem ser utilizadas do lado do cliente e do lado do servidor para o desenvolvimento de uma aplicação. No decorrer deste capitulo serão citados com mais detalhes a arquitetura *Web* (cliente-servidor) e suas tecnologias de ambos os lados.

Será visto também nesse capitulo, estudos realizados no âmbito de CMSs que ganham destaque na atualidade.

2.1 ARQUITETURA DE *SOFTWARE*

Antes do surgimento da arquitetura de *software*, o desenvolvimento de sistemas e aplicações de computação ocorria de forma desordenada e sem um padrão de estrutura e métodos o que gerou a "crise do *software*", denominada assim por Pressman (PRESSMAN, 1995). A solução encontrada para sanar esse problema foi a Engenharia de *Software* determinando "o uso de sólidos princípios de engenharia para que se possa obter economicamente um *software* que seja confiável e que funcione eficientemente em máquinas reais" (PRESSMAN, 1995).

> *"A arquitetura de software é colocada como uma ferramenta para lidar com a complexidade do software possibilitando a criação de um conjunto de componentes e definindo seus relacionamentos. O autor enfatiza que arquitetura deve satisfazer os requisitos funcionais e não funcionais do sistema. Portanto, é possível notar que a arquitetura de software é mais do que a descrição dos componentes que a compõem e do relacionamento entre eles"* (JAZAYERI, 2000).

Para Astudillo, "A arquitetura é a interface entre duas partes distintas: o problema de negócio e a solução técnica" (ASTUDILLO, 1998).

Como o sistema descrito neste documento se trata de uma plataforma de colaboração em ambiente *Web*, e o contexto de sistemas *Web* impõe problemas característicos, neste sentido a seguir detalharei a arquitetura *Web* denominada Cliente-Servidor e suas principais tecnologias.

2.1.1 ARQUITETURA DE SISTEMAS *WEB*

Existe na programação *Web* uma arquitetura chamada Cliente/Servidor, a comunicação entre o cliente e o servidor é feita sobre TCP/IP, onde é usado um protocolo de alto nível denominado HTTP (*Hipertext Transfer Protocol)* (BERNERS-LEE, 1989).

O Cliente são os *browsers* (navegadores *Web* como Firefox, Safári, Netscape, Internet Explorer) que executados em um host que solicitam informações ao servidor normalmente por meio da rede (BERNERS-LEE, 1989).

Servidores *Web* (exemplos: *Apache* e IIS) são aplicações que ficam em espera, e aguarda requisições do lado do cliente. A requisição vai para o servidor e este por sua vez verifica se é necessário que se rode uma aplicação por meio da extensão do arquivo. A aplicação é iniciada pelo servidor e recebe as informações em forma de entrada padrão e variáveis de ambiente. As informações são processadas pela aplicação que inclui acessos aos sistemas de informações ou aos repositórios de informações e produz uma saída, inclui ainda um *header* que informará ao servidor o tipo da informação produzida. O servidor recebe o *header* e os dados de respostas, e repassa em formato interpretável por *browsers Web* as informações para o cliente (WINCKLER & PIMENTA, 2002).

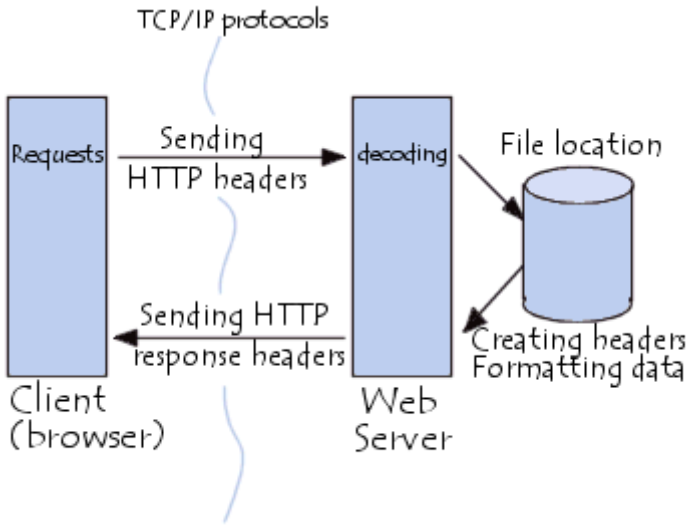


Figura 1 – Comunicação entre o navegador e o servidor - Fonte - http://pt.kioskea.net/contents/internet/http.php3

2.1.1.1 TECNOLOGIAS DO CLIENTE *WEB*

Serão a seguir citadas algumas tecnologias do lado do cliente, ou seja, tecnologias executadas pelo navegador *Web.*

2.1.1.1.1 HTML

HTML (*HyperText Markup Language)*, é uma linguagem de marcação que consiste basicamente na formatação do texto por meio de *tags*. Essas marcações também permitem a inclusão de *links*, imagens, tabelas etc. (MARCONDES, 2005) (BERNERS-LEE & CONNOLLY, Hypertext Markup Language – 2.0, 1995).

Em 1980, *Tim Berners-Lee* criou um sistema baseado em *hyperlinks* chamado *Enquire*, porém de uso privado. Só em 1989 o criador da *Web* começou a esboçar tecnologias com o intuito de publicá-las, surgindo o primeiro esboço da linguagem HTML. Inspirou-se nos padrões *HyTime* e SGML (*Standard Generalized Markup Language*) para o surgimento do HTML (LALLI, BUENO, & ZACHARIAS, 2008).

2.1.1.1.2 CSS

CSS é acrônimo para *Cascading Style Sheets*, uma linguagem de estilo que permite definir aspectos de apresentação de uma página HTML (por exemplo, tamanho e cor de fontes, espaçamento entre linhas, margens, *layout* etc). A CSS permite a separação entre o conteúdo e a formatação, sendo possível aplicar o mesmo estilo a várias páginas HTML. Sua última versão possui cerca de 100 propriedades e sua sintaxe é simples (SILVA, 2007) (CSS: conceitos e mão na massa, 2006).

2.1.1.1.3 SHOCKWAVE E FLASH

Shockwave e *Flash* são plugins multimídia para navegadores, são bastante utilizados para a reprodução de vídeos, animações e jogos. Esses objetos podem ser adicionados no HTML por meio da *tag* <OBJECT> e funcionam na maioria dos navegadores (BOARD, LUNA, & DELL, 1996).

2.1.1.1.4 JAVA APPLET

Java Applet é um *plugin* que é executado em uma maquina virtual e permite que uma aplicação Java seja executada no navegador. Surgiu junto com o Java no pacote JDK 1.0 e podia ser visualizado por meio do aplicativo *Sun's AppletViewer*. Ele pode ser embutido no HTML por meio da *tag* <APPLET> e funciona na maioria dos navegadores, em diversos sistemas operacionais (Developer Resources for Java Technology).

2.1.1.1.5 JAVASCRIPT

JavaScript é uma linguagem *script* bastante flexível, de tipagem dinâmica. É utilizada por milhares de paginas *Web* para adicionar funcionalidades, validação de formulários, detecção de *browsers*, e muito mais. A linguagem é interpretada pelo navegador *Web* e, portanto, executada do lado do cliente. Seu código pode ser embutido no código HTML ou em arquivo separado com a extensão 'js'. O *script* gerado é capaz de interagir com os objetos e *tags* do HTML, sendo por isso uma ferramenta extremamente poderosa (FLANAGAN, 2001).

2.1.1.1.6 ACTIVEX

Criado pela Microsoft o *ActiveX* surgiu na *Web* junto com o Internet Explorer 3, para concorrer com o *Applet* e o *Flash*. A principal diferença é que o *ActiveX* funciona como um executável comum do Windows, com acesso irrestrito ao *hardware* e ao sistema de quem instala o aplicativo (CHAPPELL, 1996).

2.1.1.1.7 XHTML

XHTML é uma versão mais clara e mais rigorosa do HTML. É o acrônimo para *eXtensible Hypertext Markup Language*. O XHTML é mais acessível, mais fácil de manter e de processar, a ambigüidade em sua interpretação diminui e isso facilita também a portabilidade para um maior número de plataformas de navegação (por exemplo, dispositivos móveis, celulares, carros, etc.). A versão XHTML 1.0 surgiu do HTML 4.01, com a sintaxe modificada para se adequar às regras XML (SILVA, 2007).

2.1.1.2 TECNOLOGIAS DO SERVIDOR *WEB*

A seguir serão citadas tecnologias no lado do servidor. Cada servidor deve estar preparado para interpretar ou executar determinada linguagem. Por exemplo: o *Apache* possui CGI nativo e bibliotecas plugáveis para PHP e outras linguagens; o servidor *Web Tomcat* tem suporte ao JSP nativo; o servidor IIS (*Internet Information Services)* é capaz de processar páginas ASP.

2.1.1.2.1 CGI E PERL

CGI (*Common Gateway Interface)* é um padrão que define a interface de comunicação entre um servidor *Web* e uma aplicação qualquer. Essa aplicação é executada do lado do servidor e se comunica com o servidor *Web* por meio da entrada e saída padrões do aplicativo, seguindo determinadas regras predefinidas. O CGI foi o primeiro passo para que fosse possível a geração de conteúdo dinâmico por parte do servidor (Site oficial do CGI). Sua principal restrição é que o aplicativo tem que ser desenvolvido de forma compatível com a plataforma do servidor *Web*, ou seja, o mesmo aplicativo compilado feito em C para Linux não executaria num servidor que estivesse no Windows, a não ser que seja re-compilado (Especificação oficial do CGI na IETF (RFC 3875)).

No inicio do CGI, desenvolvedores notaram que a linguagem *Perl* era muito adequada ao desenvolvimento de aplicações *Web*, pelo seu poder com a manipulação de textos principalmente. Assim sendo, a *Perl 5.0* já contava com algumas melhorias voltadas, sobretudo para a interface CGI. A partir da versão 5.0, a linguagem *Perl* passou a ser largamente utilizada pelos desenvolvedores *Web* e as bibliotecas se tornaram cada vez mais avançadas. Das bibliotecas para *Perl* surgiu o PHP, que no início era apenas uma coleção de ferramentas em *Perl* e C criadas por Rasmus Lerdorf (ASHTON, 1999-2001).

2.1.1.2.2 RUBY

Ruby é uma linguagem de programação interpretada, é orientada a objetos de tipagem dinâmica e que possui algumas características funcionais, como *closures*, *continuations* e *bindings*. Yukihiro Matsumoto lançou a primeira versão beta da *Ruby* em dezembro de 1994

com o objetivo de criar uma linguagem mais poderosa que *Perl* e mais orientada a objetos que *Python* [2] (Linguagem de programação Ruby). Em julho de 2004, David Heinemeier Hansson lançou um *framework* completo para o desenvolvimento em *Rail*, chamado *Ruby on Rails*. Apesar de recente, o *RoR* (*Ruby on Rails*) é bastante difundido entre os desenvolvedores *Web* (HANSSON, 2006).

2.1.1.2.3 PHP

PHP (*Hypertext Preprocessor)*, desenvolvida especialmente para a *Web* é uma linguagem de código aberto. A sua idéia é que seja possível embutir trechos PHP em código HTML. Como o PHP é escrito diretamente junto com o HTML, o servidor fica encarregado de processar e transformar o código misto (PHP e HTML) em um HTML puro antes de retornar a página para o usuário. O PHP é extremamente simples para um iniciante, mas oferece muitos recursos para o Programador profissional. Isso permitiu que pessoas sem muita especialidade em programação pudessem fazer *sites* dinâmicos de forma mais fácil. Esse estilo inspirou linguagens como JSP e ASP (CONVERSE & PARK, 2003).

2.1.1.2.4 COLDFUSION MARKUP LANGUAGE

ColdFusion Markup Language, também conhecida pelo acrônimo CFML, é uma linguagem baseada em *tags*, semelhante a um código HTML. Por ser uma linguagem muito simples, é utilizada por leigos em programação, sendo adotada muitas vezes por eles como sua primeira linguagem de programação para Internet (ASTUDILLO, 1998).

2.1.1.2.5 ASP

ASP (*Active Server Pages*), tecnologia que permite a inclusão de código de outras linguagens embutido no HTML, no estilo do PHP. ASP é uma estrutura de programação (não uma linguagem) em Script que podem ser: JScript, VisualBasic, PerlScript ou Python. A

[2] Python é uma linguagem interpretada, bastante portável, orientada a objetos (incluindo herança múltipla).

tecnologia ASP foi inspirada no PHP, sendo bem aceita na comunidade *Web* por permitir que programadores experientes em VisualBasic pudessem criar páginas dinâmicas facilmente. A desvantagem é que muitos componentes necessários para a programação *Web* como *upload* de arquivos, FTP etc. são geralmente pagos. A partir da versão 3.0 o ASP passou a fazer parte da plataforma.NET e outras linguagens, como o C#[3], passaram a ser também suportadas (LEVINE & WALTHER, 2003).

2.1.1.2.6 JAVA, SERVLET E JSP

Inspirada no C++ Java é uma linguagem orientada a objetos desenvolvida pela Sun. O Java foi desenvolvido especificamente com o objetivo de ser uma tecnologia multiplataforma, ou seja, permitir o funcionamento de programas independentemente do dispositivo onde fossem usados devido a sua execução se dar em uma maquina virtual. *Servlet* é uma classe especial do Java feita especialmente para interagir com um servidor *Web* preparado para executá-la (por exemplo: *Tomcat* do *Apache* ou *GlassFish* da *Sun*). JSP (*JavaServer Pages)* surgiu na versão enterprise do Java (J2EE). Essa linguagem serve para a geração automática de *Servlet* por meio de um código intermediário no estilo do PHP e ASP, ou seja, em que é possível escrever código Java embutido no HTML. Ao receber uma requisição de página do usuário, o servidor processa o código JSP, gera um *Servlet* e o executa, devolvendo o HTML gerado por esse código ao usuário (Developer Resources for Java Technology).

2.1.1.3 EXECUÇÃO MISTA

Paralelo às aplicações cliente/servidor temos tecnologias que oferecem uma execução mista que geralmente são utilizadas em conjunto outras tecnologias anteriormente citadas. O uso de aplicações no lado do Cliente em conjunto com aplicações no lado do Servidor enriquece

[3] O C# é uma linguagem de programação com finalidade geral, simples, orientada por objetos e fortemente tipada. O Visual C# fornece aos desenvolvedores focados no código ferramentas e suporte a linguagens poderosos para a criação de aplicativos cliente e web conectados e avançados no .NET Framework.

muito a experiência de navegação na *Web*. Fazer aplicações somente no lado do Cliente tem a desvantagem que os dados não podem ser salvos em lugar algum. Nos casos em que isso é possível, a exemplo do *ActiveX*, a informação fica guardada apenas em uma máquina e não é compartilhada na rede. Fazer aplicações somente no lado do Servidor causa um lapso de tempo relevante entre a requisição e a sua resposta (APPLE, 2005). Ainda, toda resposta recebida do servidor precisa ser totalmente recarregada sem algum mecanismo de processamento paralelo no lado do cliente. Assim começa a surgir a possibilidade de integrar tecnologias de forma assíncrona, ou seja, sem a espera por toda a carga de uma página e a tecnologia que ganha destaque nesse sentido é o Ajax.

2.1.1.3.1 AJAX

Ajax (*Asynchronous JavaScript and* XML), é baseado em *JavaScript* e solicitações http. Faz parte de um esforço para aumentar nos aplicativos Internet, além da interatividade, velocidade, funcionalidade e usabilidade. Apesar de a proposta desta tecnologia ser possível desde março de 1999 com o Internet Explorer 5, o termo Ajax só foi empregado em fevereiro de 2005, quando se tornou mais popular. Ajax não é uma nova linguagem de programação, mas uma nova forma de utilizar as normas existentes. O uso do Ajax também é bastante associado com a *Web* 2.0 (GARRETT, 2005). Com a tecnologia Ajax é possível a criação de aplicativos *Web* mais dinâmicos, que interagem em tempo real com um banco de dados no lado do servidor, não tendo a necessidade de recarregar uma página *Web* inteira a cada nova requisição do usuário.

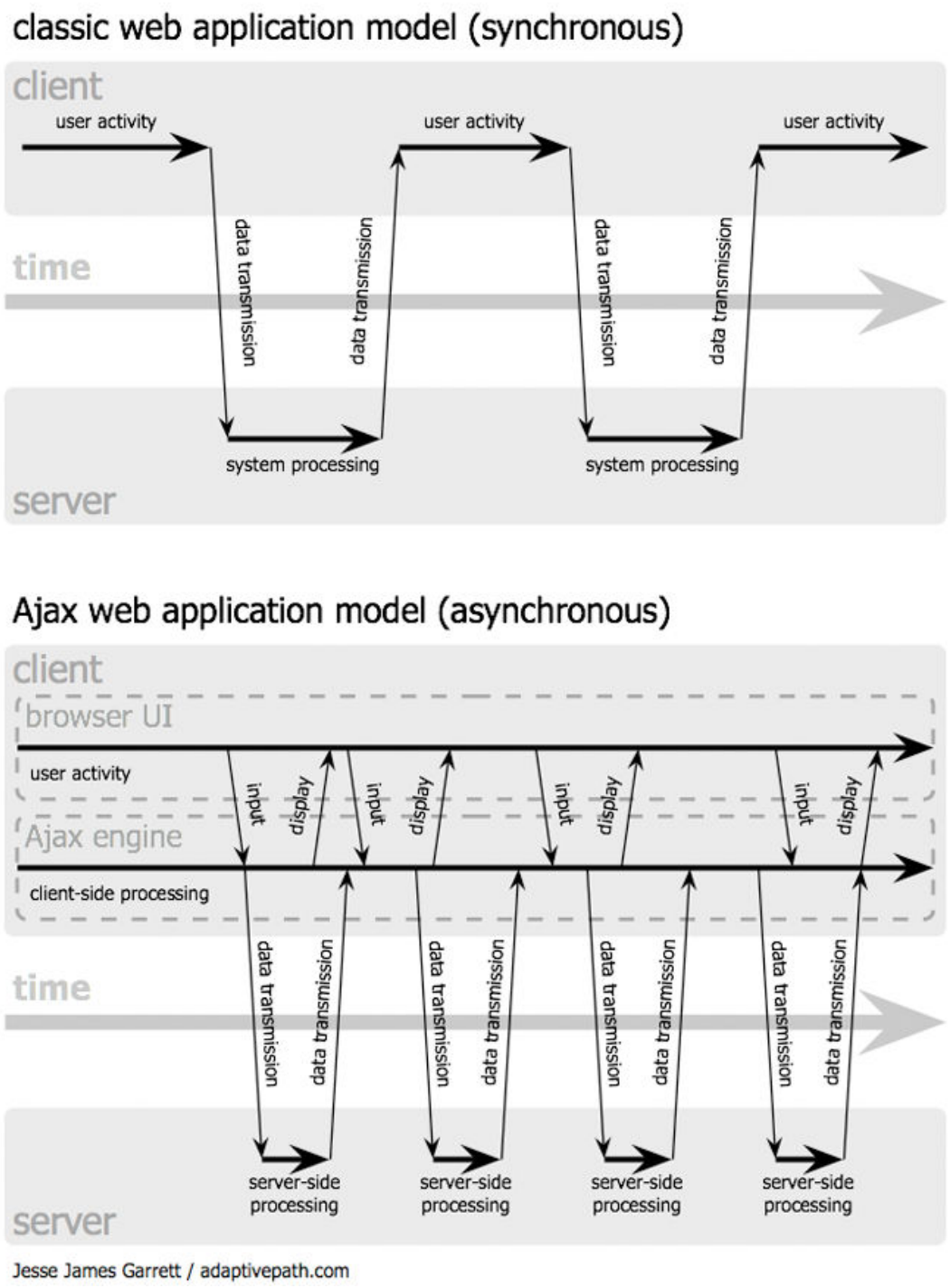


Figura 2 – Parte superior mostra a interação síncrona, padrão tradicional de uma aplicação *Web*, e a parte inferior a interação assíncrona de um pedido – Fonte: artigo Ajax: *A New Approach to Web Applications*.

2.2 AMBIENTES DE COLABORAÇÃO

Produzem-se potencialmente melhores resultados trabalhando colaborativamente do que se os membros do grupo atuassem individualmente. Em um grupo, a capacidade, conhecimento e o esforço individual de um dos membros podem complementar o dos outros. Há também interação entre pessoas com entendimentos, pontos de vista e habilidades complementares (BERNERS-LEE T, 1989). A informação não deve ser de fonte única e sim uma por meio de diversas fontes. Cada fonte de informação deve ter suas particularidades para atender corretamente em um dado momento as necessidades de um aluno na busca por novos conhecimentos. Nosso mundo se caracteriza pela abertura, flexibilidade, mudanças constantes e necessidade de respostas rápidas e nem sempre um aluno conta com a presença constante de um professor (BATES, 2005).

Trabalhar colaborativamente traz motivação para cada membro de um grupo. Seu trabalho estará sendo observado, comentado e avaliado por pessoas do grupo da qual faz parte (BERNERS-LEE, 1989). A responsabilidade de ensinar estará distribuída entre diversas fontes de informação não ficando presa somente a um professor, este passará a ser visto como um facilitador e orientará como e onde os alunos devem buscar as informações de seu interesse. O mundo atual tem feito com que as pessoas desejem e necessitem estudar em qualquer local, no ritmo e no tempo de sua necessidade ou escolha. As pessoas estão cada vez mais sem tempo para se afastar de suas atividades do dia a dia do trabalho e envolvidas com as tecnologias da informação (KOSCHMANN, 1996).

A Internet e a tecnologia da informação, usadas de modo inteligente, são excelentes ferramentas de auxílio à aprendizagem, incentivando o compartilhamento de experiências e conhecimentos entre os membros envolvidos (BATES, 2005). Ambientes de colaboração via Internet não devem substituir, mas sim estender e enriquecer a forma de ensino tradicional. O professor trabalhando em conjunto a um ambiente de colaboração continuará sendo uma peça essencial na aprendizagem (ARAÚJO & ROCHA).

2.2.1 CONCEITOS E FERRAMENTAS

Ambientes de colaboração utiliza-se de ferramentas de modo a facilitar a execução de trabalhos em grupo. Esses ambientes devem oferecer ferramentas especializadas o suficiente

para proporcionar formas de interação aos seus usuários, a fim de facilitar a comunicação, a colaboração e a coordenação entre as partes envolvidas que fazem parte de um grupo de trabalho (KOSCHMANN, 1996). Não deve se limitar a pessoas de próxima localização geográfica, e nem a presença das partes envolvidas num dado momento, pois a interação poderá acontecer ao mesmo tempo ou em tempos diferentes dependendo da ferramenta utilizada. Nota-se então, que o foco dos ambientes de colaboração é diminuir as barreiras impostas pelo espaço físico e tempo (CAMARGO, 2005).

A Figura 3 mostra de forma esquemática a estrutura de um Ambiente de Colaboração

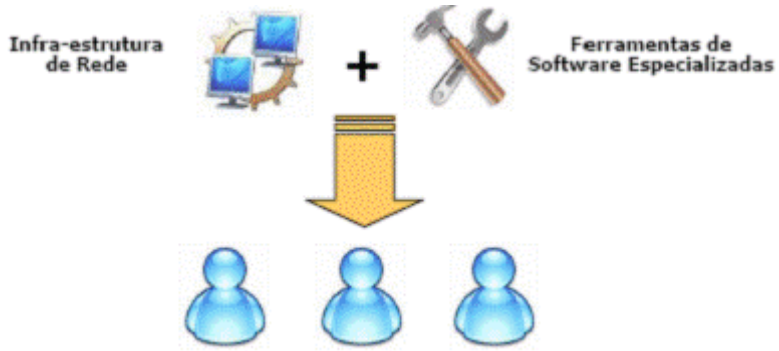


Figura 3 - Estrutura de um Ambiente de Colaboração - Fonte: CAMARGO, 2004 p. 4

Ferramentas de colaboração podem ser classificadas de acordo com o lugar das interações (presenciais ou à distância) e o tempo (síncronas ou assíncronas) (BATES, 2005). Ferramentas síncronas exigem tempo de resposta imediato, como mostra a Figura 4. Por exemplo, mensagens instantâneas (ICQ, Messenger), conferências e videoconferências (CAMARGO, 2005).

Figura 4 - Ferramentas Síncronas - Fonte: CAMARGO, 2004, p. 8

As ferramentas assíncronas por sua vez, não necessitam de um tempo de resposta imediato, conforme a Figura 5. Emails, fóruns de discussão são ótimos exemplos de ferramentas assíncronas. Ferramentas de fluxo de trabalho (*Workflow*) e calendários (*Groupware*) também

são consideradas ferramentas assíncronas (CAMARGO, 2005) (SCHWIER & BALBAR, 2002).

Figura 5 - Ferramentas Assíncronas - Fonte: CAMARGO, 2004, p. 8

2.2.1.1 REDES SOCIAIS

Uma rede social é uma estrutura social, comunidade, ou sociedade feita de perfis de usuários que são geralmente indivíduos ou organizações. Isto indica o modo como eles estão conectados por meio de várias familiaridades sociais, afiliações, e/ou relacionamentos que variam desde conhecidos casuais a ligações familiares próximas. A rede é responsável pelo compartilhamento de idéias entre pessoas que possuem interesses e objetivo em comum e também valores a serem compartilhados. Assim, um grupo de discussão é composto por indivíduos que possuem identidades semelhantes (WASSERMAN, 1994).

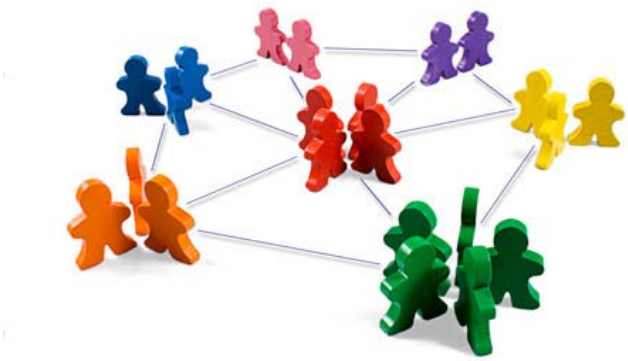


Figura 6 - Esquematização de uma rede social – Fonte: http://moodlelivre.wordpress.com

Existem vários tipos de redes sociais, como família, trabalho, diversão, espiritualidade, políticas, geográficas, de tópicos, de atividades. Nas redes sociais o indivíduo tem três principais características: identidade, reputação, e confiança (WASSERMAN, 1994).

2.2.1.2 MÍDIAS SOCIAIS

Mídias sociais são utilizadas para disseminar conteúdo, compartilhar idéias, perspectivas, experiências e opiniões. As mídias sociais são a democratização da informação, e transforma o leitor de conteúdos em conteudista. Podem se encontrar em diversos formatos, podendo englobar textos, imagens, áudio, e vídeo. Nas mídias sociais o usuário tem o controle, é uma tendência que já se pode verificar que funciona. As possibilidades não têm fim numa mídia social. A facilidade na sua utilização é o principal atrativo para cada vez mais novos adeptos a sua utilização (FONTOURA, 2008).

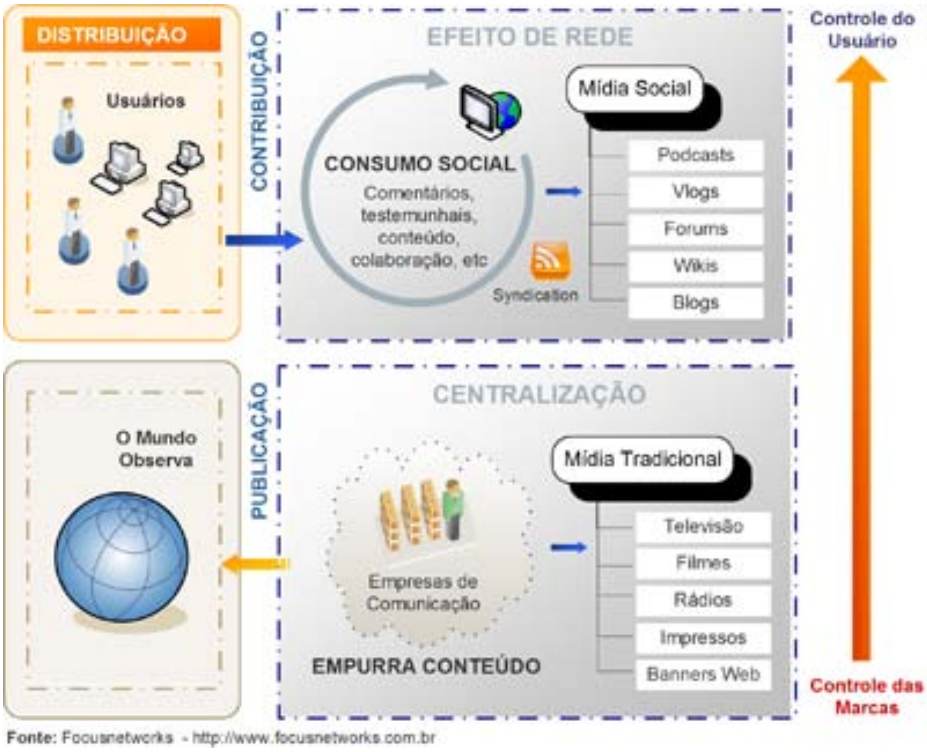


Figura 7 - Esquematização de mídias social – Fonte: http://www.focusnetworks.com.br/Noticias.aspx?v=1&nid=116

Exemplo da aplicação das mídias sociais, *blogs*, mensageiros, *podcasts*, *wikis*, social *advertising*, redes sociais, aplicativos de presença, compartilhamento de vídeo, realidade

virtual, agregadores de notícias, compartilhamento de fotos, *livecasting*, vídeo *on-line* episódico, *bookmarking* social, *games on-line*, social *shopping*, e até motores de busca otimizados em relação a grupos de interesse. As mídias sociais permitem que seus usuários possam interagir instantaneamente entre si e com o restante do mundo (FONTOURA, 2008).

2.2.1.3 FÓRUM

Ferramenta com o intuito de realizar debates sobre um determinado tema de forma assíncrona e encadeada. Podem ser referenciados também como fóruns *Web*, fóruns de discussões, quadro de mensagens, quadro de discussões, grupos de discussões, quadro de boletins, fora (plural de fórum em latim) ou somente fóruns. Um fórum engloba inúmeros sub-fóruns que por sua vez trata de um tópico especifico. Mensagens no sub-fórum aparecem na ordem cronológica ou em linhas de discussões (BRITO & PEREIRA, 2004).

2.2.1.4 *BLOG*

Blog (contração da expressão "*Web log*") é um *Web site* ou pode estar inserido em um, normalmente é mantido por um indivíduo que sempre o matem atualizado regularmente por meio de comentários, descrições de acontecimentos, ou outros materiais, tais como gráficos ou de vídeo. As atualizações são geralmente exibidas em ordem cronológica decrescente (PEREIRA, 2008).

2.2.1.5 *CHAT ROOM* (BATE-PAPO)

Um *chatroom* ou *chat room* é um termo usado primeiramente pela mídia de massa para descrever qualquer forma de conferência síncrona, ocasionalmente também assíncrona. O termo pode assim significar qualquer tecnologia variando de *chat online* em tempo real sobre *instant messaging* e fóruns *online* a ambientes gráficos sociais totalmente imersivos (BRITO & PEREIRA, 2004).

2.3 CMS

O CMS (*Content Management System*, ou em português, Sistema de Gerenciamento de Conteúdo) é na verdade um sistema gerenciador de *websites*, portais e intranets que agrega ferramentas necessárias para gerenciar (inserir, editar, remover) conteúdo em tempo real, não havendo a necessidade de programação por meio de linhas de código (BAX & PEREIRA, 2002). O CMS é uma ferramenta criada para facilitar a vida dos usuários. O objetivo é estruturar e facilitar a administração, criação e publicação de conteúdo de forma dinâmica, por meio de uma interface de usuário via Internet (BAX & PEREIRA, 2002) (RAVEN, 2007).

O CMS funciona como um *framework*, um "esqueleto" de *website* pré-programado, com recursos de administração e manutenção disponíveis. Um *website* criado com um CMS tem sua aparência facilmente customizável por meio da utilização de estilos CSS[4] e modelos (*templates*) que podem ser facilmente adaptados (libertas.pbh.gov.br, 2007).

Um CMS pode ser transformado também em um excelente ambiente para o processo de ensino e aprendizado. É utilizado muitas vezes para a organização da informação em ambientes com fins educacionais, surgindo assim o LMS - *Learning Management System* (Sistema de Gestão de Aprendizagem).

> *"a funcionalidade de um CMS pode ser subdividida em: criação e gerenciamento do conteúdo, publicação e apresentação. A criação do conteúdo é feita por um ambiente de autoria com facilidades de edição, que permitem a criação de novas páginas ou atualização de conteúdo sem a necessidade de se conhecer linguagens de programação"* (ROBERTSON, 2003).

> *"um Sistema de Gerenciamento de Conteúdo é um conjunto de processos, aplicações, e bancos de dados que auxiliam uma organização a criar, armazenar, coordenar, e publicar informação em um formato útil, agradável ao usuário, e com um padrão consistente"* (ROBERTSON, 2003).

[4] CSS é a sigla em inglês para *Cascading Style Sheet* que em português foi traduzido para folha de estilo em cascata. É um mecanismo simples para adicionar estilos (p.ex., fontes, cores, espaçamentos) aos documentos *Web*.

2.3.1 ESTUDOS DE CMSs

São inúmeras as opções de CMS no mercado. Atualmente, porem serão considerados somente CMSs mais populares e na categoria de *software* livre para efeito de pesquisa.

Os mais utilizados são os de código aberto (*Open Source*) sob termos de licença GNU (GPL - *General Public License*) e que utilizam ambiente composto por *softwares* livres (como Apache, Tomcat, Linux, MySQL) (PACHECO, 2007). Serão destacadas suas qualidades para uma posterior análise dos requisitos e escolha da melhor ferramenta para o desenvolvimento do sistema para o Ambiente de Colaboração Educacional.

Antes de tratar destes sistemas individualmente é importante mencionar as principais características e funcionalidades comuns a quase todos eles, como tais:

- Fóruns (troca de mensagens por meio de tópicos);
- Pesquisas por meio de enquetes;
- Estatísticas de acesso;
- Gerenciamento de usuários;
- Gerenciamento de Temas para cada usuário;
- Integração de usuários por meio de sistemas de mensagens privadas;
- Envio de notícias (*newsletter*);
- Gerenciamento de grupos de usuários;
- Sistema de moderação para controle de usuários;
- Ferramenta de busca;
- Geração de RSS[5] - formato de distribuição de informações pela Internet, como por exemplo, notícias.

[5] RSS é um formato baseado na linguagem XML cada vez mais utilizado para a distribuição de conteúdos. Com o RSS é possível reunir em um único ambiente conteúdos produzidos por diversas fontes, sem a necessidade de acessar cada um dos sites responsáveis por eles.

2.3.1.1 MOODLE

O Moodle (*Modular Object-Oriented Dynamic Learning Environment*) (moodle.org) é um ambiente de apoio à aprendizagem, executado em um ambiente virtual. Criado em 2001, pelo cientista computacional e educador Martin Dougiamas, e direcionado para programadores e acadêmicos da educação. É um sistema de administração de atividades educacionais que se destina à criação de comunidades *on-line*, em ambientes virtuais de aprendizagem colaborativa. Um estudante ou um professor pode facilmente integrar-se, estudando ou lecionando, num curso *on-line* de sua escolha.

É um *software* livre (ver licença GPL) e pode ser instalado em diferentes ambientes (Windows, Unix, Linux, Mac OS) que tenham os requisitos necessários para executar a linguagem PHP. Sua base de dados pode ser MySQL, *PostgreSQL*, *Access*, *Oracle*, ODBC ou *Interbase*. É desenvolvido por uma comunidade virtual que reúne programadores e desenvolvedores de *software* livre, administradores de sistemas, *designers* instrucionais, professores e usuários de todo o mundo. É disponibilizado em diversos idiomas, inclusive em português.

Diversas instituições de ensino superior e básico trabalham com o Moodle nos seus próprios conteúdos. A plataforma é utilizada também para atividades que envolvem formação de grupos de estudo, treinamento de professores e até desenvolvimento de projetos. Empresas privadas, grupos independentes e ONGs, ou seja, setores não ligados à educação, também utilizam o Moodle quando necessitam interagir colaborativamente na Internet.

Os cursos Moodle podem ser configurados em três formatos, de acordo com a atividade a ser desenvolvida:

- Social – Quando o tema é articulado em torno de um fórum publicado na página principal
- Semanal - no qual o curso é organizado em semanas, com datas de início e fim;
- Em Tópicos - Onde cada tema a ser discutido representa um tópico, sem limite de tempo pré-definido. Os recursos disponíveis para o desenvolvimento das atividades

são: Chat; Diário; Pesquisa de Opinião; Questionário; Fórum; Glossário; Lição e tarefa; Materiais (transparências e texto das aulas); Avaliação do curso; Trabalho com Revisão; *Wiki*.

2.3.1.2 DRUPAL

O Drupal é um CMS rápido e com muitos recursos. Já vem com módulos para criação de *blog*, fórum de discussão, matérias (com RSS criados automaticamente) e páginas de Internet. Ainda é possível criar *sites* de comércio eletrônico, classificados, bate-papo, *wikis*, leitores de RSS e muitas outras ferramentas, bastando apenas usar as dezenas de módulos gratuitos disponibilizados no sítio. Temas para mudar o visual e traduções também estão disponíveis gratuitamente pela comunidade (drupal.org).

Administradores podem escolher entre múltiplos temas ou criar o seu próprio para dar ao *site* uma aparência única. O sistema de classificação flexível permite classificações hierárquicas, indexação cruzada de postagens e definição de categorias múltiplas para a maioria dos tipos de conteúdo. O acesso ao conteúdo é controlado por meio de definições de papéis pelo administrador. Páginas do *site* podem exibir mensagens por tipo de módulo ou conteúdo categorizado, com exportação em RSS separada por cada tipo de exibição. Usuários também podem fazer busca por palavra chave em todo o *site*.

A ferramenta Drupal é escrita em PHP e funciona em qualquer sistema operacional (Windows, Linux, entre outros) e servidores *Web* (Apache, IIS). O núcleo do Drupal foi bem projetado com um sistema de “ganchos” conhecido como *hooks*, ou *callbacks*, que permite que módulos insiram funcionalidades dentro do Drupal.

Modular e extensível. O Drupal tem por objetivo prover um núcleo que suporte ser estendido por meio de módulos personalizados, código documentado é uma prioridade sobre funcionalidades desordenadas.

O Drupal é baseado na de colaboração de *software* livre por meio do código aberto, bem como é liberado sob a licença GPL. Especificamente o Drupal é codificado na linguagem PHP e tem como formato primário de fonte de dados os bancos de dados MySQL ou PostgreSQL.

2.3.1.3 PHP-NUKE

O PHP-Nuke (phpnuke.org) é um sistema para publicação automatizada de notícias para a *Web* e um CMS baseado em PHP e MySQL. O Sistema é totalmente controlado por meio de uma interface *Web*. PHP-Nuke originou-se como derivação do sistema de portais de notícias *Thatware*. Sistema é desenvolvido sob a Licença Pública Geral GNU. O *software* é liberado de duas maneiras: a primeira é a versão estável e gratuita, e a segunda, onde o usuário contribui para o desenvolvimento do *software* e paga para fazer o *download*. Isso é permitido pela Licença GNU GPL (desde que o código do *software* esteja incluído), mas o comprador do *software* tem total liberdade para distribuir o código do produto.

Para que ele funcione, é necessário um servidor de páginas que suporte a extensão PHP, assim como um Banco de Dados SQL (MySQL, mSQL, *PostgreSQL*, ODBC, *Adabas*, *Sybase* ou *InterBase*).

Módulos podem ser adicionados ao sistema do PHP-Nuke, permitindo ao *webmaster* adicionar mais serviços (como uma galeria de fotos ou um calendário de eventos) ao seu PHP-Nuke em complementação aos módulos padrão que acompanham a distribuição original do sistema, como Notícias, FAQ e Mensagens Privadas. O PHP-Nuke possui suporte a muitos idiomas, inclusive português. Seu visual e interface gráfica podem ser customizados por ter suporte a temas (como no Drupal).

Vários projetos criaram derivações desse sistema. Os mais conhecidos são: eNvolution, myPHPNuke, NPDS, openPHPnuke, phpWebSite, Postnuke, XOOPS e Xaraya.

Estas derivações, como muitas outras, possuem suas próprias idéias e linhas de desenvolvimento, e todos tentaram solucionar os mesmos problemas e tornar seus sistemas melhores e mais seguros.

2.3.1.4 JOOMLA

Joomla! (joolma.org) é um CMS desenvolvido a partir do Mambo. É escrito em PHP e roda no servidor *Web* Apache ou IIS e banco de dados MySQL. O Joomla é um projeto de código

aberto com licença GNU/GPL. Hoje, mesmo com pouco mais de um ano desde a primeira versão, o Joomla! é o CMS em maior expansão, sendo provavelmente o CMS mais procurado, com a maior comunidade e recursos disponíveis. A grande vantagem do Joomla é sua diversidade de extensões extras, feitas não necessariamente pelos desenvolvedores do projeto. Componentes, módulos e *plugins* são atualizados constantemente e possuem grande valia para profissionais de *Web* interessados em um *site* bem feito.

O conceito de componentes do Joomla talvez seja uma das grandes vantagens em relação à maioria dos CMS disponíveis. O componente é uma forma de gerenciar conteúdos ou agregar funcionalidades muito específicas que não seria possível com as funções padrões do Joomla. Por exemplo, o componente *Web Links* permite gerenciar a área de *links* do *site*. O componente *Banners* permite gerenciar *banners*, rotacionando aleatoriamente e contando o número de cliques.

Existem centenas de componentes sendo que em grande parte são gratuitos (GNU GPL), porém alguns possuem licença comercial. Caso não exista um componente adequado à necessidade do *site*, pode ser feita a adaptação de um componente existente ou pode-se criar um componente específico.

Vantagens de utilizar o Joomla:

Existem vários módulos e componentes disponíveis, herdados do Mambo; A equipe de desenvolvimento está fortemente comprometida com o projeto e está disposta a modernizar o *software* continuamente; É um dos CMS com mais recursos disponíveis e de fácil utilização; Foi escrito com PHP e MySql, dois dos *softwares open source* mais populares; O Joomla! Recebeu o prêmio *Linux Awards*, além de possuir uma comunidade ativa, que está sempre buscando inovações.

2.3.1.5 MAMBO

Assim como os demais sistemas apresentados, baseia-se no conjunto Apache, MySQL e PHP (mambo-foundation.org). Foi criado pela empresa *Miro International* no início de 2000 (versão 1.0). Atualmente o *Mambo Server* é utilizado por milhares de pessoas e empresas ao

redor do mundo, desde os mais simples *websites* pessoais até grandes portais de empresas como Porsche e Mitsubish.

Além de seu uso, o *Mambo Server* mantém uma comunidade ativa ao redor do projeto que conta com mais de vinte mil desenvolvedores, mil projetos abertos e algo em torno de vinte e três mil usuários de seus fóruns de discussão relacionados a questões que vão desde a forma de licenciamento quanto o desenvolvimento ou suporte para plataformas específicas. O grupo que trabalha diretamente no desenvolvimento do *Mambo Server* é dividido em vários times que cuidam das novas versões, documentação, testes, suporte e outras áreas. Então, com estes times trabalhando paralelamente é possível manter o produto operacional enquanto novas funcionalidades vão sendo desenvolvidas.

Junto com estes times existem vários outros formados por pessoas que trabalham, por exemplo, com a sua internacionalização (vários idiomas) e também em outras áreas como componentes de terceiros e legislação. Estas divisões são necessárias devido a grandiosidade da ferramenta e devido a forma de trabalho colaborativo imposta pelo modelo FLOSS.

2.3.1.6 XOOPS

XOOPS (xoops.org) é um CMS sob os termos da licença GPL. Tem como requisitos de sistema: um servidor *Web* com PHP com programação orientada a objetos com um banco de dados MySQL.

Sua maior característica é a facilidade de instalação, operação e o fato de existirem uma infinidade de módulos que possibilitam agregar funções ao portal que se deseja criar. É uma ferramenta flexível e fácil de usar na criação e administração dos mais variados *sites* ou portais pequenos, médios ou grandes, sendo capaz de estruturá-los. Todas as ações são efetuadas por meio de uma interface *Web* simples e funcional, deixando aos administradores, praticamente só a tarefa de gerenciar o conteúdo do *site*.

Um *site Web* em XOOPS pode ser gerenciado sem a necessidade de ferramentas externas ou conhecimentos avançados de tecnologias da Internet, como FTP, HTML, JavaScript, CGI etc. Contando com diversos recursos modulares: administração de associados, troca simples de *lay-out*, além da inserção e administração visual de conteúdos.

O *software* facilita a atualização, alteração e o gerenciamento de publicações eletrônicas em rede, pois as páginas da publicação são geradas dinamicamente, a partir de um banco de dados. O sistema pode ser operado por meio de *softwares,* cliente *Web* gráficos, que seguem os padrões de navegadores *Web* do mercado. E é composto também por base de dados e documentos *on-line* ou eletrônicos.

O XOOPS está em constante evolução. Ele vem sendo desenvolvido por meio de um processo organizado e efetivo, sempre levando em conta a opinião da comunidade mundial. Oito equipes foram criadas para assegurar um desenvolvimento coerente, consistente e contínuo. São elas: Desenvolvimento do Núcleo (*core*), Desenvolvimento de Módulos, Desenvolvimento de Temas, Controle de Qualidade, Suporte, Documentação, Relações com as Comunidades

2.3.1.7 WORDPRESS

WordPress é um CMS escrito em PHP e em MySQL, e desenvolvido especialmente para a criação de *blogs* (wordpress.org). Foi criado a partir do b2/*cafelog* e é hoje, junto com o *Movable Type,* o mais popular na criação de *weblogs*. As causas do seu rápido crescimento são, entre outras, seu tipo de licença código aberto, sua facilidade de uso e suas características como gerenciador de conteúdos. Tem como características: gera XML, XHTML, e CSS em conformidade com os padrões W3C; gerenciamento de ligações integrado; estrutura de *permalink* amigável aos mecanismos de busca; suporte extensivo a *plug-ins*; categorias aninhadas e múltiplas categorias para artigos*; TrackBack* e *Pingback*; filtros tipográficos para formatação e estilização de texto corretas, e suporta páginas estáticas e múltiplos autores.

3. DETALHAMENTO DO CMS SELECIONADO

Alguns critérios foram considerados para a decisão do CMS a ser utilizado, primeiro, todas principais plataformas abertas são desenvolvidas na linguagem PHP, com utilização do banco de dados MySQL ou PostgreSQL. Ou seja, independente da plataforma base, o sistema será escrito em PHP e utilizará ou MySQL ou PostgreSQL.

Segundo, parece claro que o Wordpress se destaca na criação de *weblogs* e não é destinado aos objetivos deste projeto e somente a parte dele, limitando a seis possíveis plataformas base. Embora o Moodle seja um *software* muito utilizado, na verdade é um LMS (*Learning Management System),* ele é muito bom como plataforma de conteúdos, mas não como plataforma de aplicativos; ambientes muito fora do seu paradigma de funcionamento exigirão mudanças profundas no código. Caso o projeto desvie do modelo do Moodle, ele também não poderá ser utilizado.

Investigando a fundo cada plataforma e suas comunidades ativas, nota-se que em termos de código e API, o Drupal seria a plataforma com a mais vasta documentação (api.drupal.org). O Drupal venceu o ultimo premio geral de melhor CMS *(Overall Open Source CMS Award 2008)* (Open Source CMS Award). Apesar de praticamente não apresentar construtores de classe, aplica princípios de orientação a objetos como encapsulamento, herança, polimorfismo, etc. Sua API além de bem documentada tem uma busca que facilita na hora de necessidade de consultas.

Por ser a primeira experiência com CMS, decidiu-se pela facilidade de pesquisa a documentação e por meio de maior compatibilidade com as necessidades da plataforma, o Drupal como CMS. E para descrever a arquitetura de *software* devemos antes conhecer o Drupal e a sua arquitetura.

3.1 DEFINIÇÃO

Drupal (VANDYK & WESRGATE, 2007) (SANTOS, 2007) (drupal.org) é uma ferramenta desenvolvida justamente para se encaixar principalmente em duas dessas classificações: Sistemas Colaborativos de Gerenciamento de Conteúdo e Portais e Comunidades *on-line*, mas também pode ser aplicado nas outras classificações sem problemas. Sendo o Drupal um sistema colaborativo de código aberto criado justamente para colaboração entre usuários. Esta ferramenta mantida e desenvolvida por uma comunidade ativa pode ser facilmente adquirida por meio do próprio *site* (www.drupal.org). Além de um sistema gerenciador de conteúdo ele também é uma plataforma de desenvolvimento, onde desenvolvedores *Web* utilizam-no para criar sistemas complexos que vai desde comércio eletrônico até produção de livros colaborativos.

3.2 HISTÓRIA

O Drupal teve inicio em 2000, quando, após configurar uma rede sem fio de computadores para compartilhar a conexão do modem ADSL de Hans Snijder entre oito dormitórios de estudantes na universidade da Antuérpia, Dries Buytaert pensou em desenvolver uma ferramenta que permitisse a um grupo de amigos para discutir e partilhar coisas simples. Dries começou então a trabalhar em um *site* de noticias com um quadro de avisos, onde os amigos podiam deixar notas sobre o estado de rede, anunciar onde iriam jantar ou comunicar coisas sem importância.

Quando Dries concluiu seu curso de graduação, o grupo decidiu publicar o *site* na Internet para que pudessem continuar mantendo contato, compartilhando seus achados e narrando pedaços de sua vida pessoal. Enquanto procuravam por um nome para o *software*, Dries registrou o domínio “drop.org”.

Uma vez na *Web*, o drop.org tornou-se lentamente um ambiente de experimentação pessoal, onde fluíam novas idéias. Os usuários do *site* começaram a conversar sobre novas tecnologias para *Web*. As idéias que surgiram foram implantadas e testadas no drop.org, tornando-se, em seguidas adições do *software*.

Em janeiro de 2001, Dries decidiu lançar o *software* no drop.org com o nome de Drupal. Após o lançamento do Drupal como condigo aberto, o projeto e a comunidade tiveram um enorme crescimento. Em julho de 2006, em uma viagem a São Francisco, Califórnia, Dries e a comunidade de desenvolvedores do Drupal decidiram criar um organismo centralizador, que cuidava da hospedagem de infra-estrutura da promoção e comercialização, e de outras questões periféricas projeto.

Em setembro do mesmo ano, em Bruxelas Dries anunciou a criação da associação Drupal, uma sociedade sem fins lucrativos que tem por objetivo dar apoio ao projeto Drupal.

3.3 AMBIENTE DO DRUPAL

O Drupal é uma ferramenta para a criação de sites na *Web*, constituindo-se num sistema de código livre para gerenciamento de conteúdo, ditado de uma estrutura altamente modular e direcionado para a colaboração de usuários. Abaixo uma lista das tecnologias que são pré-requisitos no servidor onde estamos executando o Drupal.

- PHP, o Drupal e escrito na linguagem PHP, que adquiriu má fama porque sendo fácil de aprender, permitiu a iniciantes escrever grande quantidade de códigos. Todavia é também uma linguagem de programação orientada a objetos e pode ser usado para escrever códigos sólidos.

- Apache, devido a sua longa historia com servidor Apache, o Drupal opera com arquivos .htaccess em sua raiz, o que dá segurança a instalação. É o servidor *Web* mais popular na Internet, existem milhões de *sites* utilizando Apache.

- MySQL, a interface com o nível de banco de dados no Drupal, é feita através de uma camada leve de abstrações de banco de dados. Essa camada faz o saneamento das questões SQL e torna possível a utilização de bancos de dados diferentes sem precisar refazer o código.

Com a exceção da linguagem PHP, o Drupal possui outras opções no ambiente, como mostra a figura abaixo.

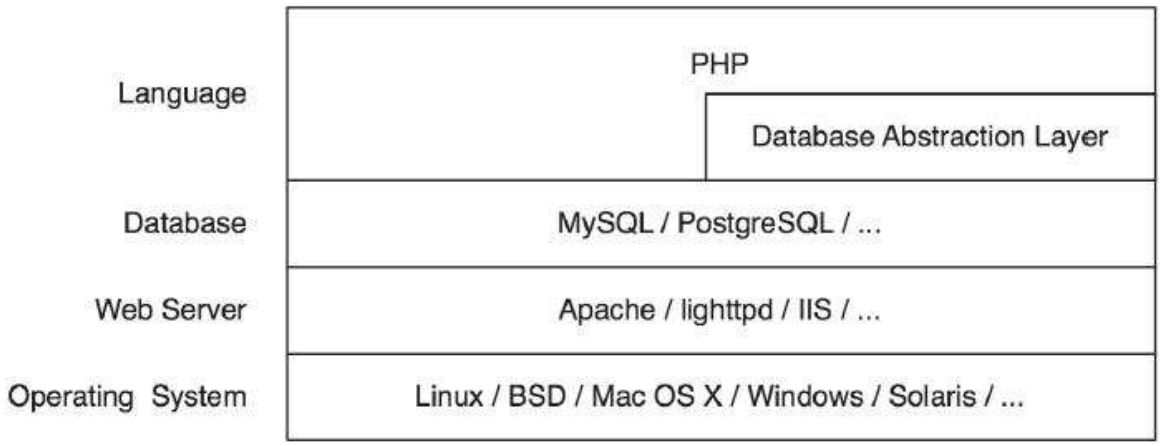


Figura 8 - Camadas do ambiente Drupal - Fonte: VANDYK, 2007, p.2

3.3.1 ESTRUTURA DE DIRETÓRIO

A compreensão e uma estrutura de diretório de uma instalação default do Drupal o ajudará a depurar o seu *site* e lhe ensinará práticas importantes, como o lugar como os módulos e Themes baixados devem residir e como o obter diferentes perfis do Drupal. Uma instalação default do Drupal tem a estrutura mostrada na figura 9.

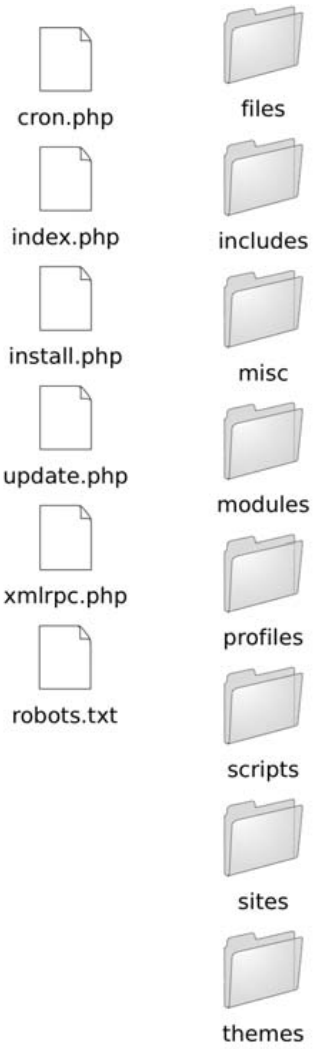


Figura 9 - Diretório padrão da estrutura da instalação do Drupal – Fonte: VANDYK, 2007.

O diretório *files* é utilizado para *uploads* de arquivos, não vem por padrão na instalação do Drupal quando criado deve-se conceder permissão de escrita.

O diretório *includes* contém as bibliotecas de funções de uso comum no sistema.

A pasta *misc* contém diversos JavaScripts, imagens e ícones disponíveis.

No diretório *modules* contém os módulos do core do Drupal, cada módulo esta separado em um subdiretório, os módulos extras inseridos no Drupal posteriormente devem ser armazenados no diretório *sites/all/modules* do *site* específico.

Na pasta *profiles* contém diferentes perfis de instalação de um *site*. O principal objetivo de uma instalação de perfil é ativar módulos automaticamente. Por meio do perfil pode-se configurar uma instalação personalizada do Drupal.

O diretório *scripts* possui *scripts* para verificação da sintaxe, limpeza de código por meio do *cron*[6].

O diretório *sites* contém as personalizações do usuário para o Drupal com relação a novos mudulos e temas. Quando se adiciona módulos a instalação do Drupal esses devem ser inseridos em *sites/all/modules*. Isto evita a mistura dos módulos do *core* do drupal, com módulos adicionados posteriormente. O Drupal permite vários sites rodando na mesma instalação, para isso é necessário somente copiar e alterar o nome do diretório default para o domínio do *site* desejado e configurar o arquivo settings.php. A Figura 10 representa a estruturação do diretório *sites*.

[6] ***Cron*** é um agendador de tarefas. Com ele é possível agendarmos rotinas ou tarefas, de modo que o sistema execute-as periodicamente.

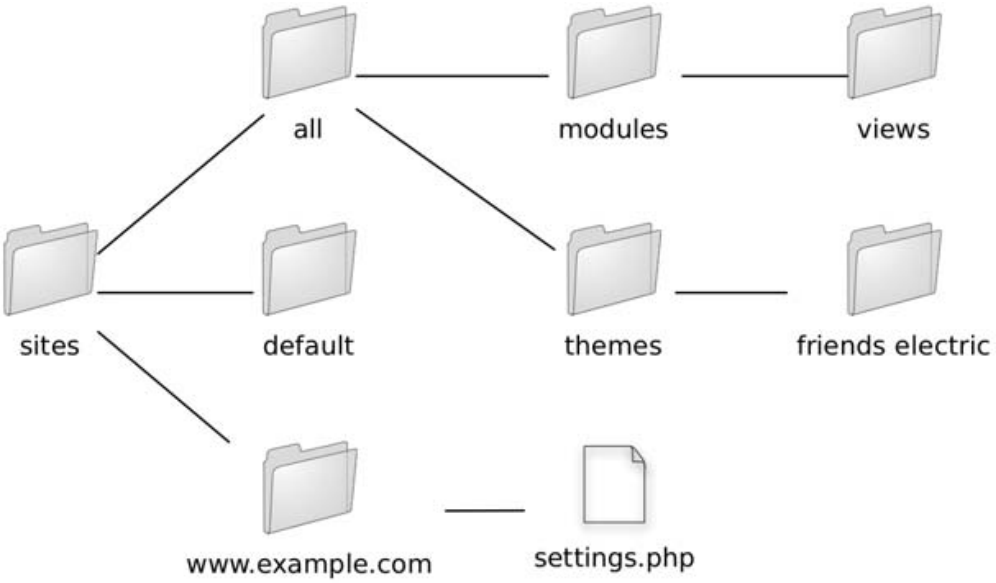


Figura 10 - O diretório *sites* armazena todas as modificações feitas no Drupal – Fonte: VANDYK, 2007.

3.3.2 RECURSOS PADRÕES

O núcleo do Drupal se constitui de uma estrutura base que é encontrada ao se baixar o Drupal do “drupal.org”. O núcleo é responsável pela previsão de funcionalidades básicas que será utilizada para dar suporte a outras partes do sistema.

O núcleo inclui um código que permite ao sistema Drupal responder por si mesmo quando recebe uma requisição, uma biblioteca de funções comuns é usada freqüentemente com o Drupal, e módulos que provem funcionalidades básicas, como gerenciamento do usuário, taxonomia, e template, conforme se vê na figura 11.

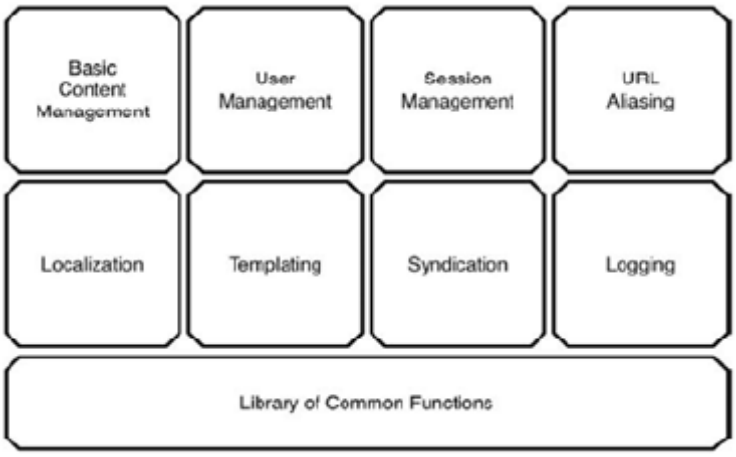


Figura 11 - Visão geral de parte do núcleo do Drupal – Fonte: VANDYK, 2007.

A interface administrativa do Drupal é estreitamente integrada ao resto do site e, por default, usa o mesmo tema do *site*. O primeiro usuário, o "usuário 1", é o super usuário com acesso total ao site.

O Drupal é verdadeiramente uma estrutura modular. Pode-se adicionar funcionalidade ao um site Drupal pela habilitação de módulos existentes ou pela instalação de módulos escritos por membros da comunidade Drupal ou ainda pela criação de módulos próprios.

3.4 CONCEITOS E CARACTERÍSTICAS DO DRUPAL

3.4.1 MÓDULOS

Em muitas aplicações de código aberto, é possível a customizacao pela alteração do código fonte, no Drupal esse método não é recomendado. Em lugar disso o Drupal é projetado desde a base para ser modular e extensível. Módulos são *plugins* para o Drupal. O sistema Drupal consiste em uma estrutura bastante enxuta para a criação de aplicativos e a instalação default é considerado o núcleo (*core*) do Drupal. No core do Drupal existem os módulos obrigatórios, que não podem ser desabilitados, e módulos opcionais que adicionam funcionalidade ao núcleo pela sua habilitação. Os módulos nada mais são do que arquivos contendo o código PHP e quando adicionados pelo usuário residem no subdiretório sites/all/modules de sua instalação Drupal.

Desenvolvidos e mantidos por membros da comunidade Drupal e até mesmo por empresas que ao criarem novos projetos e desenvolverem módulos contribuem com a comunidade disponibilizando o código para outras pessoas, esses módulos são encontrados no site oficial do Drupal, todos os módulos que não fazem parte de qualquer lançamento oficial pode não ser otimizados ou podem até mesmo não funcionar corretamente. Os módulos são desenvolvidos para versões específicas, um módulo da versão 5.x não ira funcionar na versão 6.x.

A Figura 12 mostra alguns módulos que compõem o sistema, tanto módulos do núcleo do Drupal quanto os módulos adicionais.

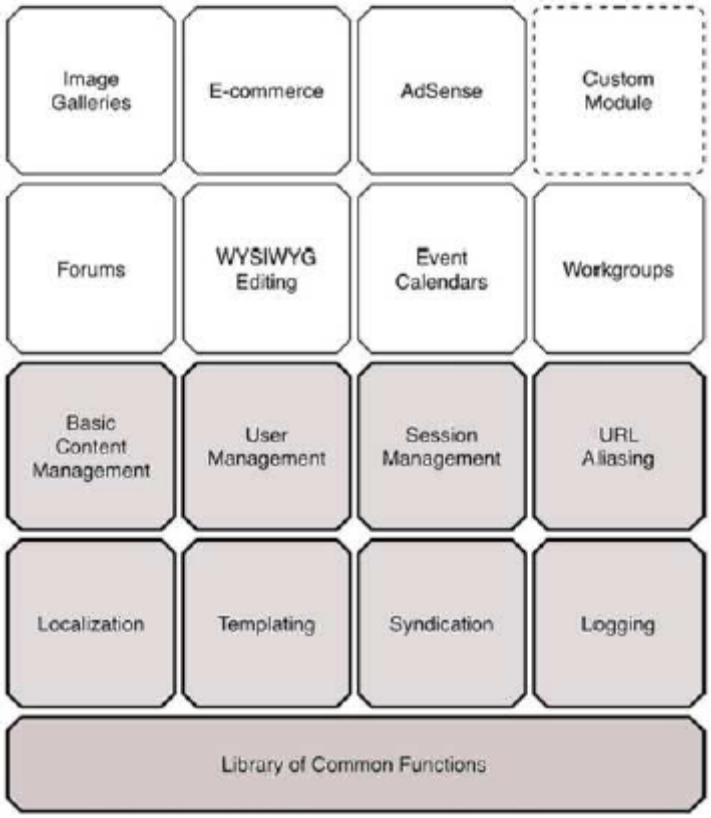


Figura 12 - Ativando módulos adicionais para aumentar as funcionalidades. – Fonte: VANDYK, 2007.

3.4.2 HOOKS

Os *hooks* (ganchos) podem ser entendidos como eventos internos do Drupal. Eles são também chamados de '*callbacks*' (recuperação, resgate). Os *hooks* permitem aos módulos se inserirem

no que acontece no resto do Drupla. Suponha que um usuário entre (faça *login*) no seu site Drupal. Quando ele entra, o Drupal aciona o hooks do usuário, chamando funções apropriadas a ele.

O Drupal utiliza o padrão de design inversão de controle, no qual a funcionalidade modular é chamada pela estrutura no tempo apropriado. Essas oportunidades para os módulos exercerem sua função são chamadas de *hooks*.

Durante o curso das execuções no sistema, o Drupal pergunta aos módulos se eles gostariam de fazer alguma coisa. Por exemplo, quando o Drupal quer determinar qual módulo é responsável pela requisição atual, ele pergunta para todos os módulos ate encontrar o módulo certo que conterá o hook que por sua vez irá mostrar o caminho para o tratamento dessa requisição.

Quando acessamos uma página como a página de profile do usuário, é chamado o *hook_user* que invoca todos os módulos que implementam o *hook_user* por meio da nomenclatura nomedomodulo_user(). Utilizamos assim o *hook user* para adicionar conteúdo na página *profile* do usuário, porém podemos utilizar o *hook user* para diversas outras situações.

3.4.3 SISTEMA DE MENU

O termo sistema de *menu* é de certa forma um nome equivocado. É mais apropriado se pensar no sistema de menu como como tendo três responsabilidades primarias: mapeamento de chamadas (callbacks), controle de acesso e customização de *menu*.

Menu callback

Quando um navegador faz uma solicitação para Drupal, dá ao Drupal uma URL. A partir desta informação, o Drupal deve descobrir como lidar com a requisição e qual o código a ser executado. Isto é comumente conhecido como “*dispatching*”. O Drupal analisa apenas o *path* da URL e ignora o *host*. Por exemplo:

URL = http://exemplo.com/?q=node/3
PATH = node/3

Por meio do path o Drupal encontra o item de *menu* responsável, no *menu* haverá configurações que apontam para uma função *callback*.

```
function mymenu_menu($may_cache) {
        // Create an array to hold the menu items we'll define.
        $items = array();
        if ($may_cache) {
                // Define a static menu item.
                $items['mymenu/'] = array(
                        'title' => t('Greeting'),
                        'page callback' => 'mymenu_hello'
                );
        }

        return $items;
}

function mymenu_hello() {
        return t('Hello!');
}
```

O código acima possui uma implementação do *hook_menu*, que cria uma URL chamada "/mymenu" que por sua vez aponta para uma função *callback* chamada mymenu_hello.

Neste exemplo, quando o usuário digitar site.com/mymenu o Drupal irá executar a função mymenu_hello que irá imprimir na tela "Hello!" como mostra a figura 13. O parâmetro *access => true* indica que todos os usuários terão acesso ao *menu*.

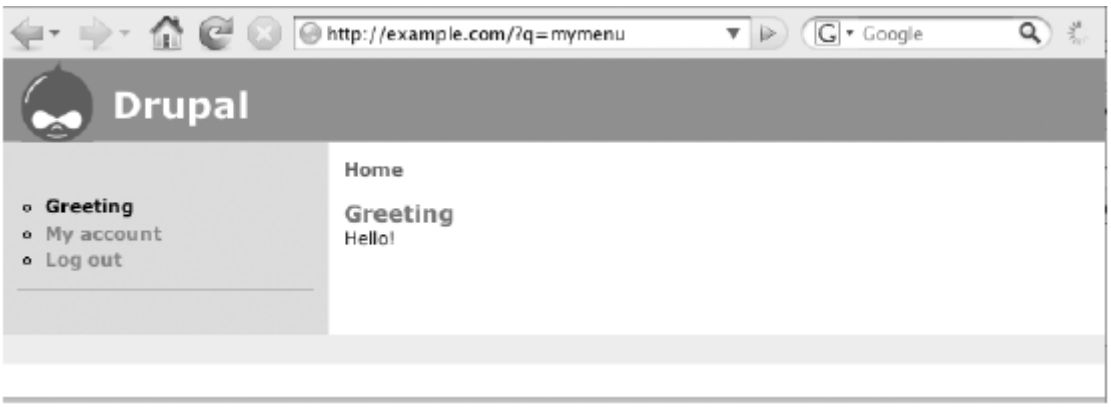


Figura 13 - O item de *menu* (*Greeting*) aparece no bloco de navegação – Fonte: VANDYK, 2007.

Podemos adicionar o controle de acesso a um endereço atribuindo a permissão que queremos verificar ao acessar a URL no parâmetro *access* do *menu*. Para isso devemos adicionar em $items['mymenu/'] do codigo acima a linha a seguir:

'access arguments' => user_access(‘receive greeting’) // Returns TRUE or FALSE.

Com a adição dessa linha no código é verificado se o usuário que está acessando a URL *mymenu* possui a permissão “*receive greeting*”, caso não possua será apresentada uma mensagem dizendo que o usuário não possui permissão, caso contrário executa a função *mymenu_hello*.

3.4.4 CAMADA DE ABSTRAÇÃO DE BANCO DE DADOS

No Drupal existe uma camada leve de abstração com seu banco de dados. Por padrão o Drupal possui suporte a MySQL e PostgreSQL, sendo mais indicado o primeiro, porém é possível estender a classe de abstração e escrever os métodos necessários para utilizar qualquer outro banco de dados.

A conexão com o banco de dados é configurada no arquivo sites/default/settings.php, ou em sites/seusite/settings.php quando existir mais de um *site* configurado em uma mesma instalação do Drupal, e sua sintaxe é:

$db_url = 'mysql://usuario:senha@localhost/databasename';

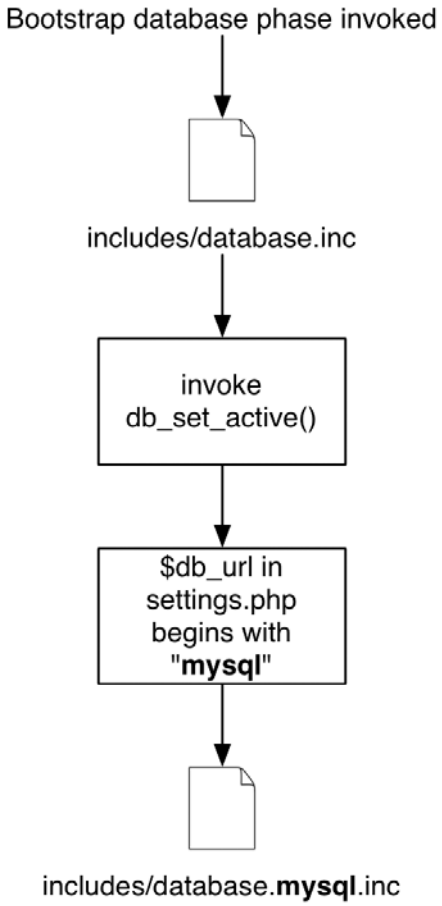
No caso de um banco PostgreSQL deve-se alterar o início de mysql para pgsql.

Trabalhar com uma API[7] de abstração de base de dados é algo que não é apreciado até que quando é utilizada pela primeira vez, não se consegue mais viver sem. Quando é preciso fazer uma mudança de base de dados em um sistema acaba sendo um transtorno, pois é necessária a

[7] **API**, *Application Programming Interface* (ou Interface de Programação de Aplicativos), é um conjunto de rotinas e padrões estabelecidos por um software para utilização de suas funcionalidades por programas aplicativos -- isto é: programas que não querem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.

mudança em todas as *querys* do sistema. Utilizando-se de uma camada de abstração, ao invés de chamar *mysql_query* () ou *pg_query* () por exemplo, o usa Drupal *db_query* (), que mantém a lógica para tratamento devido das *querys* para uma base de dados especifica. A camada de abstração de dados do drupal é leve e tem dois objetivos principais. O primeiro é manter o seu código desvinculado a uma base de dados especifica. O segundo é o de filtrar dados de usuário colocados em consultas para evitar SQL *injections*[8].

O Drupal determina o tipo do banco de dados a ser conectado por meio da análise da variável $db_url. Se a variável iniciar com *mysql*, por exemplo, então o Drupal faz o include do arquivo *includes/database.mysql.inc*. Se iniciar com *pgsql* o arquivo incluído é o *includes/database.pgsql.inc*. Esse mecanismo é mostrado na Figura 14.



[8] **SQL *Injection***, é uma técnica que explora a vulnerabilidade da segurança na camada de banco de dados de uma aplicação. Isso ocorre quando o usuário entra com comandos SQL e a aplicação não faz a verificação dos dados informados aplicando filtros.

Figura 14 - O Drupal determina qual arquivo de *database* será incluso analisando a variável $db_url – Fonte: VANDYK, 2007.

É criado para cada banco de dados um arquivo específico no diretório includes com o nome de database.nome.inc. Esse arquivo sobrescreve as funções da API com funções específicas do banco de dados. Possibilita-se assim trabalhar com qualquer banco de dados, desde que utilizem o padrão SQL ANSI[9].

```
// From database.mysqli.inc.
function db_fetch_object($result) {
        if ($result) {
                return mysql_fetch_object($result);
        }
}

// From database.pgsql.inc.
function db_fetch_object($result) {
        if ($result) {
                return pg_fetch_object($result);
        }
}
```

O código acima compara as diferenças entre as funções db_fecth_object() para o MySQL e o PostgreSQL.

O comando db_query () do Drupal é utilizado para executar uma consulta à base de dados por meio de uma conexão ativa. Estes comandos incluem consultas como SELECT, INSERT, UPDATE e DELETE. Alguns exemplos serão demonstrados a seguir.

[9] **SQL ANSI**, é uma linguagem de consulta estruturada utilizada na comunicação com banco de dados, este padrão foi estabeleceu pela ANSI ou American National Standards Institute .

Selecionando todos os campos de todas as linhas da tabela *joke* onde *vid* tem o valor de $node->vid.

db_query('SELECT * FROM {joke} WHERE vid = %d', $node->vid);

Inserindo aos campos nid, vid e *punchline*, os valores de $node->nid, $node->vid e $node->punchline respectivamente. Sendo que '%d' são inteiros e '%s' são *strings*.

db_query("INSERT INTO {joke} (nid, vid, punchline) VALUES (%d, %d, '%s')",

$node->nid, $node->vid, $node->punchline);

Atualizando na tabela *joke* o valor do campo *punchline* nas linhas onde o valor de vid é igual a $node->vid.

db_query("UPDATE {joke} SET punchline = '%s' WHERE vid = %d", $node->punchline, $node->vid);
Apagando da tabela *joke* todas as linhas onde vid é igual a $node->nid.
db_query('DELETE FROM {joke} WHERE nid = %d', $node->nid);

No Drupal, as *querys* são sempre escritas com *placeholders* que são as variáveis precedidas de % nos exemplos anteriores, elas são automaticamente substituídas com os valores dos parâmetros passados no final da query, como $node->vid mostrado no primeiro exemplo. Por meio dos *placeholders* também feita a análise para saber qual o tipo de informação que esta sendo passado e por meio do tipo da variável é feita a validação aplicando filtros. Na Figura 15 é mostrada a lista de *placeholders* disponíveis.

Placeholder	Meaning
%s	String
%d	Integer
%f	Float
%b	Binary data; do not enclose in ' '
%%	Inserts a literal % sign (e.g., `SELECT * FROM {users} WHERE name LIKE '%%%s%%'`)

Figura 15 - Database Query Placeholders and Their Meaning - Fonte: VANDYK, 2006.

A API da camada de abstração também permite recursos como à conexão as múltiplas bases de dados, paginação de resultados e outras facilidades.

3.4.5 PERSONALIZAÇÃO

Na criação de uma página para ser enviada para um browser existem duas preocupações principais: Reunir os dados apropriados e formatar os dados para a *Web*. No Drupal a camada *theme* é responsável pela criação do HTML que o browser irá receber. É importante lembrar que o Drupal incentiva a separação entre o conteúdo e a sua apresentação, permitindo diversas formas de customização e melhoria de aparência do *site*. A forma mais simples é a utilização de CSS para incrementar a utilização de espaços embutidos das classes IDs no Drupal. Mas é possível ir mais além, customizando o produto real do HTML, o que pode ser feito com facilidade utilizando-se os arquivos de *template* ou declarando funções com nomes apropriados nas páginas.

3.4.6 SISTEMA DE PERMISSÃO BASEADO EM REGRAS

O sistema de permissão como função responsável por definir regras de acesso o hook_perm(). Quando definido um hook_perm() dentro de um módulo do Drupal, o sistema gera automaticamente no gerenciador de permissão uma entrada onde atribui-se ou não a permissão para determinado papel de usuário. As permissões são atribuídas aos papéis que os usuários desempenham no sistema, e um mesmo usuário pode desempenhar mais de um papel.

```
/**
* Implementation of hook_perm().
*/
function mymenu_perm() {
        return array('receive greeting');
}
/**
* Implementation of hook_menu().
*/
function mymenu_menu($may_cache) {
        $items = array();
```

```
    if ($may_cache) {
        $items[mymenu/] = array(
        'title' => t('Greeting'),
        'callback' => 'mymenu_hello',
        'access' => user_access('receive greeting') // Returns TRUE or FALSE.
        );
    }
    return $items;
}
```

No código acima é definida uma permissão chamada ‘*receive greeting’* caso um usuário sem essa permissão tente acessar a URL http://site.com/?q=mymenu, esse usuário irá receber uma mensagem de “acesso negado”. Dessa forma o sistema de *menu* age de acordo com as permissões dos usuários, e define quais os caminho que esse usuário tem permissão e quais ele não tem permissão (VANDYK & WESRGATE, 2007) (SANTOS, 2007) (drupal.org).

4. REQUISITOS FUNCIONAIS DO SISTEMA

O ambiente de colaboração educacional baseado em CMS será tratado deste ponto em diante pelo nome Espaço Conhecimento. Esta nomenclatura foi assumida por tratar-se de um ambiente virtual utilizado para conectar pessoas e idéias.

O Espaço Conhecimento, de forma colateral, irá congregar funções atualmente exercidas pela página do curso, sistema de gestão acadêmica, gestão de conteúdos e ferramenta colaborativa. O funcionamento do sistema será baseado em torno de dois conceitos: comunidades e contatos.

A metodologia utilizada para levantamento de requisitos incluiu as técnicas de observação, entrevistas, questionários e reuniões com alunos e professores da instituição de ensino.

4.1 FUNCIONALIDADES

As funcionalidades do sistema surgiram com os resultados obtidos da fase de levantamento de requisitos. Serão citadas a seguir as funcionalidades com detalhes característicos para atender as necessidades da instituição pública de ensino.

4.1.1 COMUNIDADES

A característica principal do Espaço Conhecimento a formação de comunidades, construídas a partir de disciplinas, temas de interesse, grupos de trabalho, ou redes de relacionamento quaisquer. Por exemplo:

Professores

Alunos 2006

Introdução a Computação

Estrutura de Dados

Engenharia de *Software*

Dentro de uma comunidade, participantes podem realizar debates, trocar documentos, agendar atividades. Todos esses eventos ficam registrados, formando um histórico dos eventos, tornando possível entender a evolução do grupo de trabalho.

O acesso a comunidade depende da permissão configurada. Um participante com permissão para modificar esse acesso é chamado gestor. Quando o usuário cria uma nova comunidade, ele é por padrão um gestor. No perfil dessa comunidade pode-se indicar outros participantes como gestores, sendo que não há limite para o número de gestores.

Os tipos de acesso são: aberto, restrito ou fechado:

Aberto: A Comunidade com acesso aberto permite qualquer usuário visualize o conteúdo e se inscreva no grupo.

Restrito: A comunidade com acesso restrito permite que qualquer usuário veja o conteúdo, mas necessita aprovação de um gestor para inscrição.

Fechado: A comunidade com acesso fechado só permite o ingresso, seja para visualização ou cadastramento, se o usuário for convidado.

4.1.2 CONTATOS

O segundo conceito que permeia o Espaço Conhecimento é a noção de contatos. Contatos são pessoas de alguma forma relacionadas com o usuário.

A rede de relacionamento de um determinado indivíduo ajuda a identificar esse sujeito, e da mesma forma, uma pessoa pode qualificar um certo grupo.

Do ponto de vista do usuário ter uma lista de contatos é útil visto que o sistema notifica alterações, como novas mensagens, novos documentos, aniversário etc.

4.1.3 VISUALIZAÇÃO

O usuário não pode ser “invisível” aos outros participantes do Espaço Conhecimento, mas ele pode configurar itens do seu perfil que poderão ser visualizados ou não.

É possível também configurar se a área Recados pode ser visível para outros usuários e se o seu *Blog* pode receber comentários.

4.1.4 ADIÇÃO

Para adicionar uma pessoa à lista de contatos do usuário, aquela deve explicitamente aceita-lo, sendo que quando positivo, um passará a ser contato do outro.

Da mesma forma, quando um contato for removido, automaticamente o usuário sai da lista de contatos dessa pessoa.

4.1.5 COMUNICAÇÃO

Cada usuário do sistema tem duas ferramentas para comunicação: recados e *blog*.

Recados são mensagens deixadas pelas outras pessoas do sistema, sejam elas contatos do usuário ou não. Por padrão, os recados são abertos, isto é, qualquer um pode vê-los, mas isso pode ser alterado conforme descrito no item “Visualização”.

A outra ferramenta, *blog*, existe para que o usuário converse com todos e/ou com ninguém; ele deixa um artigo, um pensamento, uma opinião, e quem se interessar pode ir ler e comentar.

4.2 INTEGRAÇÃO ENTRE SISTEMAS

O Espaço Conhecimento foi planejado para possuir uma integração com os sistemas de gestão acadêmica do PPEGC/UFSC e DEPS/UDESC, possibilitando a comunicação dos cadastros, e com isso evitando redundância dos dados e, conseqüentemente, as falhas provenientes dessas inconsistências.

Do ponto de vista do PPEGC/UFSC, o ambiente foi planejado para conversar com a plataforma de gestão atualmente utilizada no curso para realização de matrículas, cancelamentos e pedidos de declarações. Essa conexão cria mais uma área no ambiente, chamada Secretaria.

Já na DEPS/UDESC, planeja-se na integração a possibilidade de unificação de cadastrados com o sistema atual, permitindo que se utilize o mesmo *login* e senha para entrar no ambiente.

Em ambos os casos, a intenção é que o usuário importe dados da plataforma *Lattes* (CNPq) para o seu perfil.

Estrutura do Sistema

O sistema foi estruturado para ser formado por duas espaços principais: o Espaço Conhecimento acessado principalmente por alunos e professores e o Ambiente Administrativo acessado pelos responsáveis pelo gerenciamento do Espaço Conhecimento, conforme mostra a Figura 16.

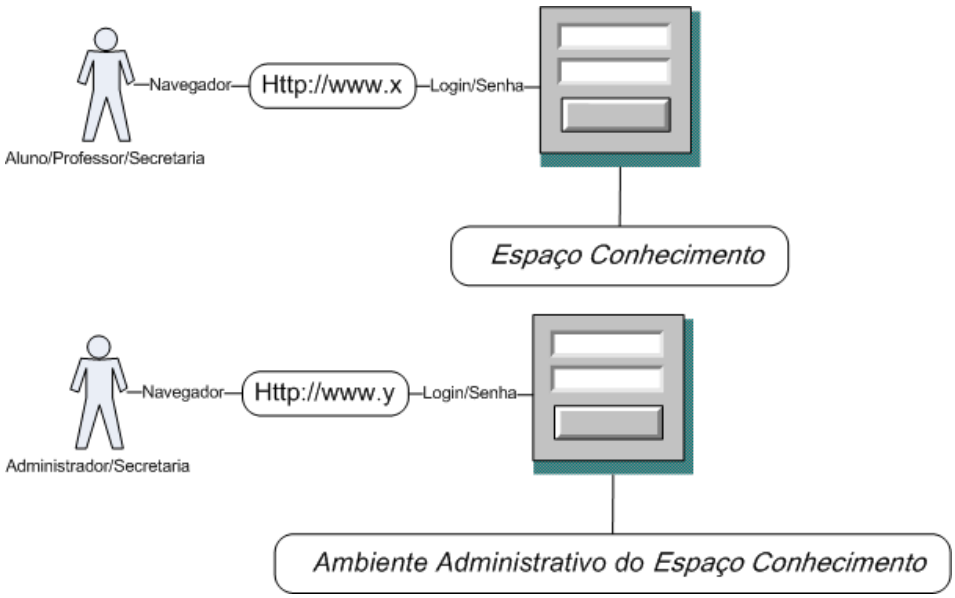


Figura 16 - Duas estruturas do Espaço Conhecimento.

O Espaço Conhecimento é dividido em sete áreas: Meu Espaço, Procurar Contato (listagem de contatos), Procurar Comunidade (listagem de comunidades), Contato (área de um contato específico), Comunidade (área de uma comunidade específica), Ajuda, Monitor de Eventos.

4.3 ÁREAS DO ESPAÇO CONHECIMENTO

4.3.1 MEU ESPAÇO

Dividida em:

Perfil: Traz informações concisas e informais sobre o usuário, últimos eventos, últimos contatos que acessaram o sistema. Permite que o usuário configure seu ambiente.

Contatos: Mostra a lista de contatos do usuário. A lista leva ao perfil dessas pessoas.

Comunidades: Mostra a lista de comunidades do usuário. A lista leva ao perfil dessas comunidades.

Recados: Todas as mensagens recebidas pelo usuário. É possível responder, sendo a resposta enviada para Recados dessa pessoa, ou apagá-la.

Blog: Permite que usuário gerencie suas postagens e as respostas às mesmas.

Documentos: Reúnem todos os arquivos do usuário. Documentos devem ser acompanhados de rótulos, que recebem o nome de coletâneas. Um mesmo documento pode pertencer a várias coletâneas.

Secretaria: É o espaço onde o usuário pode fazer matrículas e cancelamento de disciplinas, solicitar declarações.

4.3.2 MECANISMOS DE PROCURA

Visualizando o crescimento do número de contatos e comunidades no sistema, foram projetados os mecanismos de:

Procura por Contato: Lista os contatos registrados no sistema. São apresentados em grupo de 10 em 10 e é possível fazer filtragem segundo uma série de critérios: nome, área de concentração, nível, sexo.

Procurar por Comunidade: Lista as comunidades registradas no sistema. São apresentados em grupo de 10 em 10 e é possível fazer filtragem segundo uma série de critérios: nome, área de concentração, tipo.

4.3.3 CONTATO

Dividida em:

Perfil: Traz informações concisas e informais sobre a pessoa, documentos, postagens.

Contatos: Mostra a lista de contatos. A lista leva ao perfil dessas pessoas.

Comunidades: Mostra a lista de comunidades. A lista leva ao perfil dessas comunidades.

Recados: Todas as mensagens recebidas. É possível criar uma nova mensagem para pessoa.

Blog: Permite ver as postagens e as responder às mesmas.

Documentos: Reúne todos os arquivos da pessoa.

4.3.4 COMUNIDADE

Dividida em:

Perfil: É uma sintetize da comunidade, composta pelos seus dados, processados e agrupados de forma lógica, como informações (nome, descrição, contatos externos, área de concentração, linha de pesquisa), últimos eventos (quem se associou, quem se desassociou, novos documentos, novas mensagens, outras modificações quaisquer), últimos participantes que acessam a comunidade, agenda resumida da comunidade. O quanto é apresentado depende do nível de acesso à comunidade (comunidades abertas e restritas mostram todos os dados, enquanto comunidades fechadas não mostram nada).

Participantes: Mostra quem faz parte da comunidade. A lista de integrantes leva ao perfil desses usuários.

Debates: Debates é a fusão de duas tradicionais ferramentas de comunicação: o fórum e o *chat*. Baseado no Chorum. Caso um usuário não faça parte da comunidade, se ela for aberta ou restrita é possível ver os debates, mas não é possível postar; se for fechada, não é permitido visualizar as conversar.

Documentos: Reúne todos os arquivos relativos a uma determinada comunidade. Documentos devem ser acompanhados de rótulos, que recebem o nome de coletâneas. Um mesmo documento pode pertencer a várias coletâneas, porém, um documento só pertence a uma comunidade. Os documentos podem, por exemplo, representar aulas ou tópicos de disciplinas, oferecendo suporte para converter estes documentos para o formato SCORM.

Agenda: A agenda é um recurso para permitir que integrantes de uma comunidade saibam de acontecimentos futuros e possam se organizar de acordo. Qualquer integrante pode adicionar itens na agenda, mas apenas gestores podem remover.

Ajuda: Área explicativa sobre o funcionamento do sistema. É contextualizada segundo a navegação do usuário.

Monitor de Eventos: Monitor de eventos é uma área que mostra tudo que aconteceu no sistema pertinente ao usuário.

4.4 FERRAMENTAS

O Espaço Conhecimento é composto por vários módulos, sendo válido destacar três deles, por serem pequenos sistemas com suas complexidades próprias.

4.4.1 CHORUM

O Chorum, utilizado pelos *Blogs* dos usuários e pelos Debates das comunidades, é a fusão de duas tradicionais ferramentas de comunicação: o *chat* e o fórum. Usuários podem criar tópicos, sendo cada um deles uma conversa única. As mensagens mais antigas aparecem no topo da página enquanto as mais novas aparecem na base. Não há encadeamento. Entretanto a conversa é viva, isto é, se houver várias pessoas postando no tópico, a página da discussão vai se atualizando automaticamente. Na base da página há o formulário para inserir novas mensagens. As mensagens aceitam formatação rica de texto, tipo negrito, itálico, sublinhado, colorização, lista numerada e não-numerada, e ainda inserção de *emoticons*, documentos cadastrados, *links* para outros *websites*, *links* para vídeos de *sites* tipo YouTube.

4.4.2 AMBIENTE DE ADMINISTRAÇÃO DO SISTEMA

Um aspecto diferencial do Espaço Conhecimento em relação aos ambientes virtuais de aprendizagem tradicionais é uniformização dos usuários, com os níveis de permissões emergindo naturalmente do uso do sistema, mediante a construção das comunidades.

Na verdade, continua existindo os administradores, entidades externas que podem comandar o uso do sistema. Entretanto, dentro do Espaço Conhecimento, esses administradores são usuários como outros quaisquer. A administração é realizada em um outro ambiente, que permite modificar e cancelar usuários, comunidades, documentos, mensagens, ver *logs* e estatísticas.

4.5 CONVENÇÕES DE NAVEGAÇÃO

Quando acessa a página inicial do ambiente o usuário encontra sua pagina de perfil onde são reunidas as principais informações relacionadas ao seu perfil, seus contatos e últimos eventos ocorridos no ambiente na sua ausência. Na tela inicial o usuário terá disponível um *menu* principal que dá acesso as demais áreas do sistema.

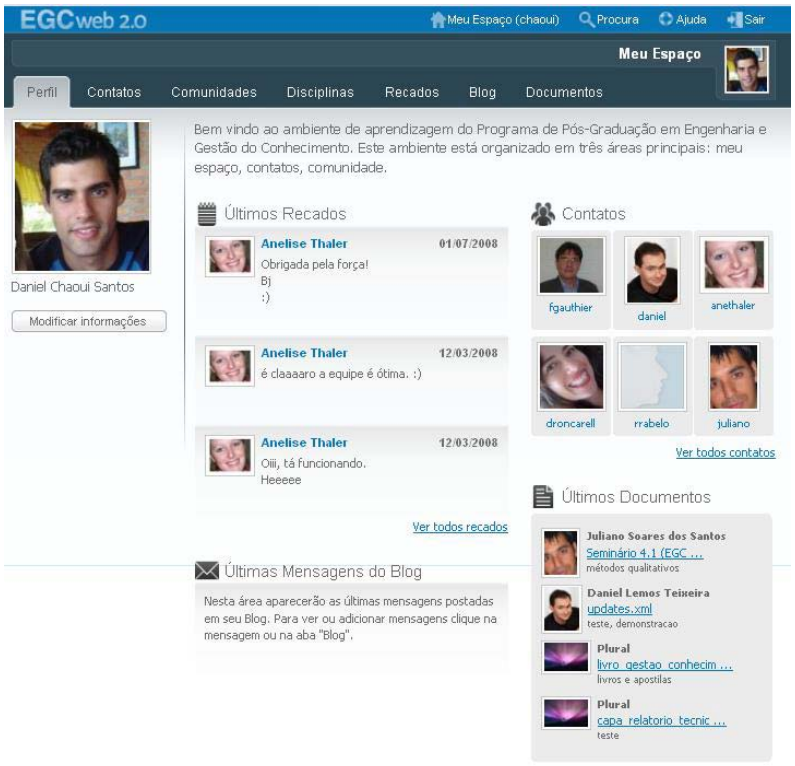


Figura 17 - *Wireframe* mostrando a estruturação do Espaço Conhecimento.

A primeira vista a navegação é inusitada, pois não se constitui em forma de árvore, e sim, literalmente, em rede. Isto é, não parte do início para vários fins, mas do centro para periferia.

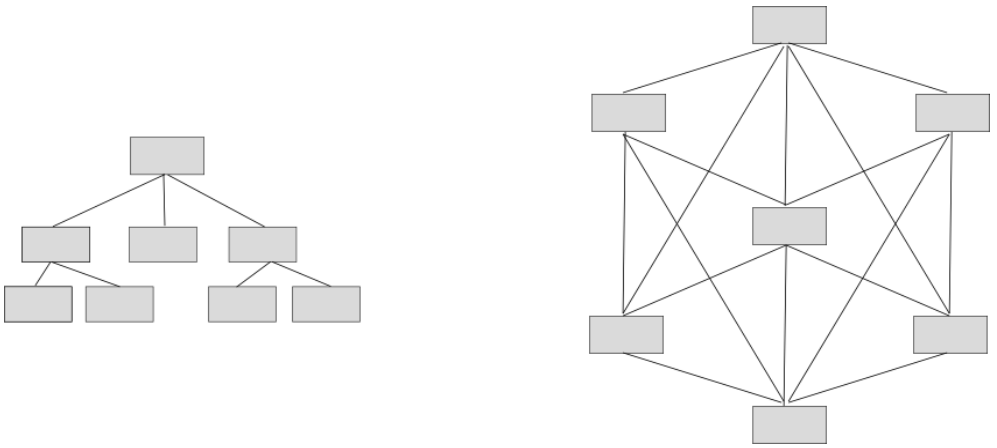


Figura 18 - Fluxo em árvore e fluxo em rede.

4.6 FERRAMENTAS FUTURAS

Algumas ferramentas ficaram fora do projeto, porém já estão planejadas para desenvolvimento futuro, como:

- Miniaturizador

Documentos enviados ao Espaço Conhecimento passam pelo Miniaturizador, sistema cria pequenas imagens que representarão graficamente esse arquivo. O Miniaturizador identifica os seguintes tipos e formatos:

Artigos – odt, doc, docx, rtf, pdf.

Imagens – jpeg, png, gif, bmp, tga.

Vídeos – mpg, mp4, avi, divx, ogg, wmv.

- Toca-Mídias

Além da possibilidade de fazer *download* dos arquivos disponíveis no Espaço Conhecimento, pode-se visualizar certos documentos diretamente no ambiente, graças ao sistema Executor de Mídias. Ele trata os seguintes tipos:

Artigos – odt, doc, docx, rtf, pdf.

Imagens – jpeg, png, gif, bmp, tga.

Áudio – mp3, wma, ogg, wav.

- Conexão com Email

Permite que mensagens recebidas no ambiente sejam redirecionadas ao email do usuário (do ambiente para pessoa), e possibilita que mensagens possam ser enviadas por *email* (da pessoa para o ambiente). Com esta ferramenta uma comunidade pode ser tornar uma lista de *email* de fato. Os usuários não necessitam entrar no ambiente para ver o que está acontecendo como mostra a Figura 19.

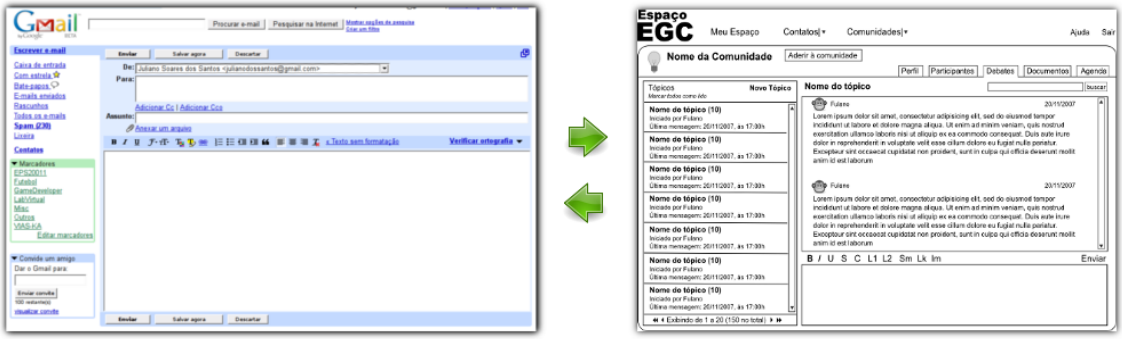


Figura 19 - Conexão com *email* padrão.

- Aplicativo *Desktop* para Transferência de Arquivos

Trata-se de um *software* de *Desktop* padrão que possibilita, como mostra a Figura 20, fazer *upload* ou *download* de documentos no sistema “arrastando” o arquivo entre janelas.

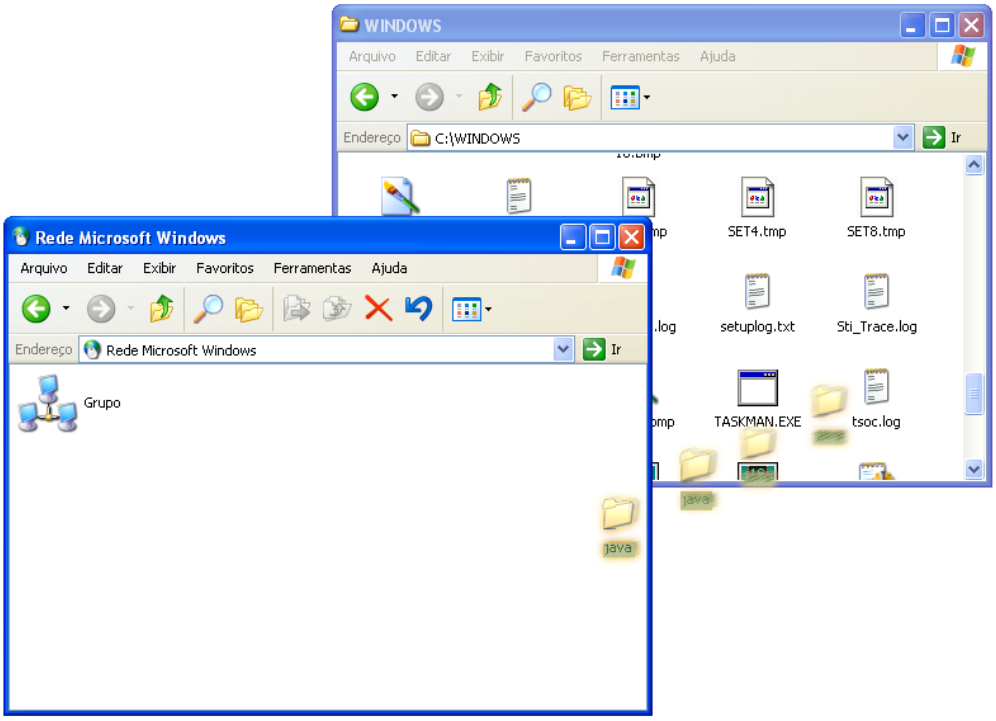


Figura 20 - Esquematização do funcionamento da transferência de arquivos.

- Vídeo-Aula

Trata-se da comunicação do ambiente virtual de aprendizagem com estúdios de videoconferência da UFSC e UDESC, possibilitando que as aulas gravadas com estes equipamentos fiquem disponíveis nos Documentos da comunidade correspondente.

5. ARQUITETURA PROPOSTA

Como principal foco do projeto foi observado à necessidade da criação de uma rede social. Uma rede social é uma estrutura social, comunidade, ou sociedade que correspondem geralmente a perfis de indivíduos ou organizações. Isto indica o modo como eles estão conectados por meio de várias familiaridades sociais, afiliações, e/ou relacionamentos que variam desde conhecidos casuais a ligações familiares próximas.

Por ter sido voltado exclusivamente para universidades publicas uma serie de dados foram levantados para a elaboração dos requisitos do sistema. A metodologia utilizada na coleta de

dados incluiu as técnicas de observação, entrevistas, questionários e reuniões com os ‘*stackholders’* que estão no anexo deste trabalho.

A seguir apresentaremos uma visão organizada da tecnologia que compõe o ambiente virtual de aprendizagem. Seu desenvolvimento é baseado exclusivamente em tecnologias de “*software* livre”, utilizando-se o CMS Drupal 6.2 (Sistema de Gerenciamento de Conteúdo / Content Management System), o banco de dados MySQL 5.1 e o servidor *Web* Apache 2.

Por meio de uma da interface *Web* do Drupal o desenvolvedor pode ativar ou desativar cada módulo, pois os módulos possuem relações de baixo acoplamento entre eles. O baixo acoplamento entre os módulos além de facilitar na personalização do ambiente para diferentes instituições (no caso de instituições que não desejam determinados módulos em sua estrutura) torna-o ainda mais seguro. No caso de uma possível brecha de segurança em alguns dos módulos do sistema, esse pode ser desabilitado facilmente até que se resolva determinado problema.

No caso específico do ambiente virtual de aprendizagem, vários módulos foram criados pela equipe do projeto, sendo denominados “Módulos Próprios”.

5.1 MÓDULOS DO NÚCLEO (*CORE*)

Com a instalação do Drupal, os módulos denominados núcleo (*core*) são automaticamente disponibilizados. No núcleo existem os módulos obrigatórios (*core-required*) e os módulos opcionais (*core-optional*). Somente os módulos do núcleo obrigatório (*core-required*) são requeridos para o sistema funcionar e não podem ser desativados. Apesar de inúmeros núcleos opcionais disponibilizados pelo Drupal, serão utilizados para o desenvolvimento desta plataforma somente dois, o *Locale* e o *Blog* que serão citados a seguir.

5.1.1 MÓDULOS DO NÚCLEO OBRIGATÓRIO (*CORE-REQUIRED*)

5.1.1.1 *FILTER*

Responsável por processar filtros em conteúdos antes de sua liberação e exibição em páginas HTML. Permite configurar formatos de texto para seu *site*. Por exemplo, você pode querer filtros para eliminar códigos HTML maliciosos em comentários de usuários.

Apesar do nome "*filter*", o módulo não só permite-lhe manter fora texto não desejado, mas também permite melhorar o texto que entra. Por exemplo, pode-se usar um filtro para transformar as usuais quebras de linha em *tags* de parágrafos HTML.

5.1.1.2 *NODE*

Responsável por encaminhar determinados conteúdos para exibição em suas páginas HTML específicas. Todo o conteúdo armazenado em um *site* Drupal é tratado como *node*. Um *node* é qualquer postagem, como uma página, pesquisa, história, fórum, ou *blog*. Tratar todos os conteúdos como *node* permite a flexibilidade de criar novos tipos de conteúdo. Ele também permite aplicar novas funcionalidades ou alterações a todo o conteúdo com menos esforço.

Discretamente o modulo *node* gerencia todos os *nodes* do sistema. Este módulo é o que permite que:

- Listar, por meio do tipo, e gerenciar todo o conteúdo do *site*;
- Definir normas para o modo como todas as postagens são exibidas, e,
- Configurar e criar novos "tipos de conteúdo" para o *site*.

5.1.1.3 *SYSTEM*: *CRON* E *CACHE*

Responsável por gerenciar as configurações gerais do sistema, sendo destinado aos administradores do ambiente. O módulo *System* fornece padrões do sistema para serem executados automaticamente em determinados períodos, o armazenamento (*caching*) de páginas da *Web* para melhorar a eficiência, alem de executar outras tarefas essenciais. O módulo também mantém um registro de como o administrador quer que o sistema se comporte.

Alguns dos módulos do Drupal exigem ações regulares. O modulo *statistics* periodicamente limpa seus arquivos de *log*. O módulo *aggregator* atualiza periodicamente seus *feeds*. *Ping* notifica periodicamente serviços de novos conteúdos em seu *site*. Todos estes serviços dependem do cron.

O *Cron* não é uma parte do Drupal. É um agendador de tarefas que reside no servidor e executa tarefas (chamado de "*cron* jobs"), com intervalos que podem ser especificados. Os trabalhos podem ser executados semanalmente, diariamente, por hora, ou o que você quiser.

5.1.1.4 *BLOCK*

É responsável por controlar quais blocos são exibidos em volta do conteúdo principal das páginas. Blocos são as caixas que agrupam funcionalidades do sistema, podem ser visíveis no lado esquerdo, direito, roda pé, cabeçalho e ainda na área de conteúdo do seu *Web site*. Os blocos são geralmente gerados por módulos, mas os administradores também podem definir os blocos.

5.1.1.5 *USER*

Responsável por gerenciar tanto o registro de usuários, quanto o sistema de autenticação. O módulo *user* permite cadastro, *login* e *logout* de usuários. Administradores podem associar conteúdos a diferentes perfis de usuários por meio de permissões de acesso.

Cada usuário pode ser atribuído a um ou mais perfis de acesso. Por padrão, existem dois perfis: anonymous (usuário não logado) e authenticated (usuário logado e autorizado). Porem pode ser criado inúmeros perfis de acesso no sistema.

5.1.2 MÓDULOS DO NÚCLEO OPCIONAL (*CORE-OPTIONAL*)

5.1.2.1 *LOCALE*

O módulo *Locale* permite apresentar o *Web site* desenvolvido com Drupal em um idioma diferente do Inglês, definido como padrão. Ele pode ser usado para criar um ambiente

multilinguagem ou facilitar a tradução para outro idioma, de textos criados especificamente para o *site*.

5.1.2.2 *BLOG*

Este módulo permite criar e atualizar facilmente uma página pessoal ou um "*blog*". Apesar de ser um módulo é opcional, sua habilitação torna-se obrigatória devido ao fato de ser utilizado por módulos próprios, explicados a seguir. Os módulos que utilizam o módulo opcional *Blog* são:

módulo "debate" – localizado na sub-área comunidades;

módulo "*blog*" – localizado na sub-área do usuário autenticado ou na sub-área de um de seus contatos.

5.2 MÓDULOS PRÓPRIOS

Os módulos próprios foram desenvolvidos pela equipe do projeto, especificamente ao ambiente virtual de aprendizagem para atender os requisitos do sistema citados no capitulo quatro deste presente trabalho. A figura 21 demonstra como eles se comunicam e suas dependências no ambiente:

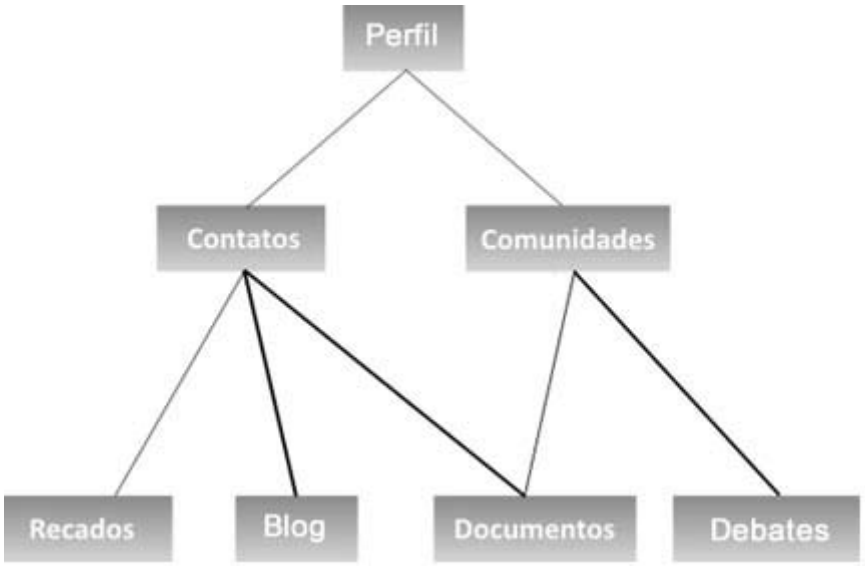


Figura 21 – Esquematização da comunicação e dependências entre os módulos próprios.

5.2.1 PERFIL

É o módulo de base da plataforma, nele são criadas as regras e a estrutura geral do sistema, como menus, autenticação e integração com a interface gráfica (*theme* do Drupal). Esse módulo também se destaca por ter a funcionalidade de gerenciamento do perfil do usuário, onde um perfil se caracteriza pelo conjunto de autorizações concedidas a um usuário. Desta forma, o gerenciamento do perfil permite:

Manutenção e visualização de usuários: criação, remoção, bloqueio e desbloqueio de acesso à área restrita do sistema (funções exclusivas para perfis administrativos);

Edição – função administrativa e de participantes;

Visualização – função exclusiva para usuário com perfil de participante;

Relação entre os usuários e perfis.

5.2.2 COMUNIDADES

Permite a criação e o gerenciamento de “Comunidades” do ambiente. Comunidades são redes de relacionamento quaisquer, como disciplinas, grupos de trabalho e temas de interesse.

Entre as funcionalidades do módulo estão:

Criação e edição da comunidade – o usuário que cria uma comunidade é por padrão seu dono (possui privilégios absolutos na comunidade, e também as permissões do perfil gestor). O dono de uma comunidade gerencia o grupo de participantes.

Configuração da comunidade – permite remover quaisquer de seus conteúdos, adicionar novos gestores (possui privilégios limitados) e donos, aceitar participantes nas comunidades restritas.

Uma comunidade pode estar configurada para acesso aberto, restrito ou fechado. Quando aberto, qualquer usuário do ambiente pode ver o conteúdo e se inscrever na comunidade; se

restrito, a inscrição necessita da aprovação de um gestor; e fechado o acesso depende do convite de um gestor.

5.2.3 CONTATOS

Permite a listagem, procura inserção e remoção de contatos da lista de contatos de um usuário ou de uma comunidade. Contatos são pessoas de alguma forma relacionadas com um usuário. Este módulo tem por objetivo gerenciar a lista de contatos para facilitar a troca rápida de informações, como por exemplo, novas mensagens e novos documentos.

Quando alguém adiciona um usuário à sua lista de contatos, o usuário adicionado recebe uma notificação e precisa aceitar a requisição para que um passe a ser contato do outro. Da mesma forma, quando um contato for removido, automaticamente o usuário sai da lista de contatos dessa pessoa.

5.2.4 DEBATES

Permite a edição e a visualização de postagens/comentários na área “Debates” do ambiente. Os comentários são específicos para uma determinada postagem. No caso de uma postagem de “Debates” ela pertence a uma comunidade e é criada por qualquer participante desta. Para comentar uma postagem o usuário deverá ser participante de tal comunidade.

5.2.5 DOCUMENTOS

Permite o carregamento, remoção e visualização de documentos na área “Documentos” do ambiente. Esses documentos são arquivos (textos, planilhas, imagens...) carregados por usuários no sistema e compartilhados para outros usuários. A área dedicada a documentos existe tanto no espaço de usuário como no espaço da comunidade. Este módulo realiza uma série de controles e particularidades referentes a permissões de usuários, são elas:

Os documentos são carregados por um usuário em seu espaço pode ser visualizado por qualquer outro usuário logado no sistema, sendo que o único que pode excluí-lo é o próprio usuário que inseriu o documento;

No espaço de comunidades do tipo disciplina são utilizadas “etiquetas padrões”, forma pré-definida para organizar os documentos. Nas etiquetas denominadas aulas, artigos, livros e programa da disciplina somente gestores poderão incluir documentos. Demais participantes da comunidade só poderão incluir documentos em etiquetas do tipo oportunidades de publicação, links e trabalhos.

Em comunidades que não sejam do tipo disciplinas a inclusão e visualização dos documentos pode ser realizada somente por participantes e a remoção somente por gestores.

5.2.6 *BLOG*

Possui o mesmo funcionamento de “Debates”, permitindo edição e visualização de postagens/comentários na área “*Blog*” do ambiente. Os comentários são específicos para uma determinada postagem o que difere em relação ao módulo “Debates” é que no caso do “*Blog*” este é feito para o espaço de usuários, e não de comunidades. A postagem de “*Blog*” é criada na área de um determinado usuário somente por ele mesmo. Para comentar uma postagem de “*Blog*” de outro usuário, o usuário pode ser qualquer um logado no sistema.

5.2.7 RECADOS

Permite envio e visualização de itens na área “Recados” do ambiente. Os recados são mensagens de texto deixadas por outras pessoas do sistema, sejam elas contatos do usuário ou não.

6. DESENVOLVENDO A FERRAMENTA RECADOS

Neste capitulo, será demonstrada a criação de um uma ferramenta por meio do Drupal, o modulo a ser desenvolvido é o modulo de recados, uma ferramenta de simples implementação e de fácil entendimento.

Para criação de um modulo no Drupal é gerada, primeiramente, uma pasta com o nome recados (dentro da pasta sites/all/modules para manter separados dos módulos do *core* do

Drupal), e adicionamos a este no mínimo dois arquivos, porem no caso a seguir serão três, pois será criada uma tabela na base de dados.

6.1 RECADOS.INFO

O primeiro arquivo a ser criado nesta demonstração será o modulo recados.info. Seu conteúdo é o seguinte:

```
; $Id$
name = Recados
description = Permite troca de mensagens entre usuários do sistema.
dependencies[] = perfil
core = 6.x
package = "Espaço Conhecimento"
version = 0.2
```

Esse arquivo contem informações a respeito do modulo, essa informações irão diretamente para a área de administração do *site*, em forma de um resumo do módulo que pode ser habilitado ou desabilitado como mostra a figura 22. No caso do exemplo acima, temos:

O nome do modulo;

A sua descrição;

As dependências que são informadas na forma de *array*, serve para alertar que este modulo depende da instalação anterior de outro modulo para ser instalado;

O *core*, descreve para qual versão do Drupal o módulo é compatível;

Packcage cria um novo grupo onde pode ser inseridos inúmeros outros módulos que se encontram relacionados entre si de alguma forma;

E finalmente a versão, para alertar em qual versão se encontra a implementação do modulo.

Outras propriedades podem ser adicionadas a este documento, porem nesse projeto utilizaremos basicamente essas.

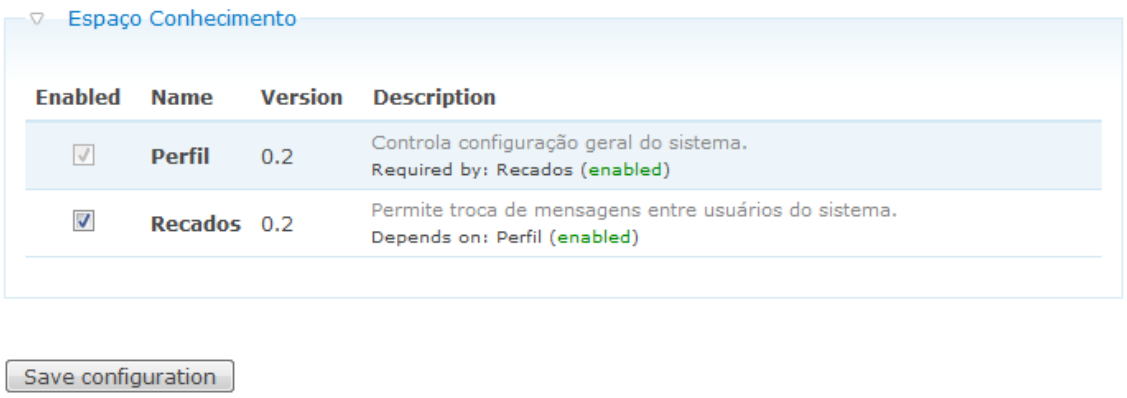


Figura 22 – Grupo (Espaço Conhecimento) de módulos próprios junto ao *Core* do drupal.

6.2 RECADOS.INSTALL

A seguir será descrito o arquivo recados.install. Um arquivo basicamente para camada de persistência. Aqui são adicionadas mais tabelas alem das que já vem por padrão no Drupal. Como já visto anteriormente as tabelas podem ser MySql ou PostgreSQL. Nesse caso utilizaremos MySQL.

Três funções básicas serão encontradas nesse arquivo. A primeira função cria a tabela por meio do código é a seguir:

```
function recados_install() {

  drupal_install_schema('recados');

}
```

A segunda função cria os campos da tabela criada no passo anterior, o código completo desse arquivo constará nos anexos deste trabalho, uma parte do código será dada a seguir.

```
function recados_schema() {

  $schema['recados'] = array(

    'description' => t('Guarda as mensagens dos usuários.'),
```

```
  'fields' => array(
    'rid' => array(
      'description' => t('Identificação do recado (chave primária).'),
      'type' => 'serial',
      'unsigned' => TRUE,
      'not null' => TRUE
    ),
    'uid' => array(
      'description' => t('Identificação do proprietário do recado (destinatário).'),
      'type' => 'int',
      'unsigned' => TRUE,
      'not null' => TRUE
    ),
```

Os campos da tabela são construídos pelo Drupal por meio de *arrays*. Note também que o Drupal utiliza descrições para cada campo da tabela, o primeiro *description* que aparece, é a descrição de toda a tabela, o segundo e o terceiro, são dos campos "*rid*" (id de recados) e "*uid*" (id do usuário a qual pertence o recado) respectivamente, outro fato que ganha destaque aqui é a função t() do Drupal, que é notada nas descrições. A função t() do Drupal é utilizada em *strings* e serve para facilitar na tradução da plataforma para qualquer língua.

Por fim temos a função que exclui a tabela em caso da desabilitação do modulo. Seu código é dado a seguir.

```
function recados_uninstall() {
  drupal_uninstall_schema('recados');
}
```

6.3 RECADOS.MODULE

O ultimo arquivo a ser criado nessa demonstração é o recados.module. Nesse arquivo consta toda a lógica de um modulo, o tornando na opinião do autor, o arquivo mais importante. O modulo ‘recados’ oferece aos usuários uma forma de comunicação. Este módulo deve possuir uma pagina para listar os recados de cada usuário, um local para o usuário poder adicionar um recado a outro usuário, e a opção de remover tais recados. Assim com esses requisitos iniciaremos a construção desta ferramenta.

No primeiro passo se desejarmos, criamos uma função para *setar* a permissão de acesso deste módulo, o código é demonstrado abaixo. Esta permissão poderá ser visualizada na área administrativa do Drupal, no item de permissões de usuários como mostra a Figura 23.

```
function recados_perm() {
  return array('visualizar recados');
}
```

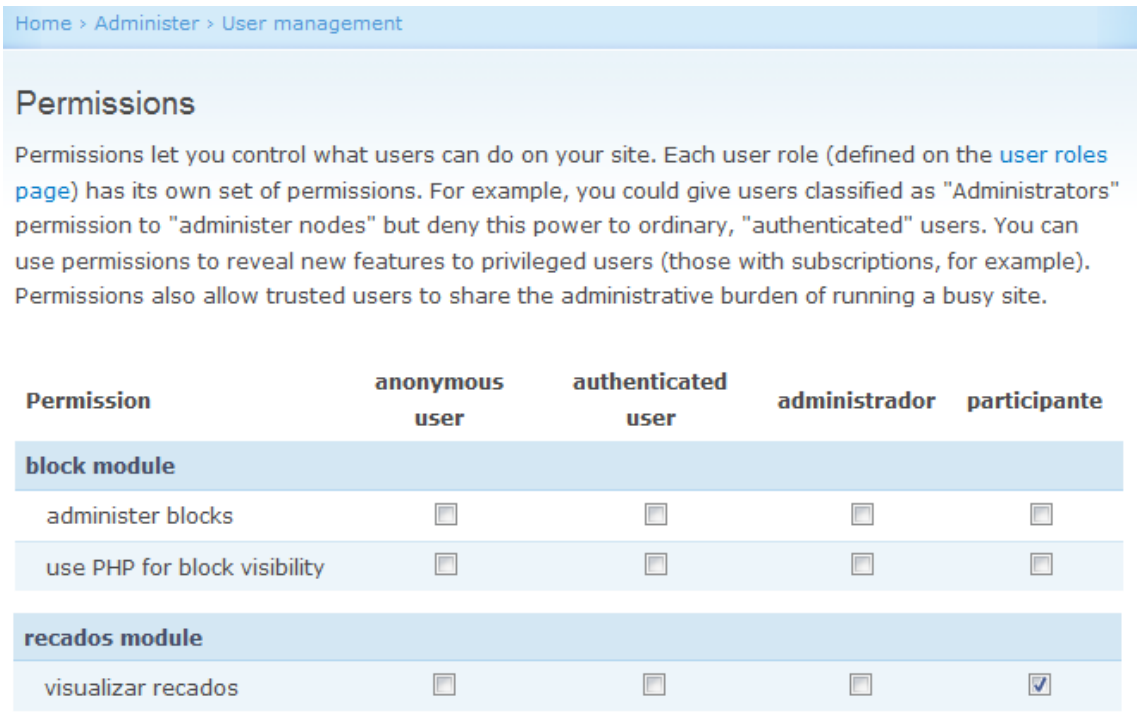


Figura 23 – Mostra a área de gerenciamento de permissões de usuários.

O passo a seguir é a criação do hook_menu, esses menus são responsáveis por montar boa parte da estrutura do Drupal, aqui é onde criamos os caminhos de acesso para uma requisição, é no hook_menu onde são chamadas outras funções, passamos parâmetros, onde decidimos quem pode acessar tal url. Parte do código será exposto a seguir e descrito com maiores detalhes.

```
function recados_menu() {
  $itens = array();
  $permissao = 'visualizar recados';
  $itens['usuario/%/recados'] = array(
    'title' => 'Recados',
    'page callback' => 'recados_listar',
    'page arguments' => array(1),
    'access arguments' => array($permissao),
  );

  $itens['recados/adicionar/%'] = array(
    'title' => 'Adicionar recado',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('recados_adicionar', 2),
    'access arguments' => array($permissao),
```

```
    'type' => MENU_CALLBACK,
  );
```

Todo *hook_menu*, é um *array* associativo, cada item desse *array* é um caminho (path), como no código acima 'usuario/%/recados' que anexados ao caminho base do Drupal, formam uma url de acesso a esse menu. Por exemplo, http://www.drupal/?q=usuario/5/recados, seria a url para acesso a lista de recados do usuário de ‘id = 5’, verifica-se assim que no caminho 'usuario/%/recados' o símbolo ‘%’ é uma variável, que no caso é substituída pelo ‘*id*’ de um usuário qualquer.

No primeiro item do *menu* consta: o *title* (título) que aparece no topo da pagina; Por meio do *'page callback'* é chamada a função recados_listar que será analisada no decorrer deste capitulo; *'page arguments'* onde é dito ao Drupal que a variável do caminho 'usuario/%/recados' se encontra na posição 1; E *'access arguments'* onde é passada a permissão de acesso para esse *menu*.

O segundo item do *menu*, difere do primeiro em dois itens. Neste caso o *'page callback'* avisa ao Drupal que a função a ser chamada se trata de um formulário. Logo, a função a ser chamada 'recados_adicionar' aparece em *'page arguments'* seguida da posição da variável nesse caso.

O último item de cada *array* é o tipo (*type*) do item do *hook_menu*, existem nove tipos de itens de *menu* no Drupal. Foram utilizados por esta implementação itens do tipo “*MENU_CALLBACK”* que criam paginas visíveis somente por meio do endereço da URL. Diversos tipos de menus podem ser criados como é apresentado na figura 24.

Constant	MENU_IS_ROOT	MENU_VISIBLE_IN_TREE	MENU_VISIBLE_IN_BREADCRUMB	MENU_VISIBLE_IF_HAS_CHILDREN	MENU_MODIFIABLE_BY_ADMIN	MENU_MODIFIED_BY_ADMIN	MENU_CREATED_BY_ADMIN	MENU_IS_LOCAL_TASK	MENU_EXPANDED	MENU_LINKS_TO_PARENT
MENU_NORMAL_ITEM		x	x		x					
MENU_ITEM_GROUPING			x	x	x					
MENU_CALLBACK			x							
MENU_DYNAMIC_ITEM		x	x							
MENU_SUGGESTED_ITEM			x		x					
MENU_LOCAL_TASK								x		
MENU_DEFAULT_LOCAL_TASK								x		x
MENU_CUSTOM_ITEM		x	x		x		x			
MENU_CUSTOM_MENU	x	x			x		x			

Figura 24 – Tipos de itens de *hook_menu*.

A seguir é demonstrado partes do código de funções implementadas pelo autor deste trabalho, onde constam algumas das regras e da lógica do modulo de recados. O código abaixo é o da função que traz uma pagina onde os recados serão listados para o usuário. Se o usuário estiver em sua página de recados, tem-se as opções de responder, ou excluir o recado.

```
function recados_listar($uid) {
 $conteudo = '';
 global $user;
 $sql = "SELECT users.name, users.picture, recados.* FROM {recados}, {users} WHERE
  users.uid=recados.aid AND recados.uid='$uid' AND recados.status=1
  ORDER BY timestamp DESC";
 $resultado = pager_query($sql, 5);
 while ($dados = db_fetch_object($resultado)) {
  $conteudo .= '<p> <img src='.$dados->picture.'>';
  $conteudo .= '<p>'.$dados->name.' - ';
  $conteudo .= format_date($dados->timestamp, 'custom', 'd/m/Y - H:i').'<br>';
```

```
    $conteudo .= $dados->texto.'<br>';
      if ($uid == $user->uid) {
        $conteudo .= l('Remover','recados/remover/'.$dados->rid).' - ';
        $conteudo .= l('Responder','recados/adicionar/'.$dados->aid).'</p>';
      }
  }
    $conteudo .= theme('pager', NULL, 5);
  return $conteudo;
}
```

Para listar os recados, deve-se fazer uma busca pelos recados pertencentes a um usuário, e imprimir na pagina em forma de conteúdo, por isso é criada a variável $conteudo, que será o retorno dessa função. Primeiro $conteudo recebe (*null*). É declarado o objeto ‘global $user’ para se obter dados do usuário autenticado no sistema. A variável $sql, guarda a consulta SQL feita ao banco de dados onde $uid é o parâmetro passado para a função por meio da URL, gerada no *hook_menu* explicado anteriormente. A *query* SQL é depois passada por um método do drupal o *pager_query* responsável pela paginação dos recados onde são passados dois parâmetros, a *query*, e o limite de elementos por pagina que no caso são cinco.

Na seqüencia do código, é utilizado outro método do Drupal, que transforma cada linha retornada da consulta SQL em objetos. Com isso é concatenado um código misto de HTML e atributos do objeto a variável $conteudo que será retornada ao final da função.

A seguir é verificado, por meio do objeto $user, se o usuário autenticado esta na sua área de recados, ou na área de recados de outro usuário. Se o usuário estiver na sua área de recados tem-se disponível na pagina abaixo de cada recado a possibilidade de resposta, e exclusão do recado.

Seguindo código abaixo, é chamado o método *theme* do Drupal, que renderiza na tela a numeração e *links* para a paginação. Por fim é retornado o “conteúdo” da página 'usuario/%/recados'.

A próxima função a ser vista, será o formulário de envio dos recados, a função de formulários no Drupal, requer mais duas funções, uma de validação dos valores enviados pelo formulário, e outra para inserção dos dados na base de dados.

```
function recados_adicionar($formulario_estado, $uid) {
 if (!is_numeric($uid)) {
  return drupal_access_denied();
 }

 $formulario['recado'] = array(
  '#type' => 'fieldset',
 );
 $formulario['recado']['texto'] = array(
  '#title' => t('Recado'),
  '#type' => 'textarea',
  '#resizable' => FALSE,
  '#required' => TRUE,
 );
 $formulario['recado']['enviar'] = array(
  '#type' => 'submit',
```

```
  '#value' => t('Enviar recado')
 );
```

O código acima esta incompleto, o completo se encontra nos anexos, a função recados _adicionar recebe um *array* com propriedades de um formulário, e o identificador do usuário ao qual será enviada a mensagem. Em seguida é criado um *array* $formulario [recado] onde serão guardados todos os campos do formulário, como o campo texto onde ira a mensagem, e o campo *submit*.

O retorno dessa função será o $formulario que será passado como parâmetro para a função recados_adicionar_validate via *post* que é a configuração padrão do Drupal. Alem de receber o formulário a ser tratado, recebe também as propriedades desse formulário por meio do $formulario_estado. Nesta função e feita toda uma verificação dos dados vindos do formulário, para um possível aviso de erro ao usuário, e checagem de algum provável código mal intencionado que o Drupal já faz automaticamente, para questão de segurança do sistema. O código se encontra nos anexos do trabalho.

Após a validação do formulário de envio, finalmente é chagada o hora de inserção deste na base de dados. Este processo é feito na função recados_adicionar_submit por uma simples query SQL. Parte do código de recados_adicionar_submit sera demonstrado a seguir.

```
 global $user;
 $aid = $user->uid;
 $uid = $formulario_estado['values']['uid'];
 $texto = $formulario_estado['values']['texto'];
 if (db_query("INSERT INTO {recados} (uid, aid, texto, timestamp)
  VALUES (%d, %d, '%s', %d)", $uid, $aid, $texto, time())) {
  drupal_set_message(t('Messagem enviada. Obrigado pela sua participação.'));
 }
 else {
```

```
    drupal_set_message(t('Houve algum problema no sistema e não foi possível enviar a mensagem.'), 'error');
  }
  $formulario_estado['redirect'] = 'usuario/'.$uid.'/recados';
```

O código uid de quem recebera a mensagem, e o texto são obtidos por meio da variável $formulario_estado, quatro campos são inseridos na tabela, na *query* de inserção, avisamos ao Drupal, que o terceiro valor a ser inserido na base de dados é uma *string* '%s' e os demais valores são inteiros '%d'.

7. CONCLUSÃO

Este trabalho de conclusão de curso baseia-se em uma tendência do ensino e aprendizagem, de modo a permitir meios alternativos a alunos e professores, possam se comunicar por meio do uso de ferramentas adequadas sem a necessidade de um encontro presencial. Seu desenvolvimento é baseado em preceitos didáticos e pedagógicos importantes que justificaram a implementação desta plataforma.

Com base nas necessidades dos usuários do sistema definiu-se uma proposta de implementação de ferramentas (funcionalidades) e da usabilidade do sistema computacional. A partir desse ponto, definiu-se o CMS a ser adotado para o desenvolvimento da plataforma, que serve de base fundamental deste trabalho. A estrutura de módulos do CMS facilita no fator de personalização do ambiente. Visto que as necessidades fugiam dos módulos existentes dos CMSs, módulos próprios foram elaborados para atender tais necessidades.

Ao concluir os estudos de CMSs, foi notado que em qualquer uma das opções, trabalhos futuros com relação a implementação eram inevitáveis, levando-se assim em conta o CMS com melhor documentação, e com características que mais se enquadravam as necessidades. Alem disso o Drupal, CMS escolhido, conta com um código robusto, seguro e com facilidades de configurações.

7.1 RESULTADOS ALCANÇADOS

Para serem alcançados os objetivos deste trabalho, foram estudadas tecnologias *Web*, arquitetura de *software* com ênfase na arquitetura *Web*, ambientes de colaboração, e sistemas gerenciadores de conteúdo. A partir da compreensão dessa fundamentação teórica foi possível uma proposta com a arquitetura desejada em função das questões pedagógicas de tratamento da informação.

7.2 TRABALHOS FUTUROS

A estrutura e soluções propostas do ambiente de colaboração deste trabalho ficarão como base para futuras pesquisas nesse sentido, principalmente quanto a utilização do Drupal. Apesar da qualidade do Drupal como CMS, pode-se pensar em estender as ferramentas desenvolvidas para outros CMSs, como Moodle, Jomla!, XOOPs, entre outros. Há ainda a possibilidade de se estender tais ferramentas para outras linguagens de programação.

Algumas ferramentas futuras foram sugeridas no tópico 4.6, também são funcionalidades que ganham destaque quando se trata de praticidade e inovação da plataforma.

Por fim, a realização de testes completos por meio de alunos e professores a fim de avaliar o real resultado do uso deste ambiente de aprendizagem, bem como de suas funcionalidades e características pedagógicas.

7.3 CONSIDERAÇÕES FINAIS

Com a facilidade de pesquisa por meio da internet, esse meio de acesso as informações torna-se inevitável para alunos e professores no mundo atual. Está clara com o rumo das tecnologias e da educação, a importância de trabalharem juntas no processo de aprendizagem, porem, deve-se haver a preocupação e exata avaliação dos resultados da tecnologia e não apenas o que acontece com o uso da tecnologia.

A busca por informação, a independência de um individuo, ou ate mesmo de um grupo, estão cada vez mais valorizadas, instituições de ensino que ignorassem essa realidade apresentariam todas as chances de obterem somente resultados fortuitos, pois segundo a pesquisa realizada

toda pedagogia que obteve sucesso foi diferenciada e adaptada aos indivíduos aos quais foi proposta.

O ambiente de colaboração se destaca por assumir também o papel de repositório de informações, onde grupos e/ou indivíduos podem buscar em tópicos antigos aquilo que lhes interessa. Essa consciência vai alem das instituições de ensino, repositórios na *Web*, são utilizados por diversos setores, como empresas, lojas, e ate indivíduos que postam suas rotinas e conhecimentos em *blogs*, e páginas pessoais.

Projetar um ambiente de colaboração baseado em CMS foi o principal objetivo deste trabalho, não se preocupando nesse primeiro momento com o desenvolvimento de todo o código, mas sim deixando uma estrutura de ferramentas e usabilidade da plataforma. Considera-se assim alcançados os objetivos deste trabalho.

8. REFERÊNCIAS BIBLIOGRÁFICAS

ARAÚJO, R. C. and ROCHA, H. V. *Um Ambiente de Colaboração para Instituições Educacionais.* Campinas : s.n.

ASHTON, Elaine. 1999-2001. The Timeline of Perl and its Culture. *CPAST.* [Online] 1999-2001. http://history.perl.org/PerlTimeline.html.

ASTUDILLO, Hernán, HAMMER, Stuart. 1998. Understanding Architecture: What we do and why we do it. *Software Architecture Workshop of OOPSLA'98.* 1998.

BATES, A. W. Tony. 2005. *Technology, e-learning and distance education.* s.l. : Routledge, 2005.

BAX, MARCELLO P. and PEREIRA, JÚLIO C. L. 2002. INTRODUÇÃO À GESTÃO DE CONTEÚDOS. *ANAIS - CONGRESSO ANUAL DA SOCIEDADE BRASILEIRA DE GESTÃO DO CONHECIMENTO.* 2002.

BERNERS-LEE, T and CONNOLLY, D. 1995. Hypertext Markup Language – 2.0. *Hypertext Markup Language – 2.0 Capítulo "Introdution".* [Online] set 22, 1995. http://www.w3.org/MarkUp/html-spec/html-spec_toc.html.

BERNERS-LEE, T. 1989. Information Management: A proposal. *The Original Proposal of The WWW.* [Online] 1989. http://www.w3.org/History/1989/proposal.html.

BOARD, P., LUNA, R. and DELL, D. 1996. Creating Shockwave Web Pages. s.l. : Que Corporation, 1996.

BRITO, R. F. and PEREIRA, A. T. C. 2004. Um estudo para ambientes colaborativos e suas ferramentas. Florianópolis : s.n., 2004.

CAMARGO, Álvaro Antônio Bueno De. KHOURI, Lourdes Halim El e GIAROLA, Paulo César. 2005. O Uso de Sistemas Colaborativos na Gestão de Projetos: Fatores Relevantes para o Sucesso. 2005.

CHAPPELL, David. 1996. *Understanding ActiveX and OLE: a guide for developers and managers.* Redmond : Microsoft Press, 1996.

CONALLEN, Jim. 1999. Modeling Web Application Architectures with UML. *Communications of the ACM (volume 42, número 10). .* Outubro 1999.

CONVERSE, Tim and PARK, Joyce. 2003. *Php - A Biblia.* s.l. : Campus, 2003.

2006. CSS: conceitos e mão na massa. *oficina da NET.* [Online] Julho 2006. http://www.oficinadanet.com.br/artigo/35/css_conceitos_e_mao_na_massa.

Developer Resources for Java Technology. [Online] http://java.sun.com/.

drupal.org. [Online] http://drupal.org/.

2005. Dynamic HTML and XML: The XMLHttpRequest Object. *Apple Developer Connection.* [Online] jun 2005. http://developer.apple.com/internet/webcontent/xmlhttpreq.html.

FLANAGAN, David. 2001. *JavaScript: The Definitive Guide.* s.l. : O'Reilly Media, 2001.

FONTOURA, Wagner. 2008. A hora e a vez das Mídias Sociais. *Ministério da Cultura.* [Online] fev 29, 2008. http://www.cultura.gov.br/site/2008/02/29/a-hora-e-a-vez-das-midias-sociais/.

GARRETT, Jesse James. 2005. Ajax: A New Approach to Web Applications. feb 18, 2005.

HANSSON, David Heinemeier. 2006. *Interview with David Heinemeier Hansson from Ruby on Rails.* [interv.] Lenz GRIMMER. fev 2006.

JAZAYERI, Mehdi, RAN, Alexander, LINDEN, Frank van der. 2000. *Software Architecture for Product Families: Principles and Practice.* s.l. : Addison Wesley, 2000.

joolma.org. [Online] http://www.joomla.org/.

KOSCHMANN, T., KELSON, A. C., FELTOVICH, P. J., & BARROWS, H. S. 1996. *CSCL: Theory and Practice of an Emerging Paradigm.* s.l. : Koschmann , 1996.

LALLI, F. M., BUENO, F. F. and ZACHARIAS, G. K. 2008. Evolução da Programação Web. Campinas : s.n., 2008.

LEVINE, JONATHAN and WALTHER, STEPHEN. 2003. *Aprenda em 21 dias E-Commerce com ASP.* Rio de Janeiro : Campus, 2003.

2007. libertas.pbh.gov.br. *Projeto Libertas.* [Online] mai 2007. http://libertas.pbh.gov.br/drupal/?q=node/86.

mambo-foundation.org. [Online] http://mambo-foundation.org/.

MARCONDES, C. A. 2005. *HTML 4.0 Fundamental: A Base da Programação para web.* 1ª ed. São Paulo : Érica, 2005.

MARTINS, J. G. et al. 1999. A transformação do ensino através do uso da tecnologia da educação. *XIX Congresso Nacional da Sociedade Brasileira de Computação, PUC. Anais.* Rio de Janeiro : s.n., 1999.

moodle.org. [Online] http://moodle.org/.

Open Source CMS Award. [Online] http://www.packtpub.com/award/.

PACHECO, CARLOS EDUARDO. 2007. ENSINO INFORMATIZADO DE MÁQUINAS ELÉTRICAS. FLORIANÓPOLIS : s.n., 2007.

PEREIRA, A. C. B. G. 2008. Blog, mais um gênero do discurso digital?. São Paulop.516-523. [Online] 2008. http://www3.unisul.br/paginas/ensino/pos/linguagem/cd/Port/9.pdf.

phpnuke.org. [Online] http://phpnuke.org/.

PRESSMAN, Roger S. 1995. *Engenharia de Software.* s.l. : Makron Books, 1995.

RAVEN, Max. 2007. CMS – Uma introdução aos Sistemas Gestores de Conteúdo Web. *Guia do Hardware.* [Online] 2007. http://www.guiadohardware.net/artigos/cms/.

ROBERTSON, James. 2003. Choosing the right CMS authoring tools. *Step two Designs.* [Online] nov 3, 2003. http://www.steptwo.com.au/papers/kmc_authoringtools/index.html.

ROBINSON, D. and COAR, K. 2004. The Common Gateway Interface (CGI) Version 1.1. *Especificação oficial do CGI na IETF (RFC 3875).* [Online] October 2004. http://www.ietf.org/rfc/rfc3875.

Ruby - The programmer's Best Friend. *Linguagem de programação Ruby.* [Online] http://www.ruby-lang.org/pt/.

SANTOS, NEI RAUNI. 2007. DESENVOLVIMENTO DE SITES COLABORATIVOS. Curitiba : s.n., 2007.

SCHWIER, A. Richard and BALBAR, Shelly. 2002. The Interplay of Content and Community in Synchronous and Asynchronous Communication: Virtual Communication in a Graduate Seminar. *Canadian Journal of Learning and Technology.* 2002.

SILVA, Maurício Samy. 2007. *Criando Sites com CSS e XHTML.* São Paulo : Novatec, 2007.

The Common Gateway Interface. *Site oficial do CGI - NCSA HTTPd Home Page.* [Online] http://hoohoo.ncsa.uiuc.edu/cgi/primer.html.

VANDYK, John K. and WESRGATE, Matt. 2007. *Pro Drupal Development: Learn How to Use The Content Management Framework to Create Powerful Customized Web Sites.* New York : Apress, 2007.

WASSERMAN, Stanley, FAUST, Katherine. 1994. Social network analysis: methods and applications. s.l. : Cambridge University Press, 1994.

WINCKLER, Marco Antônio and PIMENTA, Marcelo Soares. 2002. Avaliação de Usabilidade de Sites Web. *NEDEL, Luciana Porcher. (Org.). Escola de Informática da SBC SUl (ERI 2002). Porto Alegre, v. 1.* 2002, pp. 85-137.

wordpress.org. [Online] http://wordpress.org/.

xoops.org. [Online] http://www.xoops.org/.

9. ANEXOS

9.1 ARTIGO

Arquitetura de Software para Ambiente de Colaboração Educacional, Baseado em CMS.

Daniel C. Santos

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (USFC) – Florianópolis, SC – Brasil.

dcsantos@inf.ufsc.br

Abstract. *This paper deals with practical questions concerning the building of software architecture for the web having as purpose the creation of a collaborative environment for educational institutions. In order to facilitate the platform implementation a CMS (Content Management System) has been employed. The architecture is based upon existing concepts (such as chat, forum, blog) and some adaptations have been introduced to facilitate the system tools utilization by the user.*

Resumo. *Este trabalho aborda aspectos práticos na elaboração de uma arquitetura de software para Web tendo por objetivo a criação de um ambiente de colaboração para instituições de ensino. Para facilitar a implementação da plataforma, decidiu-se utilizar um Sistema Gerenciador de Conteúdo (CMS – Content Management System). A arquitetura é baseada em conceitos já existentes (como chat, fórum, blog), com pequenas adaptações feitas para facilitar a utilização das ferramentas do sistema por parte do usuário.*

1. Introdução

Experiências obtidas no desenvolvimento de softwares vêm trazendo a instituições e empresas preocupações com as questões de qualidade na construção de sistemas. Cada vez mais se tem a necessidade de definir processos internos de desenvolvimento, incluindo técnicas de modelagem e ferramentas de teste e ressaltando a importância da documentação como registro das decisões e definições adotadas.

A arquitetura de um sistema busca uma solução técnica para um problema. Uma arquitetura é definida de forma a satisfazer todos os requisitos (requisitos funcionais e requisitos não funcionais), além dos subsistemas e componentes que compõem este sistema. Com isso notamos que a arquitetura não engloba apenas informações tecnológicas e de infra-estrutura, outros fatores também devem ser levados em consideração.

Com o amadurecimento da tecnologia *Web*, muitos cenários são trazidos para facilitar a aprendizagem e a troca de informações entre estudante e professor. Atualmente, materiais como trabalhos, aulas, artigos e livros são facilmente disponibilizados em algum endereço *Web*. Com isso, surgiram inúmeros endereços, ambientes e plataformas para o estudante acessar e pesquisar conteúdos. Assim, surgiu a idéia de se juntar todos esses recursos em um único ambiente. Os dois

principais conceitos, comunidades e contatos (alunos e professores), trazem uma maior flexibilidade entre a troca de informação.

2. Arquiteura de Software

Antes do surgimento da arquitetura de *software*, o desenvolvimento de sistemas e aplicações de computação ocorria de forma desordenada e sem um padrão de estrutura e métodos o que gerou a "crise do *software*", denominada assim por Pressman [PRESSMAN, 1995]. A solução encontrada para sanar esse problema foi a Engenharia de *Software* determinando "o uso de sólidos princípios de engenharia para que se possa obter economicamente um *software* que seja confiável e que funcione eficientemente em máquinas reais" [PRESSMAN, 1995].

> *"A arquitetura de software é colocada como uma ferramenta para lidar com a complexidade do software possibilitando a criação de um conjunto de componentes e definindo seus relacionamentos. O autor enfatiza que arquitetura deve satisfazer os requisitos funcionais e não funcionais do sistema. Portanto, é possível notar que a arquitetura de software é mais do que a descrição dos componentes que a compõem e do relacionamento entre eles"* [JAZAYERI, 2000].

Para Astudillo, "A arquitetura é a interface entre duas partes distintas: o problema de negócio e a solução técnica" [ASTUDILLO, 1998].

- **Arquitetura De Sistemas *Web***

Existe na programação *Web* uma arquitetura chamada Cliente/Servidor, a comunicação entre o cliente e o servidor é feita sobre TCP/IP, onde é usado um protocolo de alto nível denominado HTTP (*Hipertext Transfer Protocol)* [BERNERS-LEE, 1989].

O Cliente são os *browsers* (navegadores *Web* como Firefox, Safári, Netscape, Internet Explorer) que executados em um host que solicitam informações ao servidor normalmente por meio da rede [BERNERS-LEE, 1989].

Servidores *Web* (exemplos: *Apache* e IIS) são aplicações que ficam em espera, e aguarda requisições do lado do cliente. A requisição vai para o servidor e este por sua vez verifica se é necessário que se rode uma aplicação por meio da extensão do arquivo. A aplicação é iniciada pelo servidor e recebe as informações em forma de entrada padrão e variáveis de ambiente. As informações são processadas pela aplicação que inclui acessos aos sistemas de informações ou aos repositórios de informações e produz uma saída, inclui ainda um *header* que informará ao servidor o tipo da informação produzida. O servidor recebe o *header* e os dados de respostas, e repassa em formato interpretável por *browsers Web* as informações para o cliente [WINCKLER & PIMENTA, 2002].

3. Ambientes de Colaboração

Produzem-se potencialmente melhores resultados trabalhando colaborativamente do que se os membros do grupo atuassem individualmente. Em um grupo, a capacidade, conhecimento e o esforço individual de um dos membros podem complementar o dos outros. Há também interação entre pessoas com entendimentos, pontos de vista e habilidades complementares [BERNERS-LEE T, 1989]. A informação não deve ser de fonte única e sim uma por meio de diversas fontes. Cada fonte de informação deve ter suas particularidades para atender corretamente em um dado momento as

necessidades de um aluno na busca por novos conhecimentos. Nosso mundo se caracteriza pela abertura, flexibilidade, mudanças constantes e necessidade de respostas rápidas e nem sempre um aluno conta com a presença constante de um professor [BATES, 2005].

4. CMS

O CMS (*Content Management System*, ou em português, Sistema de Gerenciamento de Conteúdo) é na verdade um sistema gerenciador de *websites*, portais e intranets que agrega ferramentas necessárias para gerenciar (inserir, editar, remover) conteúdo em tempo real, não havendo a necessidade de programação por meio de linhas de código [BAX & PEREIRA, 2002]. O CMS é uma ferramenta criada para facilitar a vida dos usuários. O objetivo é estruturar e facilitar a administração, criação e publicação de conteúdo de forma dinâmica, por meio de uma interface de usuário via Internet [BAX & PEREIRA, 2002] [RAVEN, 2007].

O CMS funciona como um *framework*, um "esqueleto" de *website* pré-programado, com recursos de administração e manutenção disponíveis. Um *website* criado com um CMS tem sua aparência facilmente customizável por meio da utilização de estilos CSS[10] e modelos (*templates*) que podem ser facilmente adaptados [libertas.pbh.gov.br, 2007].

5. Detalhamento do CMS Selecionado

Investigando a fundo cada plataforma e suas comunidades ativas, nota-se que em termos de código e API, o Drupal seria a plataforma com a mais vasta documentação (api.drupal.org). O Drupal venceu o ultimo premio geral de melhor CMS *(Overall Open Source CMS Award 2008)* [Open Source CMS Award]. Apesar de praticamente não apresentar construtores de classe, aplica princípios de orientação a objetos como encapsulamento, herança, polimorfismo, etc. Sua API além de bem documentada tem uma busca que facilita na hora de necessidade de consultas.

Por ser a primeira experiência com CMS, decidiu-se pela facilidade de pesquisa a documentação e por meio de maior compatibilidade com as necessidades da plataforma, o Drupal como CMS. E para descrever a arquitetura de *software* devemos antes conhecer o Drupal e a sua arquitetura.

O Drupal [VANDYK & WESRGATE, 2007] [SANTOS, 2007] [drupal.org] é uma ferramenta desenvolvida justamente para se encaixar principalmente em duas dessas classificações: Sistemas Colaborativos de Gerenciamento de Conteúdo e Portais e Comunidades *on-line*, mas também pode ser aplicado nas outras classificações sem problemas. Sendo o Drupal um sistema colaborativo de código aberto criado justamente para colaboração entre usuários. Esta ferramenta mantida e desenvolvida por uma comunidade ativa pode ser facilmente adquirida por meio do próprio *site* (www.drupal.org). Além de um sistema gerenciador de conteúdo ele também é uma plataforma de desenvolvimento, onde desenvolvedores *Web* utilizam-no para criar sistemas complexos que vai desde comércio eletrônico até produção de livros colaborativos.

6. Requisitos Funcionais do Sistema

[10] CSS é a sigla em inglês para *Cascading Style Sheet* que em português foi traduzido para folha de estilo em cascata. É um mecanismo simples para adicionar estilos (p.ex., fontes, cores, espaçamentos) aos documentos *Web*.

O ambiente de colaboração educacional baseado em CMS será tratado deste ponto em diante pelo nome Espaço Conhecimento. Esta nomenclatura foi assumida por tratar-se de um ambiente virtual utilizado para conectar pessoas e idéias.

O Espaço Conhecimento, de forma colateral, irá congregar funções atualmente exercidas pela página do curso, sistema de gestão acadêmica, gestão de conteúdos e ferramenta colaborativa. O funcionamento do sistema será baseado em torno de dois conceitos: comunidades e contatos.

As funcionalidades a seguir surgiram com os resultados obtidos da fase de levantamento de requisitos que incluiu as técnicas de observação, entrevistas, questionários e reuniões com alunos e professores da instituição de ensino.

- **Perfis**– Onde constam informações referentes a um usuário específico e/ou a uma comunidade
- **Comunidades** – Podem ser disciplinas, grupos de trabalho ou uma rede de relacionamento qualquer
- **Contatos** – São os usuários cadastrados no sistema e que um usuário pode adicionar/remover um contato a sua rede de relacionamentos
- **Sistema de busca** – Onde são feitas buscas de comunidades e/ou contatos no sistema
- **Comunicação** – Por meio de ferramentas como recados, blog e debates os usuários comunicam-se de maneiras diferenciadas
- **Documentos** – Repositório de documentos como artigos, trabalhos, livros...
- **Integração** – Foi planejada uma integração com sistemas de gestão acadêmica atuais das instituições para evitar redundância de dados
- **Área Administrativa** – Ambiente para administradores gerenciarem conteúdos do sistema

7. ARQUITETURA PROPOSTA

Como principal foco do projeto foi observado à necessidade da criação de uma rede social. Uma rede social é uma estrutura social, comunidade, ou sociedade que correspondem geralmente a perfis de indivíduos ou organizações. Isto indica o modo como eles estão conectados por meio de várias familiaridades sociais, afiliações, e/ou relacionamentos que variam desde conhecidos casuais a ligações familiares próximas.

Com a instalação do Drupal, os módulos denominados núcleo (*core*) são automaticamente disponibilizados. No núcleo existem os módulos obrigatórios (*core-required*) e os módulos opcionais (*core-optional*). Somente os módulos do núcleo obrigatório (*core-required*) são requeridos para o sistema funcionar e não podem ser desativados. Apesar de inúmeros núcleos opcionais disponibilizados pelo Drupal, serão utilizados para o desenvolvimento desta plataforma somente dois, o *Locale* e o *Blog* que serão citados a seguir.

Além de toda e estrutura de módulos do Drupal (Core- Required) e (Core -Optional) notou-se a necessidade da criação de módulos próprios para atender as necessidades das instituições de ensino como mostra a Figura 1.

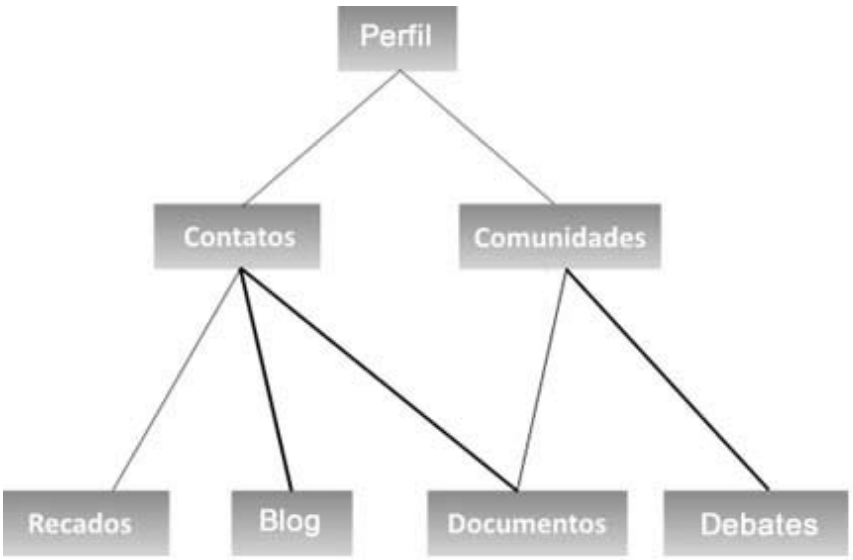


Figura 1. Esquematização da comunicação e dependências entre os módulos próprios

8. Conclusão

Este trabalho baseia-se em uma tendência do ensino e aprendizagem, de modo a permitir meios alternativos a alunos e professores, possam se comunicar por meio do uso de ferramentas adequadas sem a necessidade de um encontro presencial. Seu desenvolvimento é baseado em preceitos didáticos e pedagógicos importantes que justificaram a implementação desta plataforma.

Para serem alcançados os objetivos deste trabalho, foram estudadas tecnologias *Web*, arquitetura de *software* com ênfase na arquitetura *Web*, ambientes de colaboração, e sistemas gerenciadores de conteúdo. A partir da compreensão dessa fundamentação teórica foi possível uma proposta com a arquitetura desejada em função das questões pedagógicas de tratamento da informação.

O ambiente de colaboração se destaca por assumir também o papel de repositório de informações, onde grupos e/ou indivíduos podem buscar em tópicos antigos aquilo que lhes interessa. Essa consciência vai alem das instituições de ensino, repositórios na *Web*, são utilizados por diversos setores, como empresas, lojas, e ate indivíduos que postam suas rotinas e conhecimentos em *blogs*, e páginas pessoais.

Projetar um ambiente de colaboração baseado em CMS foi o principal objetivo deste trabalho, não se preocupando nesse primeiro momento com o desenvolvimento de todo o código, mas sim deixando uma estrutura de ferramentas e usabilidade da plataforma. Considera-se assim alcançados os objetivos deste trabalho.

9. Referências

ASTUDILLO, H. H. (1998). Understanding Architecture: What we do and why we do it. *Software Architecture Workshop of OOPSLA'98* .

BATES, A. W. (2005). *Technology, e-learning and distance education.* Routledge.

BAX, M. P., & PEREIRA, J. C. (2002). INTRODUÇÃO À GESTÃO DE CONTEÚDOS. *ANAIS - CONGRESSO ANUAL DA SOCIEDADE BRASILEIRA DE GESTÃO DO CONHECIMENTO* .

BERNERS-LEE, T. (1989). *Information Management: A proposal.* Retrieved Setembro 2008, from The Original Proposal of The WWW: http://www.w3.org/History/1989/proposal.html

BERNERS-LEE, T., & CONNOLLY, D. (1995, set 22). *Hypertext Markup Language – 2.0.* Retrieved Setembro 2009, from Hypertext Markup Language – 2.0 Capítulo "Introdution": http://www.w3.org/MarkUp/html-spec/html-spec_toc.html

drupal.org. (n.d.). Retrieved ago 2008, from http://drupal.org/

JAZAYERI, M. R. (2000). *Software Architecture for Product Families: Principles and Practice.* Addison Wesley.

libertas.pbh.gov.br. (2007, mai). Retrieved nov 2008, from Projeto Libertas: http://libertas.pbh.gov.br/drupal/?q=node/86

Open Source CMS Award. (n.d.). Retrieved abr 2009, from http://www.packtpub.com/award/

PRESSMAN, R. S. (1995). *Engenharia de Software.* Makron Books.

RAVEN, M. (2007). *CMS – Uma introdução aos Sistemas Gestores de Conteúdo Web.* Retrieved set 2009, from Guia do Hardware: http://www.guiadohardware.net/artigos/cms/

SANTOS, N. R. (2007). DESENVOLVIMENTO DE SITES COLABORATIVOS. Curitiba.

VANDYK, J. K., & WESRGATE, M. (2007). *Pro Drupal Development: Learn How to Use The Content Management Framework to Create Powerful Customized Web Sites.* New York: Apress.

WINCKLER, M. A., & PIMENTA, M. S. (2002). Avaliação de Usabilidade de Sites Web. *NEDEL, Luciana Porcher. (Org.). Escola de Informática da SBC SUl (ERI 2002). Porto Alegre, v. 1.* , pp. 85-137.

9.2 MÓDULO RECADOS

9.2.1 RECADOS.INFO

; Id

name = Recados

description = Permite troca de mensagens entre usuários do sistema.

dependencies[] = perfil

core = 6.x

php = 5.1

```
package = "Espaço Conhecimento"
version = 0.2
```

9.2.2 RECADOS.INSTALL

```
<?php
// $Id$

/**
* Implementação do hook_install().
*/
function recados_install() {
  drupal_install_schema('recados');
}

/**
* Implementação do hook_schema().
*/
function recados_schema() {
  $schema['recados'] = array(
    'description' => t('Guarda as mensagens dos usuários.'),
    'fields' => array(
      'rid' => array(
        'description' => t('Identificação do recado (chave primária).'),
        'type' => 'serial',
        'unsigned' => TRUE,
        'not null' => TRUE
      ),
      'uid' => array(
```

```
  'description' => t('Identificação do proprietário do recado (destinatário).'),
  'type' => 'int',
  'unsigned' => TRUE,
  'not null' => TRUE
),
'aid' => array(
  'description' => t('Identificação do autor do recado (remetente).'),
  'type' => 'int',
  'unsigned' => TRUE,
  'not null' => TRUE
),
'status' => array(
  'description' => t('Estado do recado (1 = visível, 0 = removido.'),
  'type' => 'varchar',
  'length' => 1,
  'not null' => TRUE,
  'default' => 1
),
'texto' => array(
  'description' => t('Conteúdo do recado.'),
  'type' => 'text',
  'size' => 'big',
  'not null' => TRUE
),
'timestamp' => array(
  'description' => t('Timestamp tipo UNIX da criação do recado.'),
  'type' => 'int',
  'not null' => TRUE
```

```
      )
    ),
    'primary key' => array('rid'),
  );

  return $schema;
}

/**
* Implementação do hook_uninstall().
*/
function recados_uninstall() {
  drupal_uninstall_schema('recados');
}
```

9.2.3 RECADOS.MODULE

```
<?php
// $Id$

/**
 * Implementação do hook_perm().
 *
 * @return  Lista de permissões para o módulo Recados
 */
function recados_perm() {
  return array('visualizar recados');
}
```

```
/**
 * Implementação do hook_menu().
 *
 * @return  Array de menus
 */
function recados_menu() {
  $itens = array();
  $permissao = 'visualizar recados';

  $itens['usuario/%/recados'] = array(
    'title' => 'Recados',
    'page callback' => 'recados_listar',
    'page arguments' => array(1),
    'access arguments' => array($permissao),
  );

  $itens['recados/adicionar/%'] = array(
    'title' => 'Adicionar recado',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('recados_adicionar', 2),
    'access arguments' => array($permissao),
    'type' => MENU_CALLBACK,
  );

  $itens['recados/remover/%'] = array(
    'title' => 'Tem certeza que deseja remover o recado?',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('recados_remover', 2),
```

```
    'access arguments' => array($permissao),
    'type' => MENU_CALLBACK,
   );

  return $itens;
}

/**
 * @param $uid = id do usuario.
 *
 * @return  Conteúdo HTML
 */
function recados_listar($uid) {
  $conteudo = '';
  global $user;
  $sql = "SELECT users.name, users.picture, recados.* FROM {recados}, {users} WHERE
    users.uid=recados.aid AND recados.uid='$uid' AND recados.status=1
    ORDER BY timestamp DESC";
  $resultado = pager_query($sql, 5);
  while ($dados = db_fetch_object($resultado)) {
    $conteudo .= '<p> <img src='.$dados->picture.'>';
    $conteudo .= '<p>'.$dados->name.' - ';
    $conteudo .= format_date($dados->timestamp, 'custom', 'd/m/Y - H:i').'<br>';
    $conteudo .= $dados->texto.'<br>';
      if ($uid == $user->uid) {
        $conteudo .= l('Remover','recados/remover/'.$dados->rid).' - ';
        $conteudo .= l('Responder','recados/adicionar/'.$dados->aid).'</p>';
      }
```

```
  }
    $conteudo .= theme('pager', NULL, 5);
  return $conteudo;
}

/**
 * Gera formulário para envio de recado.
 *
 * @param   $uid  Identificação do usuário que receberá o recado
 * @return        Formulário de para criação de recado
 */
function recados_adicionar($formulario_estado, $uid) {

  if (!is_numeric($uid)) {
    return drupal_access_denied();
  }

  $formulario['recado'] = array(
    '#type' => 'fieldset',
  );
  $formulario['recado']['texto'] = array(
    '#title' => t('Recado'),
    '#type' => 'textarea',
    '#resizable' => FALSE,
    '#required' => TRUE,
  );
  $formulario['recado']['enviar'] = array(
    '#type' => 'submit',
```

```
    '#value' => t('Enviar recado')
  );
  $formulario['recado']['cancelar'] = array(
    '#value' => l('Cancelar','usuario/'.$uid.'/recados')
  );
  $formulario['uid'] = array(
    '#type' => 'value',
    '#value' => $uid
  );
  return $formulario;
}

/**
 * Valida o formulário da função "recados_formulario".
 *
 * @param   $formulario          Formulário a ser tratado
 * @param   &$formulario_estado  Array com as propriedades do formulário
 */
function recados_adicionar_validate($formulario, $formulario_estado) {
    print_r($formulario_estado);
    print('<br><br>');
    print_r($formulario);
    exit();
  global $user;
  $uid = $formulario_estado['values']['uid'];
  $valido = db_result(db_query("SELECT uid FROM {users} WHERE uid=%d", $uid));
  if ($uid < 2 || !$valido) {
    form_set_error('', t('Contato inválido!'));
```

```
  }
  if ($uid == $user->uid) {
    form_set_error('', t('Não é possível enviar recado para você mesmo!'));
  }
}

/**
 * Processa o formulário da função "recados_adicionar".
 * Insere os dados na tabela de recados com as devidas informações.
 *
 * @param   $formulario          Formulário a ser tratado
 * @param   &$formulario_estado  Array com as propriedades do formulário
 */
function recados_adicionar_submit($formulario, &$formulario_estado) {
  global $user;
  $aid = $user->uid;
  $uid =  $formulario_estado['values']['uid'];
  $texto = $formulario_estado['values']['texto'];
  if (db_query("INSERT INTO {recados} (uid, aid, texto, timestamp)
    VALUES (%d, %d, '%s', %d)", $uid, $aid, $texto, time())) {
    drupal_set_message(t('Messagem enviada. Obrigado pela sua participação.'));
  }
  else {
     drupal_set_message(t('Houve algum problema no sistema e não foi possível enviar a
mensagem.'), 'error');
  }
  $formulario_estado['redirect'] = 'usuario/'.$uid.'/recados';
}
```

```
/*
 * Remove o recado solicitado.
 * Gera página para confirmação de remoção de recado.
 *
 * @param   $rid  Identificação do recado a ser removido
 * @return        Formulário de confirmação
 */
function recados_remover(&$formulario_estado, $rid) {
  if (!is_numeric($rid)) {
    return drupal_access_denied();
  }
  $formulario['rid'] = array('#type' => 'value', '#value' => $rid);

  return confirm_form(
    $formulario,
    t('Tem certeza que deseja remover o recado selecionado?'),
    'recados/',
    t('Após a remoção, não será possível recuperar a mensagem.'),
    t('Remover'), t('Cancelar'));

}

/**
 * Valida o formulário da função "recados_remover".
 *
 * @param   $formulario          Formulário a ser tratado
 * @param   &$formulario_estado  Array com as propriedades do formulário
```

```
 */

function recados_remover_validate($formulario, &$formulario_estado) {
  // Verifica se o recado existe e se o usuário é proprietário dele.
  global $user;
  $rid = $formulario_estado['values']['rid'];
  $uid = db_result(db_query("SELECT uid FROM {recados} WHERE rid=%d", $rid));
  if (!$rid || $uid != $user->uid) {
    form_set_error('', t('Você permissão para remover este recado!'));
  }
}

/**
 * Processa o formulário da função "recados_remover_confirma".
 * Seta status do do recado para "0", fazendo com que ele não apareça mais
 * para o usuário (exclusão lógica).
 *
 * @param   $formulario          Formulário a ser tratado
 * @param   &$formulario_estado  Array com as propriedades do formulário
 */
function recados_remover_submit($formulario, &$formulario_estado) {
  $rid = $formulario_estado['values']['rid'];

  if (db_query("UPDATE {recados} SET status=%d WHERE rid=%d", 0, $rid)) {
    drupal_set_message(t('Messagem removida. Obrigado pela sua participação.'));
  }
  else {
```

```
    drupal_set_message(t('Houve algum problema no sistema e não foi possível remover a mensagem.'), 'error');

  }

  $formulario_estado['redirect'] = 'recados';

}
```