UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ESTUDO DE CASO DO SISTEMA DE GESTÃO DE RECURSOS HUMANOS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA

Kimbow Ribeiro Clébicar

JUIZ DE FORA
DEZEMBRO, 2009

ESTUDO DE CASO DO SISTEMA DE GESTÃO DE RECURSOS HUMANOS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA

Kimbow Ribeiro Clébicar

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharel em Ciência da Computação

Orientador: Ely Edison da Silva Matos

JUIZ DE FORA
DEZEMBRO, 2009

ESTUDO DE CASO DO SISTEMA DE GESTÃO DE RECURSOS HUMANOS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA

Kimbow Ribeiro Clébicar

Monografia submetida ao corpo docente do Departamento de Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora, como parte integrante dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Aprovada pela banca constituída pelos seguintes professores:

_____

Prof. Ely Edison da Silva Matos – orientador
M. Sc. Modelagem Computacional, UFJF

_____

Prof. Luiz Felipe Carvalho Mendes
M. Sc. Modelagem Computacional, UFJF

_____

Prof. Jesuliana Nascimento Ulysses
M. Sc. em Computação, UFF

Juiz de Fora, MG
Dezembro de 2009

AGRADECIMENTO

A Deus,
pelo dom da vida.

Aos meus pais,
pela formação de meu caráter.

Ao Professor Ely Edison da Silva Matos
pelo apoio e orientação.

Aos docentes,
pelos ensinamentos.

Aos amigos pelo suporte.

“É tolo tentar responder uma questão que você não entende. É triste ter que trabalhar para um fim que você não deseja. Coisas tristes e tolas como estas freqüentemente acontecem, dentro e fora da escola, mas o professor deve evitar que ocorram em classe. O estudante deve entender o problema. Mas não basta que ele o entenda. É necessário que ele deseje a sua solução.”

G. Polya em “How to Solve it”

RESUMO

Este trabalho apresenta aspectos de desenvolvimento de sistemas de informações voltados para web, através do estudo de tecnologias como Apache, PHP, XML, Javascript, CSS e Ajax que costumam fazer parte do pacote de utilidades em um nível mais baixo de desenvolvimento. O modelo de caso de uso é apresentado, bem como o seu diagrama, pois estes são pontos de ancoragem dos desenvolvedores para execução das tarefas de implementação. São abortados temas como o paradigma da orientação a objetos, padrões de projeto, modelos de dados e controle de versão, que são temas diretamente relacionada ao assunto e ajudam a ilustrar e compreender melhor certos aspectos do desenvolvimento. O framework de desenvolvimento de aplicações para web, MIOLO, é estudado quanto as suas características, e funcionamento das camadas, levando em consideração a sua aplicabilidade ao domínio de desenvolvimento, e a capacidade de afetar diretamente a qualidade do resultado final do sistema. Este trabalho também apresenta algumas funcionalidades do Sistema Integrado de Gestão Acadêmica na parte de Recursos Humanos da Universidade Federal de Juiz de Fora como, por exemplo, a geração de relatórios, gerência de dados pessoais e funcionais, controle de férias e licenças e afastamentos.

Palavras-chave:
Recursos Humanos, *framework* MIOLO, SIGA, UFJF, sistemas de informação e desenvolvimento web.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE REDUÇÕES

CATMAT	Catálogo de Materiais (BRASIL, 2000)
CATSER	Catálogo de Serviços (BRASIL, 2000)
CGCO	Centro de Gestão do Conhecimento Organizacional
CSS	*Cascading Style Sheets*
DAO	*Data Access Object*
DARPA	*Defense Advanced Reserch Projects Agency*
DCU	Diagrama de Casos de Uso
DER	Diagrama de Entidade-Relacionamento
DTD	*Document Type Definition*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IFES	Instituições Federais de Ensino Superior
LDAP	*Lightweight Directory Access Protocol*
MCU	Modelo de Casos de Uso
MVC	*Model-view-controller*
NCSA	*Nacional Center for Computer Aplications*
OMG	*Object Manangement Group*
PHP	PHP: *Hypertext Preprocessor*
PRORH	Pró Reitoria de Recurso Humanos
RH	Recursos Humanos
SETEC	Secretaria de Educação Tecnológica
SGBD	Sistema Gerenciador de Banco de Dados
SIAFI	Sistema Integrado de Administração Financeira
SIASG	Sistema Integrado de Administração de Serviços Gerais
SIGA	Sistema Integrado de Gestão Acadêmica
SIGA-RH	Módulo de RH do SIGA
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)
SSL	*Secure Sockets Layer*
UFJF	Universidade Federal de Juiz de Fora
UML	*Unified Modelling Language*
W3C	*World Wide Web Consortium*
WWW	*World Wide Web*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

1.1 APRESENTAÇÃO GERAL

Este trabalho introduz tecnologias que são fundamentais no processo de construção de sistemas de informações voltados para web. Entre as tecnologias estão o servidor de aplicação Apache, a linguagem de programação PHP, a linguagem de marcação XML, as folhas de estilo (CSS) e a tecnologia AJAX.

Entre os temas levantados está a questão da modelagem através de casos de uso (um dos principais diagramas da UML), o paradigma de orientação a objetos que dá embasamento a toda filosofia de desenvolvimento, os padrões de projeto que conferem boas condições para atingir alto grau de qualidade no produto final, os modelos de dados que são fundamentais em qualquer criação de sistema e o controle de versão com a ferramenta de código aberto e uso gratuito Subversion, que possibilita o trabalho cooperativo de grandes equipes como é o caso do desenvolvimento do projeto Sistema Integrado de Gestão Acadêmica (SIGA) da Universidade Federal de Juiz de Fora (UFJF).

No que tange ao desenvolvimento do módulo do sistema de recursos humanos, integrado as outras áreas que compreendem o SIGA, é usada uma concepção modular proporcionada pelo *framework* MIOLO, que foi escolhido para sustentar o projeto, com base em argumentos apontados no texto.

O *framework* MIOLO, que é desenvolvido em camadas, apresenta características e funcionalidades importantes na criação de sistemas voltados para web. Discutiremos as camadas de apresentação, criação, integração e negócio deste, bem como suas funcionalidades e características.

O SIGA é um sistema de grande complexidade e será tratado em suas partes principais, que são seus módulos de Ensino, Administração e Biblioteca.

Este trabalho também procura mostrar o resultado do desenvolvimento de um sistema de informação, acessível via web, responsável por manter dados relativos à gestão de pessoal, proporcionando condições mais eficientes para execução de procedimentos administrativos e possibilitando planejamentos futuros baseando-se em levantamentos retirados desta base de dados integrada, na Universidade Federal de Juiz de Fora (UFJF) apresentando funcionalidades do módulo de Recursos Humanos integrado ao SIGA.

1.2 CARACTERIZAÇÃO DO OBJETO DE ESTUDO

A UFJF possui uma base de dados atualizada de seus alunos, docentes, técnicos administrativos em educação e empresas terceirizadas, que é acessada através de um sistema de informações disponível na web. Este sistema possui uma base de dados comum a outros subsistemas da instituição, e foi desenvolvido pela própria instituição utilizando filosofias e ferramentas de código livre (aberto). Este sistema tem como objetivo gestão das diversas áreas administrativas, inclusive gestão de pessoas, a qual será propósito fundamental da pesquisa.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Apresentar funcionalidades do sistema de informação responsável pela gestão de recursos humanos da Universidade Federal de Juiz de Fora (UFJF) o qual faz parte do Sistema Integrado de Gestão Acadêmica – SIGA.

1.3.2 Objetivo Específico

Pesquisar, produzir e apresentar material específico na área do desenvolvimento de sistemas da UFJF, a respeito dos processos, metodologias e tecnologias que compõem a elaboração, implementação e implantação do sistema integrado.

1.4 JUSTIFICATIVA E RELEVÂNCIA

Com a ampliação constante das tecnologias de comunicação e do uso dos meios digitais para armazenamento e difusão de informações, é importante que instituições de ensino como a UFJF estejam preparadas para acompanhar a evolução nessas áreas. Fazer uso inteligente das informações depende da boa manutenção dos dados e é fundamental para colaborar com a segurança e eficiência nas atividades administrativas que dão suporte ao ensino e aprendizado de milhares de jovens, e para isso é necessário pesquisa e conhecimento para produzir ferramentas que atendam efetivamente as necessidades do órgão.

1.5 ESTRUTURA DO TRABALHO

Este trabalho é constituído desta introdução, mais quatro capítulos, e referências bibliográficas.

No segundo capítulo é apresentada uma visão geral sobre desenvolvimento de sistemas, mais especificamente sistemas de informação voltados para web, suas tecnologias, conceitos, paradigmas e técnicas que servirão de suporte ao objetivo final do trabalho.

No terceiro capítulo é detalhada a ferramenta base para construção do sistema de informação em estudo. O framework MIOLO será abordado e sua arquitetura e alguns aspectos mais interessantes serão discutidos.

No quarto capítulo é apresentado o Sistema Integrado de Gestão Acadêmica (SIGA). Serão vistos também os diversos módulos que compõe esta aplicação, porém, o foco principal do capítulo será o módulo de recursos humanos. Serão apontadas as funcionalidades e características mais importantes.

O quinto e último capítulo apresenta as considerações gerais e indica sugestões para trabalhos futuros.

2 DESENVOLVIMENTO DE SISTEMAS WEB

As seções a seguir fazem uma rápida revisão das principais tecnologias utilizadas no desenvolvimento de sistemas de informações para web, além de abordar assuntos pertinentes ao contexto, como o paradigma OO, padrões de projeto e modelo de dados.

2.1 SISTEMAS DE INFORMAÇÃO

A tecnologia da informação surgiu com a finalidade de automatizar processos bem conhecidos, definidos e muitas vezes repetitivos, através da programação de computadores. Com a manipulação destes processos pode obter-se grande quantidade de dados que por sua vez são armazenados para consulta posteriores, formando um histórico que pode, ainda, ser reunido, contextualizado e analisado para gerar informações importantes usadas para tomada de decisões estratégicas pelas instituições.

Com o crescente potencial de armazenamento e o grande volume de dados gerados pelas atividades administrativas dos diversos tipos de instituições, houve necessidade de organização e manutenção do conjunto de informações e com isso o aparecimento dos primeiros sistemas destinados a tal propósito, conhecido com sistemas de processamento transacional (LAUDON, 2004).

De acordo com BEZERRA (2007) “O desenvolvimento de sistema de informação é uma tarefa das mais complexas. Um dos seus componentes é denominado sistema de software. Esse componente compreende os módulos funcionais computadorizados que interagem entre si para proporcionar ao(s) usuário(s) do sistema a automatização de diversas tarefas”.

2.2 TECNOLOGIAS PARA SISTEMAS WEB

O Departamento de Defesa Americano, através da agência DARPA (*Defense Advanced Reserch Projects Agency*), no final da década de 60, percebeu a necessidade de compartilhamento desses dados estratégicos e a importância do compartilhamento das informações criando-se assim, as primeiras interconexões de redes. Dessa forma deu-se o início da construção da pilha de protocolos responsáveis pela comunicação entre as máquinas e seus softwares (RFC 793, 1981).

A WWW (*Word Wide Web*), ou simplesmente, web, interconecta dispositivos de diversos tipos e características que vão desde celulares, *palm tops* e GPS até consoles de vídeo games, receptores de TV via satélite passando até mesmo por geladeiras residenciais. Essa infinidade de possibilidades de interconexão resultou, antes de tudo, em grande mobilidade e versatilidade no acesso e consulta a dados. Trouxe conforto na realização de tarefas e rotinas, que antes exigiam deslocamentos dos indivíduos e agora se faz de modo remoto.

Para efeito de exemplo deste avanço, pode-se citar que há alguns anos as listas dos aprovados nos vestibulares das faculdades e universidades eram divulgadas em local físico aberto ao público. No caso da UFJF, este local era a área central da universidade, onde hoje está localizado o prédio da Reitoria. Neste local eram afixados dezenas de folhas de papel impressas, lado a lado, contendo em ordem de colocação o nome e a pontuação dos candidatos separados por cursos. Hoje não só o resultado, mas também o processo de inscrição é feito em sua totalidade por meio digital, através da web, bastando o candidato acessar, de qualquer cidade onde se encontre, o *site* da universidade (www.vestibular.ufjf.br).

2.2.1 Servidor web *Apache*

Segundo MARCELO (2005), a história do *Apache* inicia-se em 1995 através da empresa NCSA (*Nacional Center for Computer Aplications*). Nesta época o *Apache* era (e ainda hoje continua a ser) o servidor mais popular. Ele era conhecido como *NCSA Web Server*.

Após o desinteresse da empresa pela manutenção do projeto, um grupo de desenvolvedores começou a criar uma série de *patches*[1] para o servidor que acabou ganhando o nome de *Apache* devido a trocadilho com o termo *Apatchy*, devido à aplicação das diversas funcionalidades desenvolvidas pelo grupo que mais tarde ficou conhecido como *Apache Fundation* (MARCELO, 2005).

Além do fato do servidor *Apache* possuir código aberto e ser gratuito, com distribuição sob regras da licença GNU (*General Public License*)[2], outras principais vantagens no uso do *Apache* podem ser citadas (MARCELO, 2005):

- Suporte a HTTP[3] 1.1;
- Suporte a SSL[4];
- Suporte a PHP;

[1] Pacotes de melhoria e expansão aplicados a programas de computadores
[2] Licença Pública Geral
[3] Protocolo de comunicação da camada de aplicação, segundo modelo OSI, utilizado para sistemas de informação
[4] Protocolo criptográfico que confere segurança de comunicação pela internet

- Logs customizáveis;
- Configuração rápida e simples.

2.2.2 PHP: *Hypertext Preprocessor*

Afirma LALLI (2008) apud TIM (2005), que a linguagem PHP surgiu na mesma época da linguagem Ruby[5], em 1994, como acrônimo, ou sigla, para *Personal Home Page Tools,* era um conjunto de ferramentas Perl[6] e C[7] feitas por Rasmus Lerdorf. Em 1995 surgiu a primeira versão, se tornando popular. Em abril de 1996 o PHP 2 foi lançado. Por uma votação na comunidade, o nome foi alterado, para PHP: *Hypertext Preprocessor* na versão 3. “Em sua terceira versão havia mais de 100 mil *sites* que o utilizavam de alguma forma e, em apenas um ano, a linguagem alcançou a marca de um milhão de domínios”. A comunidade Apache, por volta do ano 2000, disponibilizou um módulo oficial para versão 4 do PHP. De julho de 2004 em diante a versão 5 tem sido usada.

PHP é uma linguagem de programação de computadores com código aberto. Essa linguagem é interpretada por servidores de conteúdo web, como por exemplo, o *Apache HTTP Server*, apresentado na seção 2.2.1, largamente utilizado pela comunidade especializada, conforme pesquisa realizada pela *NetCraft*[8], como podemos observar na Figura 2.1.

Uma das principais características do servidor web *Apache* é possuir suporte a linguagem PHP (SILVA, 2007), logo, o uso combinado do servidor web *Apache* e a linguagem PHP trazem grande benefício, devido ao tamanho da comunidade que colabora com os projetos.

[5] Ruby é uma linguagem de programação orientada a objetos de tipagem dinâmica criada por Yukihiro Matsumoto em dezembro de 1994 com objetivo de ser mais poderosa que Perl e mais orientada a objetos que Pyton (LALLI, 2008).

[6] Perl é uma linguagem de programação de missão crítica com destaque em desenvolvimento de aplicações web .

[7] C é uma linguagem de programação desenvolvida nos anos 70.

[8] *NetCraft* é um serviço de internet de uma empresa inglesa que tem como um dos objetivos fornecer dados e pesquisas sobre diversos aspectos da internet. A *NetCraft* explora a internet desde 1995 (NETCRAFT, 2009).

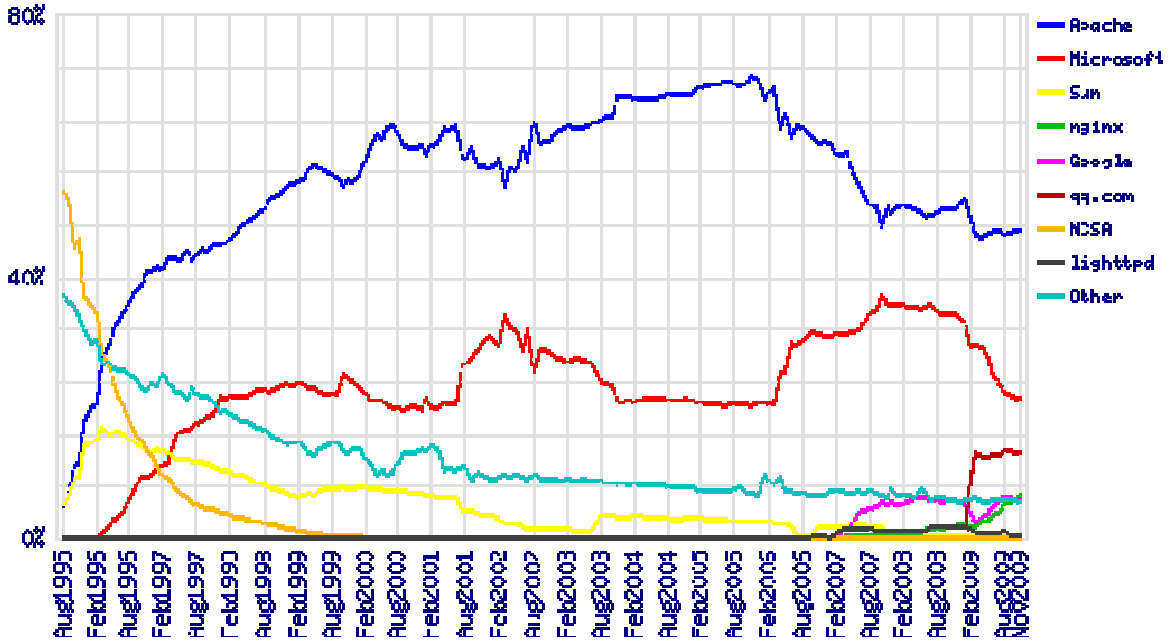


Figura 2.1 - Divisão de mercado entre grandes servidores: 1995 a 2009 (NETCRAFT, 2009)

2.2.3 XML

A Linguagem de Marcação Extensível (e*Xtensible Markup Language* - XML) foi criada em 1996 e tem como base a Linguagem de Marcação Padrão Generalizada (*Standard Generalized Markup Langage* – SGML). A criação de uma linguagem extensível, de fácil leitura e compreensão por seres humanos, que pudesse ser usada em conjunto com outras linguagens de marcação, como por exemplo a Linguagem de Marcação para Hipertexto (*Hypertext Markup Language* – HTML), também pudessem ser lida por softwares e fosse orientada para distribuição ampla na Internet, foi um grande desafio. Em 1998, XML alcançou o status de recomendação pela W3C (*World Wide Web Consortium*) e foi amplamente comentada pela comunidade especializada (TITTEL, 2002).

O que torna a XML especial é o fato de que a linguagem é extensível, escalonável e adaptável. Segundo TITTEL (2002), XML é capaz de tornar-se qualquer coisa que um documento necessite que ela seja para distribuir informações na web ou entre aplicativos (software) – sem restrições e limitações de HTML.

Com a combinação de definições de tipo de documento (*Document Type Definition* – DTD) as marcações da XML podem ser definidas para atender uma necessidade específica e permitir a troca de informações por uma linguagem que é facilmente compreendida por seres humanos e por programas de computadores. Ainda

segundo TITTEL (2002), “XML promete satisfazer às necessidades de uma Web em constante expansão, ao ir diretamente de encontro às limitações e fraquezas de HTML”.

2.2.4 Javascript

De acordo com a abordagem de TEIXEIRA (2009), Javascript é uma linguagem de programação de uso geral similar a família das linguagens C. Assim como PHP, Javascript é fracamente tipada, ou seja, suas variáveis que não são declaradas com tipos específicos. Durante a execução do *script*, diferentes tipos podem ser atribuídos à mesma variável em tempos de interpretação distintos, assumindo uma natureza dinâmica.

A linguagem Javascript é interpretada por navegadores web (*browsers*), sendo assim não é gerado código executável por meio de compilação de código fonte. Quando usada em páginas web, o servidor envia os códigos ao cliente (navegador web) que se encarrega de interpretar linha por linha os algoritmos programados (TEIXEIRA, 2009).

Segundo TEIXIERA (2009), “por ser de uso geral a linguagem pode ser usada para a maioria dos algoritmos e tarefas de programação”. A linguagem Javascrip possui suporte nativo a tipos numéricos, cadeia de caracteres, data e hora, expressões regulares, funções matemáticas e geração de números aleatórios. Também é possível definir objetos estruturados, o que auxilia no desenvolvimento de códigos mais complexos”.

A Figura 2.2 apresenta um exemplo de código em linguagem Javascript.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
2  <html>
3     <head>
4        <script type="">
5           function Inicializar()
6           {
7              window.alert("Inicializando a página HTML.");
8           }
9        </script>
10       <title>Exemplo de uso Javascrip em documento HTML</title>
11    </head>
12    <body onLoad="Inicializar()">
13    </body>
14 </html>
```

Figura 2.2 - Exemplo de código Javascript em documento HTML.

A Figura 2.3 exemplifica o uso de Javascript em um documento HTML sendo executado pelo navegador web Mozilla Firefox.

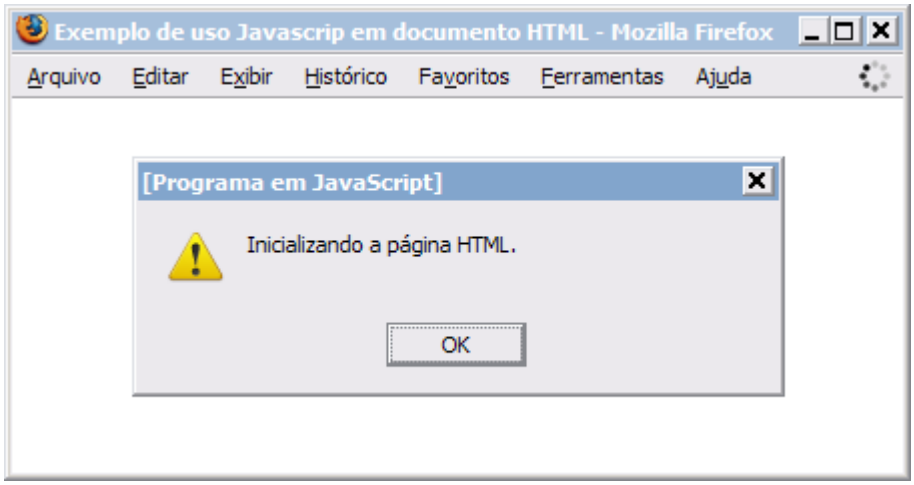


Figura 2.3 - Resultado da execução do Javascript da figura anterior

2.2.5 *Cascading Style Sheets* (CSS)

Folhas de Estilo em Cascata (CSS) permitem que o desenvolvedor, indique ao navegador (*browser*) as propriedades de estilo a serem aplicadas aos componentes de documentos HTML ou XML para apresentação do conteúdo. HTML e XML são dirigidos principalmente ao conteúdo e incluem poucas opções para marcação específica de definição de estilo de apresentação. CSS é um padrão para apresentação. Enquanto a linguagem de marcação HTML define estrutura de um documento, CSS define a forma de exibição do conteúdo dos elementos HTML. Desde a CSS2 em 1998, a W3C já estabeleceu como *status* de recomendação de uso para o propósito, as folhas de estilo. A primeira versão foi definida em 1996 e ainda hoje a W3C[9] continua a especificação do padrão. (TITTEL, 2002) e (TEIXEIRA, 2009 apud W3C, 1999).

Alguns dos benefícios práticos do bom uso de CSS envolvem a redução do tempo de design e desenvolvimento, diminuição do tempo de espera para o carregamento da página, remoção de elementos de apresentação da linguagem HTML e aumento da interoperabilidade seguindo os padrões web. (TEIXEIRA, 2009 apud REIS, 2007).

As definições CSS podem ser localizadas em três partes da página (TITTEL, 2002):

1. Em um documento separado fora de todos os documentos HTML;

[9] Consórcio formado por instituições comerciais e educacionais, com o objetivo de definir padrões para as áreas relacionadas à web.

2. No cabeçalho de um documento HTML;
3. Dentro de uma *tag* de HTML.

Segundo TITTEL (2002), dependendo de onde as definições CSS estejam, elas recebem terminologias específicas que, respectivamente, são externo, incorporado e *Inline*.

O método externo permite a criação de uma biblioteca de estilos que pode ser referenciado por diversas páginas HTML, enquanto o incorporado especifica regras que afetam somente á página que possui o cabeçalho com as especificações. E por último as regras definidas "*Inline"* têm seu escopo máximo de alcance somente a *tag* que faz uso da especificação. (TITTEL, 2002).

Para definir as regras de estilo externamente à página HTML que será exibida deve-se criar um arquivo com extensão ".css" e no código da página HTML devemos indicar o uso deste arquivo.

Arquivo contendo as descrições das regras de estilo, por exemplo: "Meu arquivo.css" é apresentado na Figura 2.4:

```
H1 {font-family: 'Comic Sans MS';
    font-size: 36pt;
    color: blue}
P  {font-family: 'Courier';
    margin-left: 0.5in}
```

Figura 2.4 - Definição de um arquivo CSS

A Figura 2.5 apresenta com exemplo um arquivo contendo descrição do conteúdo para documentos HTML a ser interpretado e exibido em um navegador web (browser):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
   <head>
      <link rel="STYLESHEET" href="./Meu Arquivo.css" type="text/css"/>
      <title>Exemplo de uso de folha de estilo externo!</title>
   </head>
   <body>
     <h1>Exemplo título H1</h1>
     <p>Parágrafo definido pela regra P.</p>
   </body>
</html>
```

Figura 2.5 - Documento HTML com link para arquivo CSS externo

O resultado final, exibido pelo navegador web, para aplicação com folha de estilo em cascata é apresentado pela Figura 2.6.

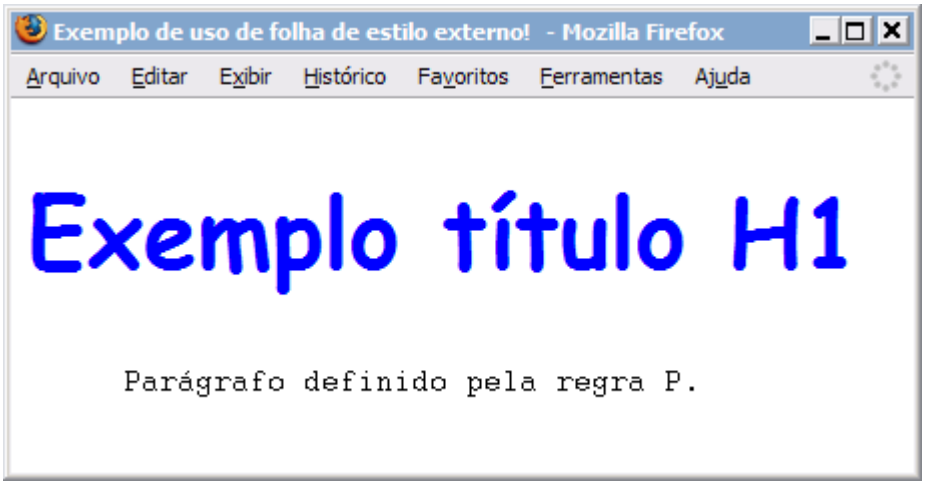


Figura 2.6 - Resultado do uso de CSS externo ao HTML.

2.2.6 AJAX

GARRET (2005), apud TEIXEIRA (2009) observa que o termo Ajax (*Asynchornous* Javascript + XML) se refere a um conjunto de tecnologias combinadas de uma nova maneira para o desenvolvimento de aplicações web. O autor da criação, Jesse James Garret, em 2005, denominou de Ajax o uso das tecnologias XML e Javascript de maneira assíncrona, ou seja, tecnologias já existentes que trabalhando em conjunto aproximam se do modelo desktop na dinâmica e interatividade das aplicações.

Algumas das tecnologias que compõe Ajax são: XHTML (*eXtensible Hypertext Markup Language* ou em português, Linguagem de Marcação de Hipertexto extensível) e CSS (*Cascading Style Sheets* ou em português, Folha de Estilos em Cascata) para apresentação baseada em padrões web. DOM (*Document Object Model*) para exibição e interação dinâmica entre cliente e servidor, XML e XSLT (*eXtensible Stylesheet Language Transformations*) para troca e manipulação de dados, XMLHttpRequest4 para recuperação assíncrona de dados e a linguagem de programação Javascript, para unir todas essas tecnologias (TEIXEIRA, 2009 apud GARRET, 2005).

Aplicações web baseadas em Ajax apresentam um acréscimo de camada entre o cliente e o servidor em comparação ao modelo tradicional conforme ilustra a Figura 2.7. Ao invés de carregar uma página no início de uma sessão o navegador carrega a Ajax-

engine, que passa então a ser responsável tanto pela interface da aplicação quanto pela comunicação com o servidor a partir do comportamento do usuário.

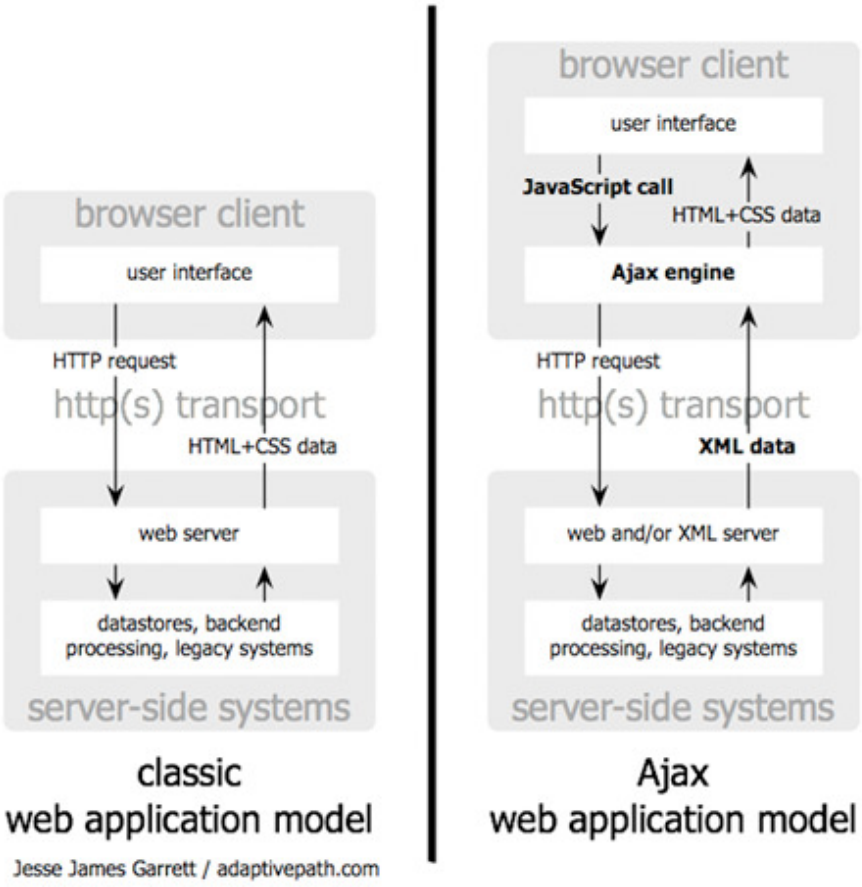


Figura 2.7 - Comparação entre modelos tradicional e Ajax *engine* (GARRET, 2005)

2.3 MODELAGENS UML

Segundo BEZERRA (2007), "a modelagem de sistemas de software consiste na utilização de notações gráficas e textuais com o objetivo de construir modelos que representam as partes essenciais de um sistema, considerando-se várias perspectivas diferentes e complementares."

Conforme colocado por BEZERRA (2007), nas décadas de 1950 e 1960 os sistemas de software eram mais simples e por conseqüência as técnicas de modelagem eram simples também. O desenvolvimento seguia o principio *ad hoc e* os modelos usados eram fluxogramas e diagramas de módulos.

Com um maior avanço computacional, e maior acessibilidade aos computadores, houve grande expansão comercial e os sistemas começavam a ficar mais complexos,

com isso, modelos mais robustos foram propostos. A programação estruturada e a análise e o projeto estruturado foram técnicas usadas durante a década de 1970 e na primeira metade da década de 1980 a análise estruturada se consolidava com a necessidade de criação de interfaces homem-máquina mais sofisticadas (BEZERRA, 2007).

O paradigma de orientação a objetos surge em resposta às dificuldades encontradas na aplicação de análise estruturada a determinados domínios de aplicação no início da década de 1990. No final desta mesma década os conceitos de padrões de projeto, framework, componentes e qualidade começam a ganhar espaço e a UML surge nessa época.

A UML foi aprovada pelo OMG[10] em 1997. Após aprovada, teve grande aceitação da comunidade de desenvolvedores de sistema. A UML ainda continua em desenvolvimento e conta com colaboração da área comercial. Atualmente a especificação se encontra na versão 2.0 (BEZERRA, 2007)

"UML (*Unified Modelling Language*) é uma linguagem para especificar, visualizar, documentar e construir um sistema de informação e pode ser utilizada em todas as etapas de desenvolvimento deste." (FURTADO, 2002)

Os elementos gráficos da UML possuem sintaxe e semântica. A sintaxe corresponde a forma como o elemento deve ser desenhado. A semântica define o significado e a utilização dos objetos. Tanto a sintaxe quanto a semântica são extensíveis tornando a linguagem adaptável aos projetos de desenvolvimento. A UML é independente de linguagem de programação e de metodologia de desenvolvimento (BEZERRA, 2007).

BEZERRA (2007) observa que "um processo de desenvolvimento que utilize a UML como linguagem de suporte à modelagem envolve a criação de diversos documentos. Esses documentos podem ser textuais ou gráficos. Na terminologia UML, esses documentos são denominados artefatos de software, ou simplesmente artefatos".

Ainda segundo BEZERRA (2007), os treze diagramas da UML estão divididos em comportamentais e estruturais. O grande número de diagramas é necessário para produzir diversas perspectivas do modelo do sistema. Os diagramas estruturais são: Diagrama de Objeto, Classes, Pacotes, Estrutura Composta (UML 2.0), Componentes e Implantação. Os diagramas comportamentais são: Diagrama de Atividades, Casos de Uso, Transições de Estado, Seqüência, Temporização (UML 2.0), Colaboração e Visão geral de Interação (UML 2.0).

[10] *Object Manangement Group*. Consórcio internacional de empresas que define padrões na área de orientação a objetos.

2.3.1 Modelo de Casos de Uso

O modelo de casos de uso (MCU) representa as funcionalidades que são observadas pelos usuários do sistema, ou seja, é uma visão externa do sistema e representa os possíveis usos deste. Cada um desses usos está relacionado a uma funcionalidade prevista para o sistema. Os usuários também fazem parte do diagrama e por definição, um caso de uso é definição da seqüência de interações entre o sistema e os agentes externos (usuários, por exemplo). Por essa visão o comportamento do desenvolvedor é direcionado às necessidades do agente externo. Um caso de uso é o relato de uma funcionalidade do sistema, porém, não revela como é o comportamento do sistema, abstraindo detalhes que não são importantes no contexto. Este modelo é considerado uma das ferramentas da UML usada para modelar o sistema de software que queremos desenvolver (BEZERRA, 2007).

Por ser um modelo de fácil compreensão, pode ser usado para dar uma visão geral dos requisitos do sistema facilitando a implementação na lógica de programação e auxiliando e agilizando a documentação.

Cada caso de uso é a descrição narrativa (textual) das interações entre o sistema e os agentes do ambiente externo, denominados atores. Os atores e os casos de uso são mutuamente dependentes para existência do diagrama. A UML não define o formato textual das descrições. O grau de detalhamento usado nas descrições pode variar (BEZERRA, 2007).

2.3.2 Diagrama de casos de uso (DCU)

"O DCU é um dos diagramas da UML e corresponde a uma visão externa de alto nível do sistema." (BEZERRA, 2007). Este diagrama tem como objetivo ilustrar quais elementos externos interage com quais funcionalidades do sistema. As relações entre os atores e as funcionalidades são representadas graficamente, utilizando-se a notação de uma figura de um boneco representando um ator (nem sempre um ser humano) com o nome logo abaixo e os casos de uso são representados por elipses com o nome do caso de uso abaixo ou dentro da mesma como podemos observar no exemplo da Figura 2.8.

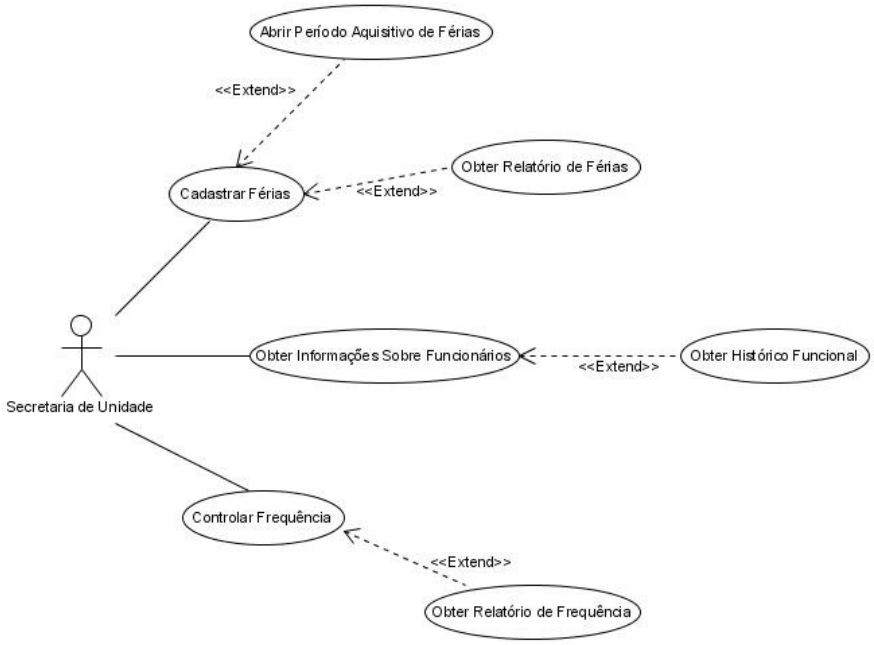


Figura 2.8 - Exemplo de Diagrama de Caso de Uso (DCU) do SIGA-RH

2.4 O PARADIGMA DA ORIENTAÇÃO A OBJETOS

Segundo BEZERRA (2007), atualmente, para o desenvolvimento de sistemas, é indispensável o uso do paradigma da orientação a objetos.

Paradigma é forma de abordar um problema que por um período de tempo mais ou menos longo que orienta o desenvolvimento de pesquisas em busca de soluções. No paradigma de orientação a objetos o sistema é um conjunto de objetos que realizam tarefas específicas e que interagem entre si para a realização de uma tarefa computacional (BEZERRA, 2007).

Segundo LEITE (2006), o conceito de classe possibilita o suporte para programação orientado a objetos, ou OOP, do inglês *Object-Oriented Programming*. As classes podem ser informalmente conceituadas como um conjunto de elementos que possuem as mesmas características. As instâncias de uma classe, são chamadas de objetos.

Na programação orientada a objetos observa-se alguns aspectos que ajudam a agilizar o desenvolvimento, como por exemplo, a herança e o polimorfismo. A herança nos permite a criação de novas classes que herdam atributos e comportamentos da classe pai ou superclasse sem perder a capacidade de implementação de novas funcionalidades. Com o polimorfismo pode-se trabalhar com as classes de maneira generalizada que, no decorrer do desenvolvimento, servirão de base para outras classes

mais especializadas. Ambas as tecnologias economizam tempo no desenvolvimento de programas complexos, principalmente por aumentar a capacidade de reutilização ao usar código já testado (DEITEL, 2001).

BEZERRA (2007) esclarece que:

> "É importante notar que classe é uma abstração das características de um grupo de coisas do mundo real. Na maioria das vezes, as coisas do mundo real são muito complexas para que todas as suas características sejam representadas em uma classe. Além disso, para fins de modelagem de um sistema, somente um subconjunto de características pode ser relevante. Portanto, uma classe representa uma abstração das características relevantes do mundo real".

2.5 PADRÕES DE PROJETO

Conforme o Guia *Sun* para *Enterprise Architects* citado por JUNIOR (2007), "um padrão de projeto é uma solução comum para um problema comum encontrado no desenvolvimento de software".

Padrões de projeto Segundo ALEXANDER (1977), apud GAMMA (2005) "cada padrão descreve um problema no nosso ambiente e o cerne da sua solução, de tal forma que você possa usar essa solução mais de um milhão de vezes, sem nunca fazê-lo da mesma maneira".

De acordo com GAMMA (2005), pelo fato do número de padrões de projeto ser grande, e variar tanto pelo nível de granularidade quanto pelo nível de abstração faz-se necessário um método de organização para facilitar o aprendizado e ajudar na descoberta de novos padrões. A Tabela 2.1 apresenta os padrões de projeto observando os propósitos de finalidade e escopo.

Os padrões de criação preocupam-se com a parte de criação dos objetos, os estruturais, com a composição de classes ou de objetos e os comportamentais a maneira pelas quais classes ou objetos interagem e distribuem responsabilidades (GAMMA, 2005).

		Propósito		
		Criação	**Estrutural**	**Comportamental**
Escopo	**Classe**	*Factory Method*	*Adapter*	*Interpreter* *Template Method*
	Objeto	*Abstract Factory* *Builder*	*Adapter* *Bridge* *Composite*	*Chain of Responsability* *Command* *Iterator*

		Prototype *Singleton*	*Decorator* *Façade* *Flyweight* *Proxy*	*Mediator* *Memento* *Observer* *State* *Strategy* *Visitor*

Tabela 2.1 - Organização dos padrões de projeto (GAMMA, 2005) [adaptado]

2.6 MODELOS DE DADOS

Segundo FERNANDES (2000), no passado a administração dos dados era dada apenas pela rigorosidade de padrões adotados na construção da estrutura de arquivos que os programas usavam para acessar os dados. Com a evolução da informática, houve a necessidade de cuidar melhor da administração dos dados e melhorar a forma de acesso a eles, devido ao crescente aumento do volume das informações armazenadas.

Ainda de acordo com FERNANDES (2000), na década de 80, técnicas de modelagem de dados começaram a sair das universidades e passaram a ser usadas no mercado e com isso os dados passaram a ter uma importância maior nas empresas, principalmente para tomadas de decisão.

Os dados passaram a ser “modelados (identificados na sua composição e na sua semântica), resguardados (na integridade, segurança e documentação) e disponibilizados (para o acesso, atualização e simulação)” (FERNANDES, 2000).

Os modelos de dados dividem-se em três grupos: modelos de dados físicos, modelos lógicos baseados em objetos e modelos lógicos baseados em registros. Entre os modelos lógicos o modelo de Entidade-Relacionamento é um dos mais usados, principalmente, devido a sua simplicidade e eficiência. Este modelo baseia-se na percepção real do mundo descrevendo objetos como entidades e o inter-relacionamento entre essas entidades (FERNANDES, 2000).

2.7 CONTROLE DE VERSÕES

O sistema de controle de versão permitir que as alterações feitas em um conjunto de códigos-fonte sejam controladas e documentadas através da criação de um histórico de modificações e também impõe um mecanismo de controle de acesso (MALHEIROS, 2005).

De acordo com COLLINS-SUSSMAN (2007), Subversion é um sistema de controle de versão gratuito e com código aberto que têm como objetivo de gerenciar arquivos e diretórios (pastas) e as mudanças ocorridas neles com o passar do tempo, possibilitando a recuperação de versões anteriores, ou examinar o histórico de mudança nos arquivos. Ainda segundo COLLINS-SUSSMAN (2007), o Subversion nasceu em 31 de agosto de 2001 após quatorze meses de codificação com o objetivo de melhorar o sistema controle de versão CVS[11].

MALHEIROS (2005), afirma que o Subversion é o provável substituto do CSV, em projetos de software livre, pois busca sanar a limitações desse, mantendo os princípios básicos. O Subversion é mantido pela CollabNet (MALHEIROS, 2005).

O Subversion pode ser operado através da rede, o que possibilita a distribuição de código em diferentes máquinas (COLLINS-SUSSMAN, 2007).

Como podemos observar na Figura 2.9, a arquitetura do Subversion se divide em duas interfaces que separam três camadas fundamentais do software.

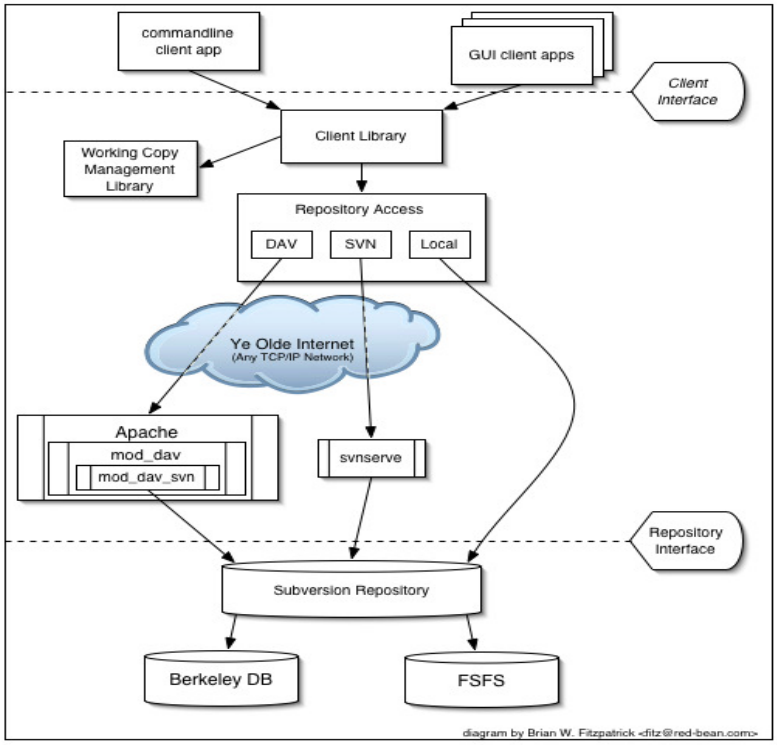


Figura 2.9 - Arquitetura do Subversion (COLLINS-SUSSMAN, 2007)

[11] O CVS, ou *Concurrent Version System* (Sistema de Versões Concorrentes) é um sistema de controle de versão que permite que se trabalhe com diversas versões de arquivos em um diretório

A primeira interface separa da camada de aplicação do cliente, que pode ser gráfica, com o Tortoise[12] no Windows, por exemplo, ou via linha de comando, com o uso do cliente svn (Subversion), da camada de transporte.

A camada de transporte possui uma verificação de acesso ao repositório por meio de sistemas de autenticação distintos. Um dos meios de autenticação pode ser baseado no uso de um servidor web Apache (através do módulo denominado “mod_dav_svn”). Outra forma de autenticação seria pelo próprio “svnserver”, um pequeno (leve) servidor que acompanha o Subversion. Quando o acesso é local, o próprio repositório pode encarregar-se da autenticação.

A terceira e última camada da arquitetura consiste propriamente no controle das versões do repositório que se baseia em FSFS[13] e Barkeley-DB[14].

O controle de versão é baseado em repositórios contendo código-fonte dos módulos de sistema, e basea-a se nos privilégios dos desenvolvedores para acesso de escrita e leitura. Na Figura 2.10 temos um exemplo esquemático do acesso a um repositório controlado pelo Subversion.

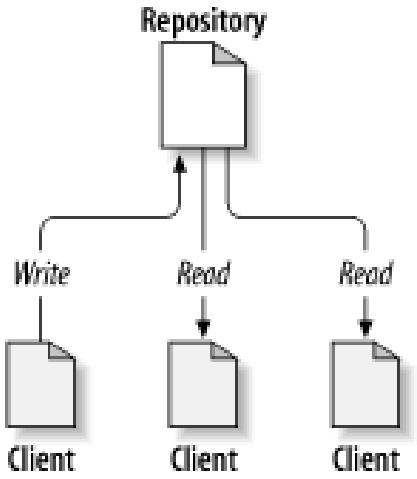


Figura 2.10 - Acesso ao repositório controlado pelo Subversion (COLLINS-SUSSMAN, 2007)

O sistema de controle de versão propiciado pelo Subversion atende a distribuição de arquivos de código fonte, arquivos de configuração e documentos indispensáveis ao processo de desenvolvimento, verificando a validade do acesso aos módulos do SIGA, e inclusive, ao MIOLO. Estes só podem ser baixados ou atualizados de acordo com os privilégios do desenvolvedor.

[12] Tortoise é um cliente Subversion para plataforma Windows para controle de versão (http://tortoisesvn.tigris.org/).
[13] FSFS é o sistema de arquivo do próprio do Subversion.
[14] Barkeley-DB é uma biblioteca de alto desempenho para banco de dados embarcados.

Em adição ao controle de verão dos arquivos e diretórios o Subversion se destaca por oferecer a integridade dos arquivos num processo de trabalho colaborativo com grande número de pessoas.

Quando, por exemplo, dois desenvolvedores trabalham em um mesmo arquivo, o Subversion bloqueia, a atualização do repositório, até que o segundo desenvolvedor atualize o conteúdo de seu arquivo, através do comando *svn updade,* conforme ilustra a Figura 2.11.

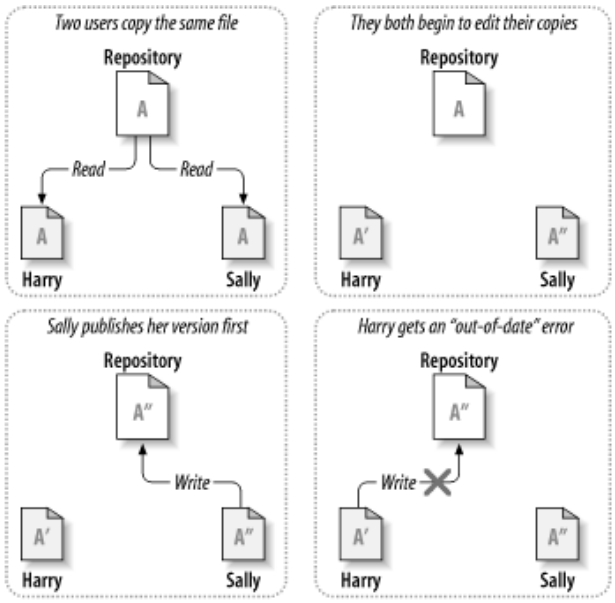


Figura 2.11 - Bloqueio de sobreposição

Quando o desenvolvedor faz a atualização do seu arquivo, o Subversion mescla as alterações feitas pelo primeiro desenvolvedor com as do segundo desenvolvedor, desde que não sejam conflitantes em relação ao repositório original, como podemos observa pelos esquemas da Figura 2.12.

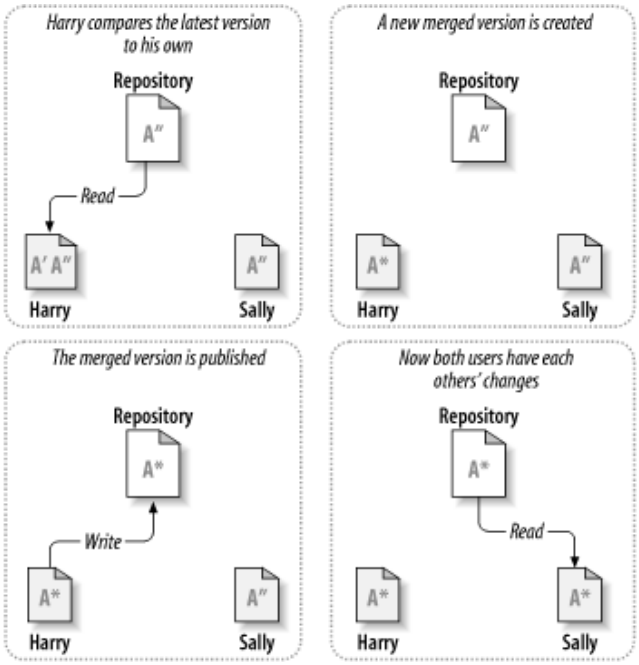


Figura 2.12 - Combinação de conteúdos alterados

No caso específico da UFJF, a cópia de trabalho do código fonte do projeto SIGA é armazenada remotamente, no servidor de desenvolvimento. Essas áreas são individualmente disponibilizadas para mapeamentos, através de um servidor Samba[15], assim, as estações de trabalho com sistema operacional Linux ou Windows podem acessar os arquivos de forma rápida e segura.

2.8 CONSIDERAÇÕES FINAIS

Este capítulo apresentou alguns recursos tecnológicos usados na implementação de projetos de sistemas de informação para web que interferem diretamente na filosofia e na forma como os recursos serão convergidos e utilizados para auxiliar o desenvolvimento.

[15] Samba é um software servidor criado por Andrew Tridgell usado para prover serviços de arquivo em plataforma Linux para plataformas Windows. http://www.samba.org/

3 FRAMEWORK MIOLO

Nas seções a seguir é apresentado o *framework* MIOLO[16] que serve de base para o desenvolvimento do projeto SIGA da UFJF.

3.1 FRAMEWORK

Um *framework* é um conjunto de classes cooperantes que constroem um projeto reutilizável para uma determinada categoria de software (DEUTSCH, 1989). Por exemplo, um *framework* pode ser orientado à construção de editores gráficos para diferentes domínios, tais como desenho artístico, composição musical e sistemas CAD para mecânica (VLISSIDES, 1990) e (JOHNSON R. E., 1992). Outro framework pode ajudar a construir compiladores para diferentes linguagens de programação e diferentes processadores (JOHNSON R., 1992). Um outro, ainda, pode ajudar a construir aplicações para modelagem financeira (BIRRER, 1993).

GAMMA *et. al*. explica que o *framework* dita a arquitetura da aplicação que será construída a partir dele, definindo a colaboração entre as classes e o controle do fluxo de dados. “Um *framework* é customizado para uma aplicação específica através da criação de subclasses específicas para a aplicação, derivadas das classes abstratas do framework” (GAMMA, 2005).

Um *framework* traz consigo decisões de projetos que serão comuns no domínio da aplicação e por este motivo a reutilização de projetos é freqüente. A reutilização de código não é tão é freqüente, mas pode existir mediante utilização de classes concretas diretamente (GAMMA, 2005).

Dessa forma nota-se uma inversão de controle entre a aplicação e o *software* a qual foi baseada, em relação a bibliotecas de rotina (*toolkit*). As bibliotecas são chamadas a partir do código da aplicação, reutilizando um trecho de código, enquanto no *framework* quem passa a ser acessada é a aplicação, ou seja, reutilização baseia-se no *framework* (GAMMA, 2005).

A adoção de um *framework* trás como conseqüência facilidade de criação e manutenção de novas aplicações principalmente pela repetição dos conceitos aplicados as estruturas básicas. Como subprodutos desta uniformização, as aplicações tendem a parecer mais consistentes para os usuários. Por outro lado, perdemos certa capacidade

[16] http://www.miolo.org.br

de customização já que determinados padrões foram estabelecidos na concepção do *framework* levando em consideração os aspectos do domínio da aplicação. (GAMMA, 2005).

É importante ressaltar que o *framework* seja construído de forma a manter a extensibilidade e flexibilidade, pois qualquer mudança na arquitetura do *framework* pode gerar grandes conseqüências em todas as aplicações, resultado da grande dependência da aplicação com o *framework*, ou seja, um fraco acoplamento[17] (GAMMA, 2005).

Os padrões de projeto ajudam a tornar a arquitetura do *framework* adaptável a diversas aplicações, sem necessidade de reconstrução (GAMMA, 2005). GAMMA et. al. relata que os *frameworks* estão se tornando cada vez mais comuns e importantes. Eles são a maneira pela qual os sistemas orientados a objetos conseguem a maior reutilização. Ainda segundo GAMMA et. al. a grande maioria dos projetos e dos códigos das aplicações virão dos *frameworks* que eles utilizarão ou será influenciada por ele.

3.2 MIOLO

Segundo GÄRTNER e MATOS (2008) o MIOLO é um *framework* voltado à criação de aplicações web, utiliza tecnologias comuns do meio, como Javascirpt, CSS, HTML, entre outros, é escrito predominantemente em PHP e utiliza conceito de POO (Programação Orientada a Objetos). Com ele, é possível gerar arquivos no formato HTML e PDF (em versões mais atuais é possível considerar a geração de outros tipos de formato de arquivo, como TXT e CSV).

A criação deste *framework* foi iniciada em 2001 na universidade UNIVATES[18] (Unidade Integrada Vale do Taquari de Ensino Superior) em Lajeado, no Rio Grande do Sul, contudo, atualmente encontra-se sob responsabilidade da cooperativa SOLIS[19] (MATOS, 2007).

O MIOLO favorece a reutilização por proporcionar criação de sistemas complexos divididos em módulos e com fácil integração entre os módulos. O MIOLO propicia diversas funcionalidades para desenvolvedores de sistemas, atuando como um núcleo para esses sistemas e define comportamentos de codificação para agilizar e simplificar o resultado final (GÄRTENER; MATOS, 2008).

[17] Acoplamento é o nível de interdependência entre os módulos de um programa de computador
[18] http://www.univates.br
[19] A Solis, Cooperativa de Soluções Livres, é uma cooperativa de serviços formada no início de 2003, com o apoio da Univates, Centro Universitário, em Lajeado, RS. Originada a partir de seu Centro de Processamento de Dados (CPD), é composta por um grupo de alunos, professores e ex-funcionários da Univates. A Solis implementa e desenvolve soluções tecnológicas livres para os mais variados setores da academia, indústria, comércio e serviços (http://www.solis.coop.br).

Vale ainda destacar que atualmente o MIOLO está sendo utilizado inclusive fora do Brasil, mais precisamente na Espanha, pela Agropic (*Aplicación Web de Producción Integrada y Certificación Agroalimentaria*), no projeto Rastre (AGROPIC, 2009).

O MIOLO é considerado modular, e baseado em componentes tendo diversas camadas em sua arquitetura (GÄRTNER; MATOS, 2008). Em MATOS (2009) é relatado que "cada módulo agrupa funcionalidades comuns a um certo domínio de aplicação".

Entre algumas das funcionalidades mais comuns que o *framework* MIOLO disponibiliza, são (MATOS, 2007):

- Controles de Interface com o usuário;
- Autenticação de usuários;
- Perfis de permissões de acesso;
- Camada de persistência de objetos.

Como o *framework* MIOLO utiliza PHP, apresentado na seção 2.2.2, a capacidade de atender especificidades é facilitada pelas inúmeras bibliotecas de classes disponíveis na comunidade de *software* livre.

É importante esclarecer o conceito de aplicação, no contexto do MIOLO, definido por MATOS (2009):

> "O framework MIOLO tem por objetivo a construção de sistemas de informação baseados em web, oferecendo a infra-estrutura necessária para que o desenvolver se preocupe apenas com o domínio da aplicação e não com os detalhes de implementação. Estes sistemas são construídos através do desenvolvimento de módulos[20]. O conjunto de módulos é chamado aplicação. Assim, de forma geral, cada instalação do framework está associada a uma única **aplicação**, composta por um ou vários módulos integrados."

Nos trabalhos de MATOS (2009) observamos que o framework MIOLO é dividido em camadas, que separam o código responsável pela apresentação, das regras de negócio, e das partes de recursos e integração, possuindo um rico conjunto de interface com o usuário, que pode ser estendido. O próprio framework é capaz de gerenciar sessão e estado. A geração do código HTML segue o padrão Tableless [21]. permitindo a criação de aplicações *cross-browser*[22]. O MIOLO possui entre seus mecanismos de

[20] Um módulo é um componente de uma aplicação. De forma geral, um módulo reflete um sub-domínio da aplicação, agregando as classes de negócio que estão fortemente relacionadas e provendo o fluxo de execução e a interface com o usuário para se trabalhar com tais classes.

[21] *Tableless* é uma forma de desenvolvimento de sites que não utiliza tabelas para disposição de conteúdo na página, pois defende que os códigos HTML deveriam ser usados para o propósito que foram criados, sendo que tabelas foram criadas para exibir dados tabulares. Para a disposição da página o recomendado seria usar CSS.

[22] *Cross-browser* refere-se à habilidade de uma aplicação suportar múltiplos navegadores que suportem as especificações do W3C

segurança, sistema de autenticação por LDAP[23] ou SGBD, controle de permissões (perfil de acesso) e geração de *log* nas transações.

Outras vantagens do uso do MIOLO como *framework* de apoio ao desenvolvimento de aplicações web, está na camada de acesso ao banco de dados através de persistência de objetos e uso da camada DAO[24] (*Data Access Objetcts*) para abstração de banco de dados, além da possibilidade de customização de temas já existentes ou criação de uma nova aparência (estética) para o sistema (MATOS, 2009).

A geração de arquivos no consagrado formato PDF é realizada por classes encapsuladas no MIOLO facilitando a apresentação de relatórios (MATOS, 2009).

O MIOLO, por ter seu código aberto, tem grande potencial de crescimento devido a colaborações externas. A UFJF, através do CGCO vem colaborando de forma contínua e acentuada no projeto.

3.3 ARQUITETURA EM CAMADAS

Conforme explicado anteriormente (seção 3.2.1) o MIOLO está organizado em camadas, como pode ser observado pela Figura 3.1. Estas camadas se dividem basicamente em componentes de interface com o usuário (camada de apresentação), camada de negócios (*business*) e camada de acesso a dados e/ou recursos (*data access*) seguindo o padrão MVC[25] (*Model–view–controller*), contudo, o MIOLO é flexível para permitir a não adoção do padrão (MATOS, 2009).

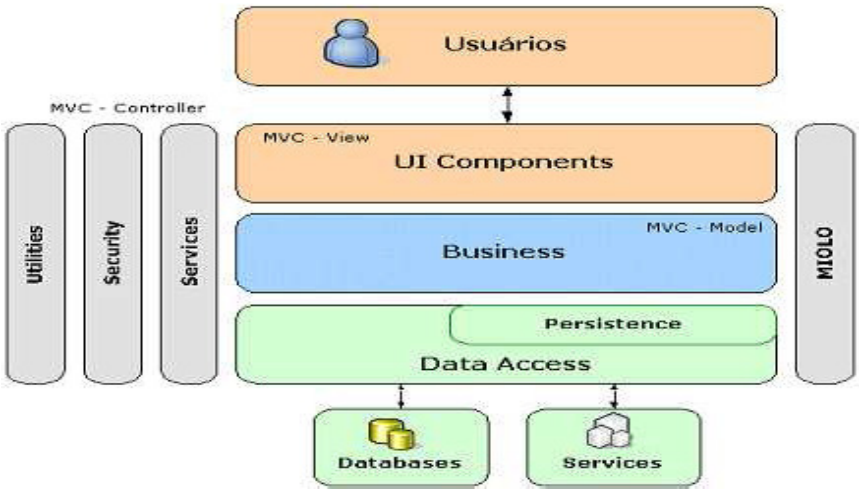


Figura 3.1 - Camadas do *framework* MIOLO (MATOS, 2009)

[23] LDAP é um protocolo para atualizar e pesquisar diretórios rodando sobre TCP/IP
[24] DAO é um padrão para persistência de dados
[25] MVC é um padrão arquitetural de separação de camadas no desenvolvimento de *softwares* que torna independente a parte de regras de negócio da apresentação

3.3.1 Camada de Apresentação

MATOS (2009) sustenta que a camada de apresentação é responsável por renderizar os componentes de interface com o usuário (controles do MIOLO) em HTML conforme os padrões adotados pelo framework e mencionados anteriormente neste trabalho, além de disponibilizar ao navegadores dos clientes os códigos Javascripts que serão utilizados.

É esta camada que dá aparência aos formulários codificados através de classes estendidas do *framework* pelos desenvolvedores, para interação com usuário do sistema. Interação que pode ser pela exibição de um relatório em formato PDF, solicitado pelo operador do sistema, visualização de um *grid* (grade) contendo informações ou controles, um *menu* de opções, ou controles mais simples para entrada de dados, simulando um formulário de papel, porém eletrônico.

Conforme explicado por GÄRTNER (2009), uma página HTML poderá apresentar vários formulários que são identificados por *tags* HTML (<form>) e exibidos em dentro de um elemento do tema ( ). Cada *form* é composto pelos controles cujos dados serão enviados em sua submissão.

O formulário principal da aplicação desenvolvida com MIOLO é o __mainForm, identificado por . Ele recebe a resposta do servidor, solicitada pela chamada Ajax, e servirá de base para a exibição dessa resposta, no elemento do tema correspondente (*grid, menu, top, content* etc.).

Como mencionado anteriormente o MIOLO disponibiliza para o desenvolvedor um rico conjunto de componentes visuais, comumente chamado de controles (*widgets*). Eles podem ser customizados, na aparência com o uso de temas e *templates* e possuem propriedades e eventos associados. O conjunto de controles fica armazenado em <miolo>/classes/ui/controls e cada um é definido em um arquivo próprio. Ver Anexo 1 – Árvore de controles do MIOLO. A Figura 3.2 contém controles de tratamento de data, texto, grades de exibição de informações com paginação, controles de seleção, entre outros.

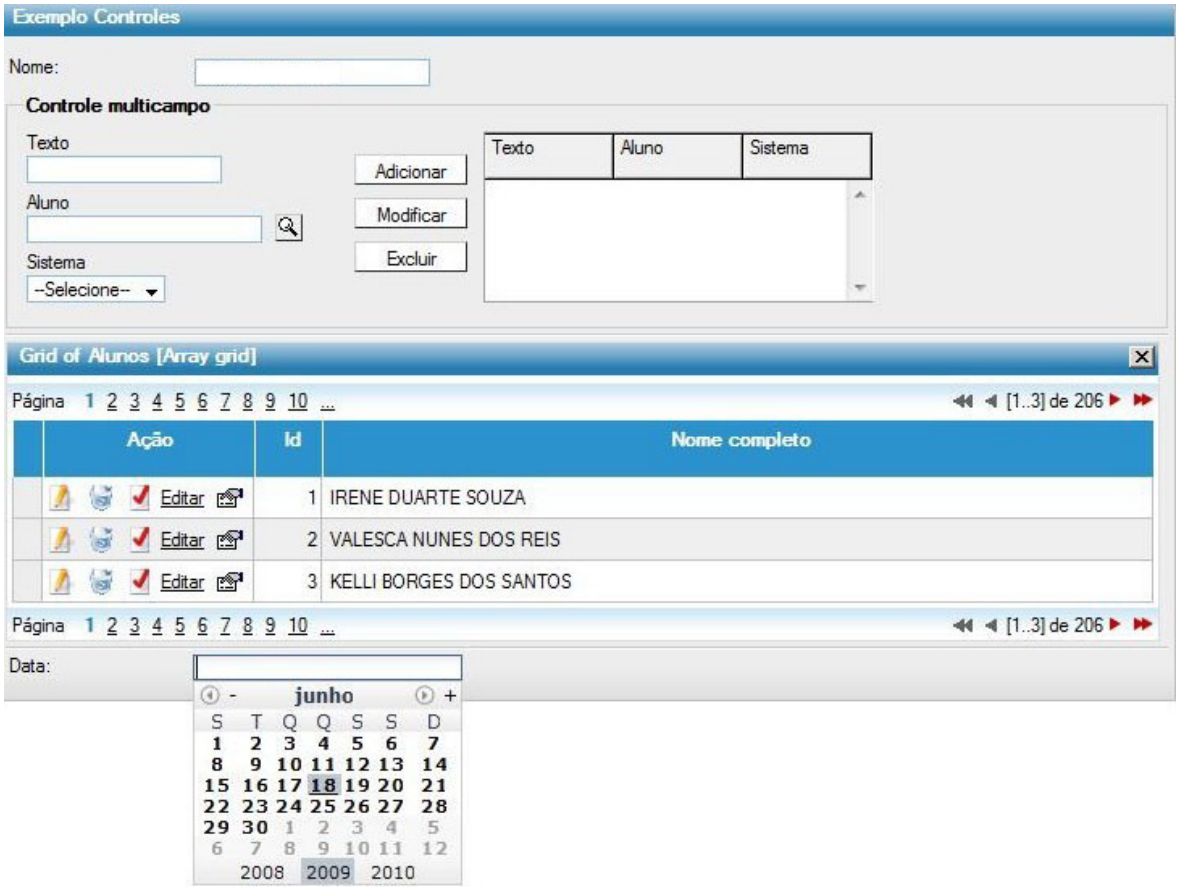


Figura 3.2 - Exemplo de formulário criado no MIOLO (TEIXEIRA, 2009) [adaptado]

De acordo com TEIXEIRA (2009) é importante atentar para o conceito de controles atômicos. Eles se caracterizam por serem exibidos diretamente por meio de uma tag HTML. Um controle não-atômico, como regra geral, é uma junção de controles atômicos e sua exibição é realizada com a execução dos respectivos métodos dos seus componentes.

É fundamental evitar colocar regras de formatação e códigos HTML em meio a codificação da lógica de operação dos controles, pois existe um local apropriado para essas formatações (configuração/renderização do tema). Com isso, não há fuga dos padrões de separação de camadas (MVC).

MIOLO reúne usa entre outras tecnologias, HTML, DHTML (*Dynamic HTML*), Javascript e CSS. O tema define a apresentação da página HTML que será exibida ao usuário pelo *browser*. Se a aplicação precisar de uma atualização estética ou funcional, mas na parte visual, basta o desenvolvedor projetar um novo tema e inseri-lo no diretório de apropriado do *framework* (<miolo *home*>/html/themes). Com o novo tema definido, basta configurar o arquivo de configuração miolo.conf (localizado em <miolo>/etc) para usar este novo tema. A Figura.

```
-----<diretório-base>(<miolo home>)
|
+--- html
|       +--- themes
|       |      +--- blue
|       |      | +--- images
|       |      | +--- templates
|       |      |           | +--- base.php
|       |      |           | +--- content.php
|       |      |           | +--- default.php
|       |      |           | +--- menu.php
|       |      |           | +--- navbar.php
|       |      |           | +--- window.php
|       |      | +--- blue.css
|       |      | +--- dojo.css
|       |      | +--- miolo.css
|       |      | +--- theme.class.php
```

O tema é constituído por elementos (título, barra de navegação, menus, área de conteúdo, barra de *status*), que são representados por controles da classe *MThemeElement*. Classes internas ao MIOLO definem e manipulam o tema, que é responsável por definir como os controles HTML serão exibidos e como a página será recortada e preenchida (MATTOS, 2009). As Figuras 3.3 e 3.4 mostram o recorte de uma tela e os elementos de um tema, respectivamente.

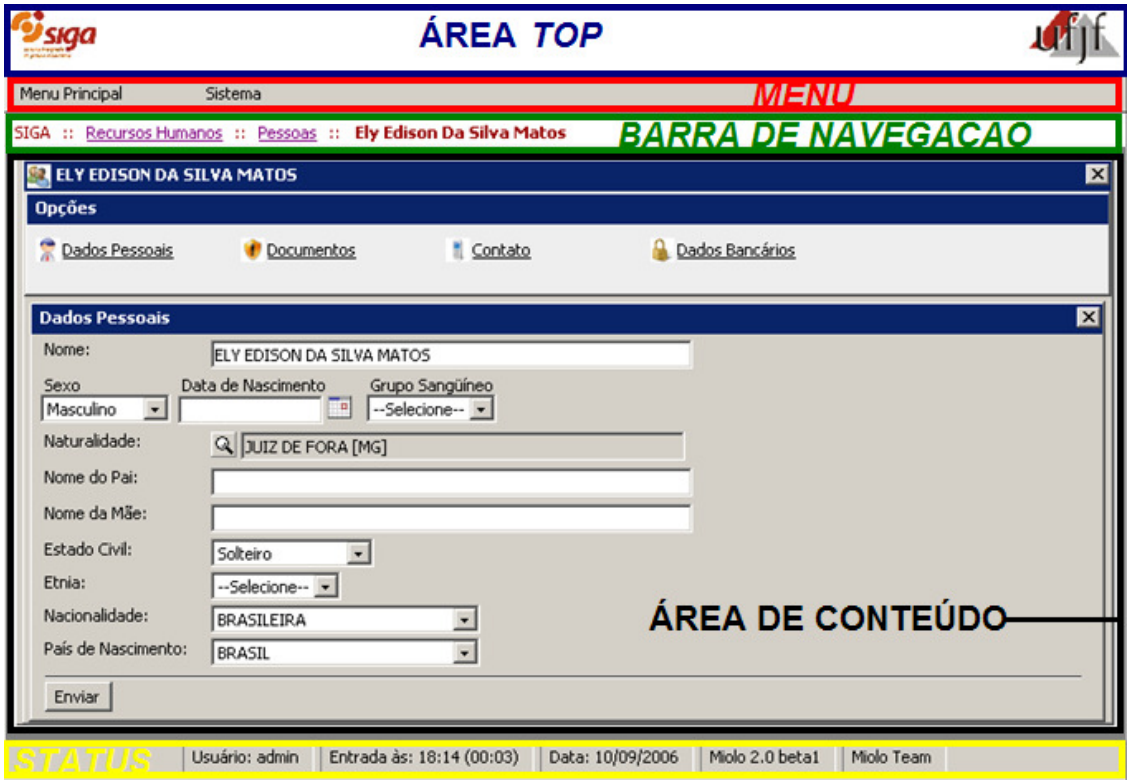


Figura 3.3 - Recorte da interface projetado no tema (MATOS, 2007) [adaptado]

Cada elemento do tema pode ser constituído por um ou mais controles do *framework* (*menu*, *image*, div, *form* etc.) e é renderizado como um controle HTML Div, com um atributo identificador. Quando uma parte do documento HTML não tem uma *tag* correspondente, utilizamos uma *tag* div para delimitá-la. Desta forma, usamos o HTML Div para indicar a área da página que receberá o código do elemento do tema com seus respectivos controles (TEIXEIRA, 2009).

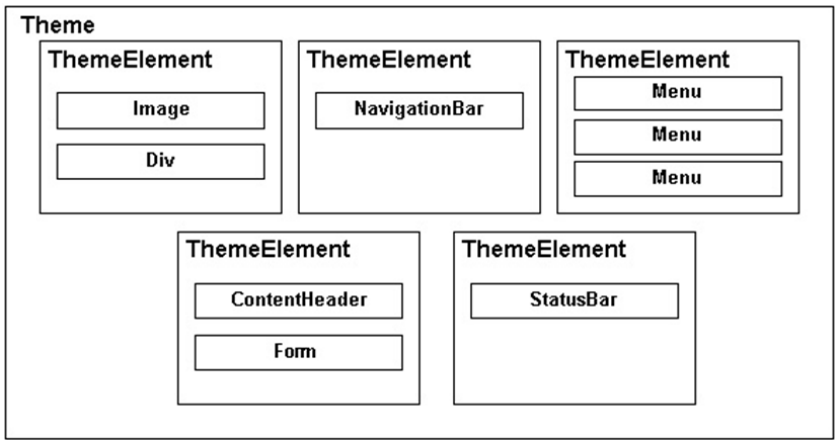


Figura 3.4 - Elementos de controle usados pela classe responsável pelo tema (MATOS, 2007)

3.3.2 Camada de Negócio

Ao desenvolver aplicações utilizando o MIOLO, dada as funcionalidades do *framework*, o desenvolvedor pode investir uma quantidade maior de horas do projeto analisando e construindo as regras que fazem parte do negócio, garantindo melhor conhecimento do domínio da aplicação imprimindo maior eficiência na implantação e qualidade no resultado final.

As regras de negócio, estão implementadas no código das classes de negócio, estes ficam gravados em arquivos dentro do módulo da aplicação, mais especificamente dentro do diretório de classes, seguindo a estrutura de organização apresentada no Anexo 3 - Árvore de controles do MIOLO.

Como exemplo, podemos imaginar que um sistema de gestão de recursos humanos faz uso de componentes como funcionário, férias, aposentadoria, etc. que remetem a criação de classes abstraindo-se a qualidade desses “objetos”.

Segundo GAMMA (2005), e GÄRTNER (2008) o nível de aprofundamento nos detalhes desses objetos, ou seja, a granularidade depende da análise do desenvolvedor dependendo do tamanho e complexidade do projeto.

GÄRTNER (2008) relata que estes objetos podem utilizar outros objetos do domínio de aplicação ou acessar o banco de dados por meio da camada de persistência ou camada DAO.

A camada de negócio (*Model)* do MVC, no caso do MIOLO, está ligada a camada de apresentação, através dos *grids*, *forms*, *menus*, etc. As classes de modelo de negócio, para exibirem um conteúdo desejado, quase sempre lançam mão do uso das classes de recursos e acesso ao SGBD, já que, a maioria dos sistemas usa banco de dados para armazenar suas informações.

Para acessar o conteúdo para exibição, é importante que o relacionamento entre a descrição dos dados no SGBD seja condizente com a implementação dos objetos da camada de negócio, e para isso é feito um mapeamento entre atributos das classe de modelo de negócio e os campos da tabelas no SGBD e assim como as tabelas se relacionam no SGBD, as classes se relacionam no sistema. As instruções contidas nesse mapeamento são descritas em XML e ficam armazenadas em arquivos dentro da pasta *map* sobre a pasta classes, dentro de um módulo (que faça uso de persistência).

As classes de domínio devem ser estendidas da classe *MBusiness* para herdar métodos como *save*, *delete*, *retrieve*, que partindo da definição de mapeamento, executam tarefas comuns como salvar, apagar e recuperar dados automaticamente, sendo totalmente transparente o acesso à fonte de dados.

GÄRTNER (2009) especifica que as classes devem ser declaradas usando a seguinte sintaxe:

```
class Business<modulo><classe> extends Mbusiness
```

Assim, em outros módulos, se necessário, é possível fazer uso destas classes, com isso temos o aumento do reuso de código.

Por exemplo, temos logo abaixo a declaração da classe funcionário do SIGA:

```
class BusinessRHFuncionário extends Mbusiness
```

3.3.3 Camada de Integração

A camada de integração implementa o padrão *Facade* onde encontramos a classe MIOLO, que representa o *framework* e expõe métodos que integram as diversas camadas (GÄRTNER, 2008).

A interação com o usuário é controlada pelos *handlers*, que , de modo geral, são os responsáveis por integrar a camada de apresentação com a camada de negócio. São os *handlers* que são acionados quando um usuário acessa um painel, ou executa algumas ações no sistema. Em resposta, o sistema pode, por exemplo, em virtude do perfil de acesso, pode direcionar o usuário a um formulário.

MATOS (2009) descreve os *handlers* como “Classes que representam a parte funcional da aplicação, criadas pelo desenvolvedor para fazer o tratamento dos dados enviados pelo cliente. Definem o fluxo de execução e implementam os casos de uso. Os *handlers* podem acessar a camada de negócios para obter/gravar dados e usam a camada UI para definir a saída para o cliente”.

Afirma MATOS (2009) que os *handlers* representam a camada controle do MVC e que as implementações devem ficar no diretório *handlers* do módulo.

A Figura 3.5 apresenta a camada de integração em destaque na arquitetura do *framework* MIOLO.

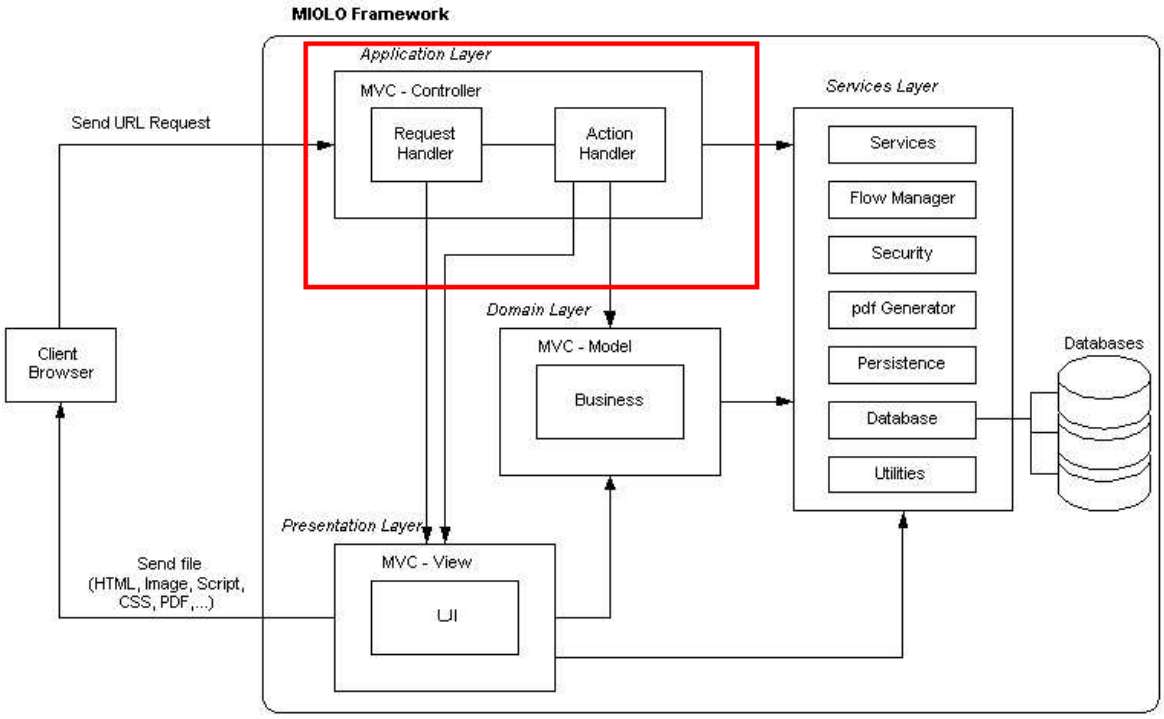


Figura 3.5 - Camada de integração em destaque na arquitetura MVC no MIOLO MATOS (2007)

3.3.4 Camada de Recursos

Na camada de recursos, entre outros dispositivos, encontramos uma camada de acesso a dados, mais precisamente a camada DAO (*Data Access Objects*) que objetiva abstrair o acesso a banco de dados relacionais, encapsular diversão opções de extensão propiciadas pela linguagem PHP, trazendo assim, a possibilidade de programar através de um única interface, independente das características particulares dos diversos SGBDs (MATOS, 2007).

MATOS (2007) observa que apesar de existirem outras soluções para esse problema, o framework MIOLO implementa sua própria versão DAO e descreve algumas características:

- Através do encapsulamento do mecanismo de acesso a dados oferecido pelo PHP, é usada uma única interface de programação de aplicação (API - *Application Programming Interface*);
- São fornecidos mecanismos básicos para geração automática de código SQL adaptado ao banco (inclusive *joins*, *offsets* e número máximo de linhas retornadas);
- Geradores de *sequences* e conversão de datas e horários para um formato padrão;
- Suporte a transações e
- Abstração de resultados de consultas (*queries*) em *ResultSets*, permitindo operações como travessia (*browse*), paginação, filtragem e ordenação do resultado de uma consulta.

A seguir temos um exemplo da especificação das *tags* de marcação que correspondem as informações de configuração de acesso ao banco de dados usado pela camada DAO:

```
<db>
     <base_name>
          <system>dbms</system>
          <host>address</host>
          <name>db_name</name>
          <user>username</user>
          <password>password</password>
     </base_name>
</db>
```

As informações acima ficam gravadas no arquivo de configuração do MIOLO, localizado em: <miolo_home>/etc/miolo.conf, conforme MATOS (2007) explica.

A *tag* <db> (*data base*) indica que a seção de configuração que se apresenta a seguir é relativa ao acesso de dados via banco de dados.

A *tag* <*base_name*> indica o nome da configuração de base de dados que será usada para um acesso definido nas classes de negócio. Uma mesma base de dados pode ser configurada de formas diferentes, através de usuários diferentes por exemplo. Da mesma forma, objetos podem ser criados com base em definições distintas de banco de dados, até mesmo em *hosts* diferentes com SGBDs diferentes.

A *tag* <*system*> indica o sistema SGBD que será usado pela DAO, por exemplo Oracle8, define o uso do SGBD Oracle.

A *tag* <*host*> indica o endereço lógico do servidor, geralmente um IP.

A *tag* <*db_name*> refere-se ao nome do banco de dados, referenciado pelo dbms (*tag* <*system*>) ou pelo *software* cliente do banco de dados.

A *tag* <*username*> identifica o usuário que será usado para fazer a conectividade de acesso ao banco de dados.

A *tag* <*password*> armazena a senha do usuário informado na *tag* anterior (<*username*>) autenticação no SGBD.

Nos trabalhos de MATOS (2007), podemos observar um claro exemplo do uso da DAO em uma aplicação. Neste exemplo, que veremos a seguir, usa-se a própria estrutura do MIOLO para exemplificar e demonstrar o uso de informações:

Descrição do modelo para tabela usado no exemplo:

```
miolo_transacao (idtrans (PK), transacao, idsistema (FK))
```

O código fonte de exemplo a seguir, mostra todos os campos de todos os registros da tabela MIOLO_transacao, em uma configuração chamada “admin”:

```
global $MIOLO;
$db = $MIOLO->GetDatabase('admin');
$sql = new sql('*','miolo_transacao');
$query = $db->GetQuery($sql);
$n = $query->GetRowCount();
$result = $query->result;
for ($i=0; $i < $n; $i++)
{
     echo "#$i - " . $result[$i][1] . '<br>';
}
```

A camada DAO do MIOLO, segundo MATOS (2007), encapsula os SGBDs PostgresSQL, MySQL, Oracle, Interbase, Firebird, SQLite, MSSQL entre outros, além do uso através de ODBC.

Ainda como recurso do MIOLO e diretamente relacionada à camada de acesso a dados, a persistência também é uma camada que oferece facilidades no tratamento das informações.

JUNIOR (2009) relata que a persistência de dados consiste no armazenamento não volátil e coerente das informações em um sistema de armazenamento de dados e que a persistência de objetos é o armazenamento consistente de objetos de uma aplicação orientada a objetos para que estes existam em diferentes tempos de execuções de diferentes aplicações.

A base para o desenvolvimento da camada de persistência do MIOLO teve como referência os trabalhos de Ambler e Artyom Rodoy, de acordo com MATOS (2007).

O mecanismo de persistência está encapsulado através da classe PersistentObject que por sua vez é usada pela classe MBusiness que passa a herdar daquela métodos tais como *save*, *delete* e *retrieve* que tratam automaticamente o acesso ao banco de dados, tornando os objetos de negócio virtualmente persistentes MATOS (2007).

Com a persistência são fornecidos mecanismos para recuperação e remoção de múltiplos objetos através dos objetos da classe MQuery que trabalham com ResultSets e cursores que são implementados como um vetor de objetos MATOS (2007).

Temos também suporte a "*lazy read*" através do uso de *proxies*. Um objeto *proxy* permite recuperar apenas alguns atributos do objeto, evitando o *overhead* de recuperar todos os atributos MATOS (2007).

A camada de persistência trata quando um objeto é recuperado, removido ou atualizado, e a mesma ação pode ser realizada nos objetos associados, se necessário for. Associações do tipo *ManyToMany* podem ser tratadas automaticamente pela camada de persistência e é possível mapear uma árvore de herança para um esquema no banco de dados MATOS (2007).

O *framework* conta com suporte a transações, geração automática de identificadores (OID), geração automática do comando SQL, acesso paginado e acesso a diferentes bancos de dados (características implementadas pela camada DAO do MIOLO).

A seguir temos um exemplo onde o resultado de um objeto MQuery ($query) é obtido aplicando uma definição de critério para pesquisa como, neste caso, sendo o apelido (*Nick*) do setor iniciado pelas letras 'PROR' e obtendo o *login* do usuário e apelido

do setor como colunas a serem resgatadas. No Anexo 4 (Modelagem de algumas tabelas/classes do Miolo) apresentamos, para aspecto geral, o modelo de classe, esquema de banco de dados e mapeamento usado como exemplo.

Trecho de código da classe *user*:

```
$criteria = $this->user->getCriteria();
$criteria->addCriteria('sector.nick','LIKE',"'PROR%'");
$criteria->addColumnAttribute('login');
$criteria->addColumnAttribute('sector.nick');
$query = $criteria->retrieveAsQuery();
```

Consulta respectiva ao exemplo anterior, codificado em linguagem SQL:

```
SELECT cm_usuario.login, cm_setor.siglasetor FROM
cm_usuario,cm_setor WHERE (cm_setor.siglasetor LIKE 'PROR%')
and (cm_usuario.idsetor=cm_setor.idsetor)
```

A geração de documentos para impressão, visualização e manipulação, é fundamental para qualquer sistema.

Com o MIOLO, podemos oferecer estes documentos através da geração de arquivos nos formatos PDF, CSV, TXT, entre outros.

Segundo MATOS (2007) no MIOLO os *reports* são implementados através da geração de arquivos PDF que, sendo acessados via browser, podem ser impressos a partir da máquina local. As classes do framework permitem que sejam utilizados diversos mecanismos para a geração de arquivos PDF, entre esses mecanismos estão a biblioteca JasperReport, e a biblioteca ezPDF.

Estrutura das classes de geração de relatório segundo MATOS (2007):

```
MReport
+----MCrystalReport
+----MezPDFReport
+----MJasperReport

CPdf (biblioteca ezPDF)
+----MCPdf

CEzPdf (biblioteca ezPDF)
+----MCEzCPdf
```

```
          MGridColumn
          +----MPDFReportColumn
          MGridControl

          +----MPDFReportControl
          MGrid

          +----MPDFReport
```

3.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou aspectos e conceitos relativos ao *framework* MIOLO detalhando pontos importantes entre as diversas camadas que compõem a arquitetura e funcionalidades além dos motivos que o tornarão apto a ser definido como base para utilização no projeto de desenvolvimento do SIGA.

4 SISTEMA INTEGRADO DE GESTÃO ACADÊMICA: ÁREA DE RH

A maior motivação para a elaboração deste trabalho foi a possibilidade de apresentar a comunidade acadêmica os aspectos e particularidades que envolvem a construção de um grande e complexo sistema de informação. Sistemas como o SIGA, da UFJF, demandam tempo, força de trabalho especializada e colaboração de diversas áreas do conhecimento.

Durante o tempo de desenvolvimento, alunos-bolsistas, docentes, técnicos administrativos em educação e empresas terceirizadas participaram ativamente contribuindo para construção dos diversos módulos que compõem o sistema de gestão sem deixar que as funções e atividades diárias necessárias ao funcionamento do órgão fossem interrompidas, mesmo diante dos desafios nas primeiras etapas de implantação quando as primeiras versões eram disponibilizadas e duramente criticadas.

Nas próximas seções deste capítulo trataremos do SIGA em especial do SIGA-RH, que é o motivo maior da realização deste trabalho.

4.1 SIGA

Segundo MATOS (2007), a UFJF na década de 90 utilizava um sistema de informação desenvolvido pela TECHNE S.A. Esse sistema possuía código aberto, o que permitia melhorias e adaptações. O código fonte era escrito em linguagem Algol e era executado em *mainframe*. Os relatórios dá época eram escritos em linguagem Cobol.

Com o *bug* do milênio, que atingiria o sistema da instituição, devido a ambigüidade causada pela forma de armazenamento dos dados relativos a datas, a universidade decidiu adquirir sistemas de informação para área acadêmica (LYCEUM), para área de gestão de recursos humanos (ERGON) e para controle de biblioteca (ALEPH) (MATOS, 2007).

Estes novos sistemas tinham alguns pontos negativos para instituição, como explica MATOS (2007): “... o código fonte era completamente fechado; as customizações eram difíceis, demoradas e, em alguns casos, impossíveis; os custos de aquisição e manutenção mensal eram elevados e as bases de dados não eram integradas entre si”. Era notório o não atendimento das expectativas da instituição, logo, sistemas secundários e sem integração se multiplicavam, nos diversos setores da universidade (MATOS, 2007).

Em 2002 a UFJF passou a adotar novas premissas na área de informática (MATOS, 2007):

> - Preservação da independência da universidade em relação ao desenvolvimento, implantação e manutenção de sistemas informatizados; ampla utilização da política de software livre; uso de ferramentas com código aberto; incentivo à inteligência disponível na própria UFJF.
> - Transparência das informações, com acesso "universal" (via web), como ferramenta para validação e correção das bases de dados existentes.
> - Discussão dos fluxos de trabalho e de documentos adotados pela universidade, uma vez que muitos destes fluxos eram considerados ultrapassados ou não adaptados às tecnologias existentes.
> - Foco na integração, não apenas dos sistemas administrativos e de gestão, mas também em programas de uso mais geral como o correio eletrônico ou sistemas externos a UFJF (como os sistemas do governo federal).
> - Implantação de processos de melhoria da qualidade.

Com a mudança de visão, foi adotada uma plataforma para auxiliar o desenvolvimento de sistemas, o *framework* MIOLO, que possibilitou o surgimento do primeiro módulo do SIGA (MATOS, 2007).

O SIGA é um sistema integrado, formado por quatro grandes módulos principais que são: Módulo de Ensino, Módulo Administrativo, Módulo de Recursos Humanos e Módulo de Biblioteca que além de compartilharem dados entre si, com a mesma base de dados, também contam com diversos outros módulos que os auxiliam, como por exemplo o Módulo de Terceirizados que liga-se ao Módulo de RH, o Módulo de Apoio ao Aluno que usa dados do Módulo de Ensino, o Módulo de Bolsas que relaciona-se diretamente com dados acadêmicos junto ao Módulo de Ensino e dados financeiro referente ao processamento de pagamentos baseando-se em informações contidas no Módulo Administrativo, os Módulos de Almoxarifado e Protocolo que se misturam ao Módulo Administrativo nas diversas fases de aquisição de materiais na universidade, além, ainda, de um módulo comum que praticamente todos os demais módulos se apóiam, que é denominado Módulo *Common*, e que provê dados básicos e essenciais ao funcionamento do sistema como um todo. Outros módulos com funções mais específicas também compõem a extensa e complexa trama de desenvolvimento. Podemos observar na representação da Tabela 4.1, um *hall* de módulos produzidos e mantidos pelo Centro de Gestão do Conhecimento Organizacional (CGCO), publicada junto a uma notícia de atualização do sistema, no dia 16 de julho de 2009, na página principal do sistema, relacionando os módulos que o compõem.

Biblioteca	PingIFES	Orçamento Interno/Externo
Graduação	Registro de Diplomas	Contratos
Pós-Graduação	Eventos	Terceirizados
Ens. Fundamental/Médio	Egresso	Controle de Recursos
Moodle NEAD	Avaliação Institucional	SEFIP
Projetos de Pesquisa	**Recursos Humanos**	Webmail
Projetos de Extensão	Protocolo	Documentação
Apoio Aluno	Bolsas	Version
Estágio	Requisições	Common
Concurso Público	Licitação	Datamap
Programa de Ingresso	Almoxarifado	
Eleições	Patrimônio	
Lattes	Financeiro	

Tabela 4.1 - Módulos componentes do SIGA [adaptado] [sem grifo no original]

Como discutido na seção 2.1 o objetivo desse sistema de informação é gerenciar os dados obtidos através dos diversos processos administrativos que diariamente agregam grande volume de informações dos diversos setores da universidade.

Em MATOS (2009), existe relato sobre os bons resultados decorrentes da implantação do SIGA e do domínio da tecnologia utilizada na construção do projeto com um todo. Relatos esses que apontam o reconhecimento da qualidade do sistema pelo Ministério da Educação através de convênios com a Secretaria de Educação Tecnológica (SETEC) recomendando o uso do sistema pelas Escolas Profissionais e Técnicas. Além de facilitar o fornecimento de dados para o PingIFES[26] comparando-se as dificuldades e desafios enfrentados por outras instituições.

Segundos citado por MATOS (2007), no que tange o aspecto tecnológico, o desenvolvimento do SIGA é baseado no *framework* brasileiro MIOLO (capítulo 3 deste trabalho).

É levantado por MATOS (2009) que o SIGA continua em pleno desenvolvimento e que entre as perspectivas futuras estão os seguintes pontos:

- Integração com outros sistemas em uso na universidade (Vestibular);
- Criação de ferramentas para geração de informações gerenciais;
- Ampla documentação dos módulos desenvolvidos;
- Implantação de processos de melhoria de qualidade em todas as fases do desenvolvimento.

Os servidores dos ambientes de desenvolvimento e de produção do SIGA, são sustentados por sistemas operacionais Linux. Como ferramenta de controle de versão foi adotado o Subversion (seção 2.7). Como servidor web é utilizado o Apache, conforme seção 2.2.1 (MATOS, 2007).

[26] PingIFES é a denominação adotata para a plataforma tecnológica que define como ocorre a troca de informações entre as IFES e demais órgãos do governo.

Em 2009, o SIGA começou a ser processado em máquinas de alto desempenho do conjunto de soluções *Blade*[27] da *Hewlett Packard*[28] (HP), economizando espaço físico e proporcionando maleabilidade através das opções de virtualização das máquinas, tornando possível adequar o poder de processamento a demanda em situações distintas no calendário da instituição.

Figura 4.1 - Lâminas do Blade

Figura 4.2 - Discos rígidos para armazenamento (*storage*)

[27] Nome comercial para solução em servidores de missão crítica da empresa HP expansível através de lâminas de processamento.

[28] http://www.hp.com/

4.2 ÁREAS DO SIGA

Conforme mencionado anteriormente o desenvolvimento com o *framework* MIOLO pode ser modularizado, sendo assim, no projeto SIGA decidiu-se desenvolver módulos para áreas de atuação do sistema. Como exemplo, pode-se dizer que o módulo Recursos Humanos faz parte da área administrativa e tem com módulos componentes terceirizados, PROADES, capacitações e integração com o SIAPE (SIGA, 2009). As áreas mais relevantes, para este trabalho, serão resumidamente apresentadas nos próximos tópicos.

4.2.1 Área de Ensino (Acadêmico)

A primeira área a ser desenvolvida que abrange as principais atividades relacionadas ao registro acadêmico referentes às unidades acadêmicas, alunos e professores. Faz parte da área de cobertura do módulo Ensino, além da graduação, o ensino médio, com a utilização do sistema pelo Colégio de Aplicação João XXIII, a pós-graduação tanto para *Stricto Sensu* quanto para *Lato Sensu* (MATOS, 2009).

Algumas das funcionalidades são (MATOS, 2009):

- Registro e manutenção de cursos, currículos e disciplinas;
- Oferta de disciplinas a cada período letivo, através dos planos departamentais;
- Pré-matrícula e matrícula de alunos;
- Lançamento de notas parciais e finais pelos professores;
- Histórico escolar acessível aos alunos;
- Registro das bancas e dissertações dos alunos de pós-graduação;
- Registro de referências para o ENADE;
- Registro de diploma;
- Geração do diploma para impressão, entre outros recursos.

4.2.2 Área das Bibliotecas

A área que envolve as Bibliotecas busca atender as atividades relacionadas as bibliotecas da UFJF que até o memento correspondem a 13 bibliotecas setoriais, além de uma biblioteca principal (central) (MATOS, 2009).

Alguns dos recursos dos módulos de bibliotecas são (MATOS, 2009):

- Manutenção das tabelas de operações, gêneros, direitos, coleções, infrações, regras de circulação, situação de reserva, etc.;
- Catalogação de obras e exemplares seguindo formato MRAC21[29];
- Empréstimo para alunos, professores e funcionários;
- Devolução de exemplares;
- Pagamento de multas;
- Histórico de empréstimo por usuários ou obra;
- Verificação da situação de empréstimo;
- "Nada consta" para usuários da biblioteca (exigido para os formandos);
- Pesquisa no acervo;
- Reserva, renovação e cancelamento de empréstimos;
- Visualização de informações de empréstimo e multas;

O módulo de biblioteca faz uso do módulo de recursos humanos quando precisa validar a situação de um funcionário ou aluno, por exemplo, se está regularmente matriculado, com isso percebe-se a importância da integração dos sub-sistemas (MATOS, 2009).

4.2.3 Área da Administração

Segundo MATOS (2009), "o módulo [área] de administração é, sem dúvida o mais complexo e o mais abrangente, envolvendo as 'atividades meio' da universidade".

Alguns dos papéis do módulo administrativo são listados abaixo (MATOS,2009):

- Requisições de almoxarifado, compras, diárias, hotel, passagens, restaurante, serviços externos e internos e veículos;
- Controle de pagamento de bolsa que evita acúmulo de bolsas e interrompe bolsa de alunos que se formam ou façam matrícula;
- Fornecendo relatórios gerenciais e de controle, através da transferência de dados da folha do SIAFI[30] - Sistema Integrado de Administração Financeira;
- Licitação de materiais e serviços com:
  - Seleção de materiais;
  - Definição de empresas participantes;
  - Lançamento de propostas enviadas pelos fornecedores;
  - Definição de vencedores e fornecimento de relatórios[31];

[29] http://www.loc.gov/marc/bibliographic/

[30] SIAFI é um sistema do governo federal que integra a programação financeira, de execução orçamentária e de controle interno do Poder Executivo e fornece informações gerenciais, confiáveis e precisas para todos os níveis da Administração (http://www.tesouro.fazenda.gov.br)

- Controle orçamentário interno e externo;
- Controles contábeis com geração de relatórios de controle e gerenciais;
- Controle de empenho, com liquidação e pagamento;
- Controle de patrimônio integrado ao controle de empenho;
- Controle de almoxarifado integrado ao controle de empenho;
- Controle dos trâmites dos processos da UFJF pelo módulo de protocolo.

4.3 ÁREA DE RECURSOS HUMANOS (SIGA-RH)

Conforme MATOS (2007), a área de recursos humanos do SIGA, compreende as principais atividade ligadas aos registros de servidores da UFJF, embora não execute a folha de pagamento, pois esta é centralizada pelo governo federal pelo Sistema Integrado de Administração Pessoal (SIAPE).

Segundo MAULER (2003) apud MARQUES (2002) “os quadros de pessoal das universidades federais representam um campo muito rico de profissionais, ocupantes de cargos que abrangem desde as atribuições mais genéricas até as mais especializadas”.

Ainda de acordo com MAULER (2003), apud FARIA e LOUREIRO (1996) os esforços pela maximização da qualidade tornam-se importantes diante das profundas transformações mundiais e forçaram organizações públicas ou privadas, a buscarem novos padrões de qualidade e produtividade. Sendo que a estratégia para alcançar o objetivo está relacionada com o investimento nos recursos humanos.

Em MAULER (2008) o módulo de recursos humanos da UFJF aparece, em novembro de 2005, como disponível para os usuários, embora sua elaboração tenha sido iniciada em julho de 2004.

MAULER (2008) faz questão de lembrar que o processo de desenvolvimento é contínuo e dinâmico. E que depende da legislação em vigor e suas alterações, que muitas vezes, exige remodelagem do algumas funcionalidades do sistema.

No Anexo 5 – Diagrama de Casos de Uso do Módulo RH encontra-se os casos de uso que nortearam o desenvolvimento do módulo.

O escopo de recursos humanos do SIGA é descrito por MAULER (2008) numa lista de abrangência:

[31] O módulo administrativo usa base de dados do CATMAG e CATSER do SIASG (Sistema Integrado de Administração de Serviços Gerais) do governo federal (MATOS, 2009).

- Dados pessoais de servidores, incluindo informações pessoais com foto, documentos, contato, dados bancários, controle de dependentes, escolaridade, pós-graduações, capacitações;
- Dados funcionais atuais e dados de histórico funcional (provimentos gerados por ocorrências funcionais);
- Ocorrências funcionais: ingresso, exclusão, remoções, remoções múltiplas, progressões funcionais, aposentadoria, mudança de cargo, mudança de ambiente organizacional;
- Controle de períodos aquisitivos e usufruição de férias;
- Controle de licenças/afastamentos;
- Exercício em outros órgãos: requisições, cessões, mandatos classistas, lotações provisórias;
- Controle de contratos temporários (professores substitutos e visitantes);
- Apoio aos processos de aposentadoria: controle de averbações de tempo de serviço, cálculo da média remuneratória para pagamento de proventos;
- Saúde e segurança: controle de adicionais de insalubridade, periculosidade e raios-x;
- Controle de vagas do quadro de pessoal;
- Módulo PROADES: Programa de Avaliação de Desempenho dos servidores Técnico-Administrativos em Educação;
- Módulo de Capacitação: controle de inscrições e participação de servidores em programas de capacitação;
- Integração SIAPE: carga dos arquivos-espelho de servidores e pensionistas/geração de arquivo de férias para carga no SIAPE;

Quase todas as áreas do SIGA tem alguma dependência do módulo de Recursos Humanos. As regras implementadas e oferecidas pelo módulo RH tem, por exemplo, a função de verificar em seus registros e informar ao módulo de requisições se um servidor encontra-se em férias, quando um lançamento de diária é requisitado. A biblioteca quando empresta um livro a um servidor, antes verifica através do RH se este está ativo no sistema.

4.3.1 Funcionalidades

O sistema de recursos humanos da UFJF oferece diversas funcionalidades ligadas diretamente à área de gestão de pessoal, porém, mesmo dentro da área de recursos humanos encontraremos várias divisões de perfis de acesso. Algumas funcionalidades

estão disponíveis somente para alguns setores do RH enquanto outras estão distribuídas entre outros agentes da UFJF. A seguir apresentamos duas figuras ilustrando a visão de acesso por um administrador de sistema e por um secretário de unidade respectivamente na Figura 4.3 e Figura 4.4.

Figura 4.3 - Funcionalidade do módulo RH pela visão do administrador do sistema

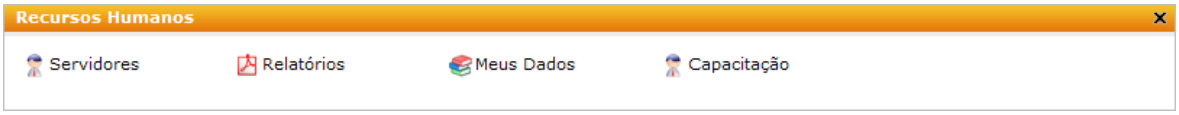


Figura 4.4 - Funcionalidade do módulo RH pela visão de secretário de unidade

Entre algumas das funcionalidades do RH, podemos citar a visualização do organograma administrativo, contendo todas as unidades na UFJF, sob o formato de uma árvore de diretórios (Apêndice 1), embora, seja um recurso simples é muito útil em diversas partes dos sistema.

4.3.1.1 Dados pessoais

O ícone representando a funcionalidade “Pessoas” apresenta em seu interior um formulário contendo ligação (*link*) para criação de uma nova pessoa no sistema, que servirá para todos os demais módulos do SIGA, outra característica oferecida é por uma grade (*grid)* que possibilita a pesquisar pelo nome ou pelo CPF de um indivíduo, com a finalidade de manter os seus dados atualizados. Ver Figuras 4.5 e 4.6. Esta opção está visível apenas aos administradores do sistema e permite além dos dados pessoais básicos, visualizar e manter dados de contato, documentos e dados bancários.

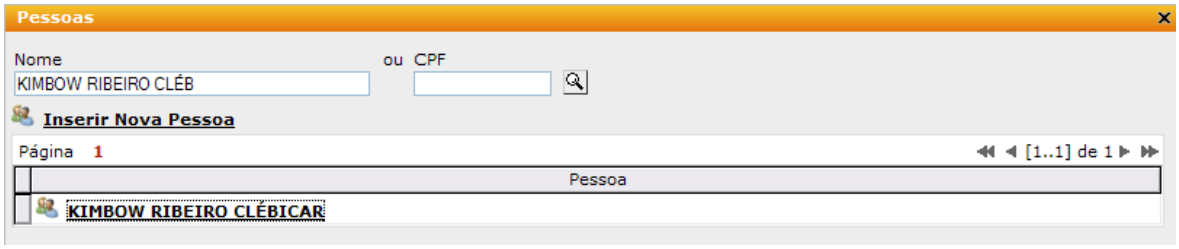


Figura 4.5 - Formulário para pesquisa de pessoas

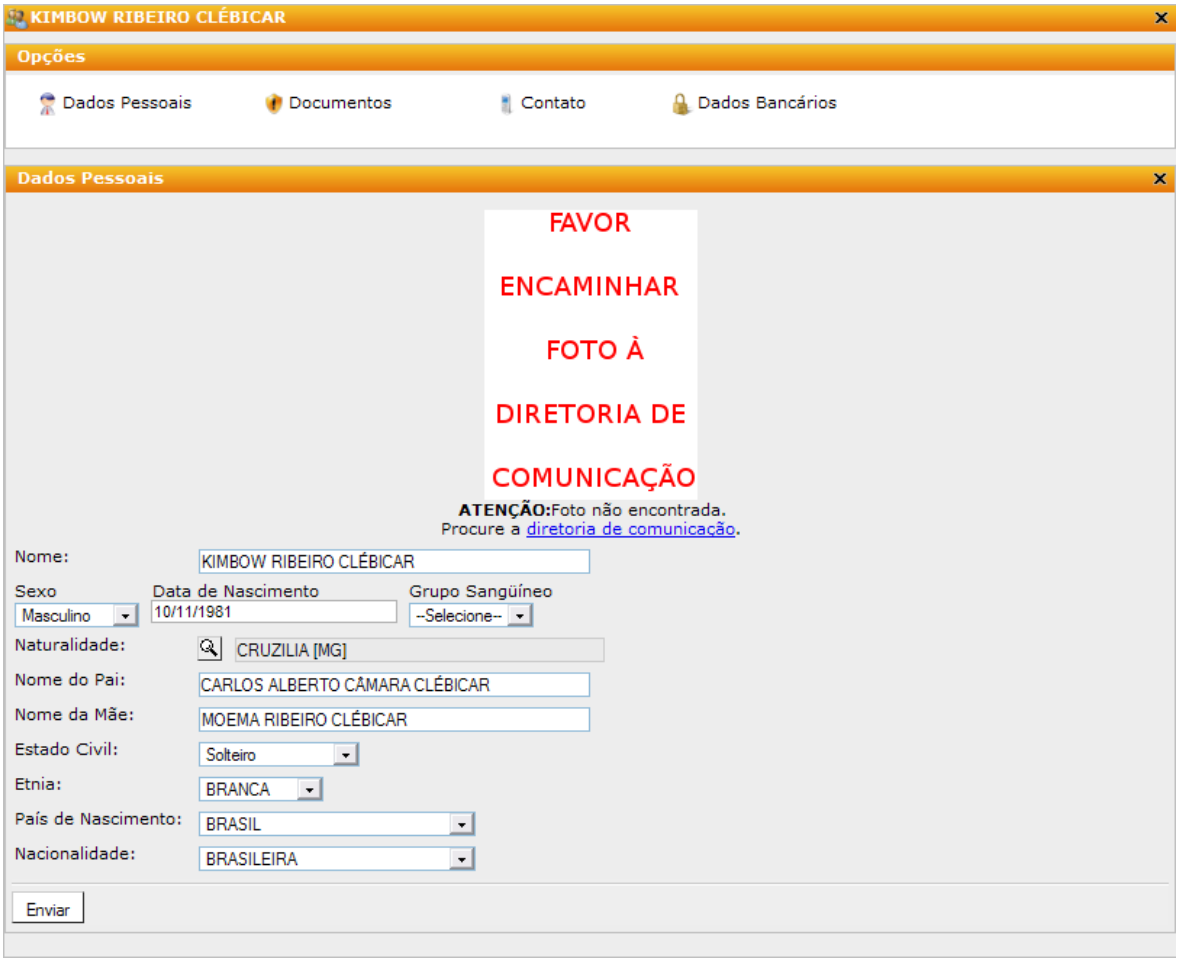


Figura 4.6 - Menu de opções de pessoa e formulário de Dados Pessoais

4.3.1.2 Dados funcionais

Os dados relativos ao servidor no que tange o seu trabalho na instituição são disponibilizados aos usuários devidamente credenciados e autenticados com a funcionalidade “Dados funcionais”. Esta opção é alcançada pelo *menu* de “vínculo” após selecionar o servidor do qual se quer obter as informações através da grade de pesquisa de servidores. A Figura 4.7 traz na sua parte inferior um painel de informações do servidor relacionadas diretamente ao exercício de sua função.

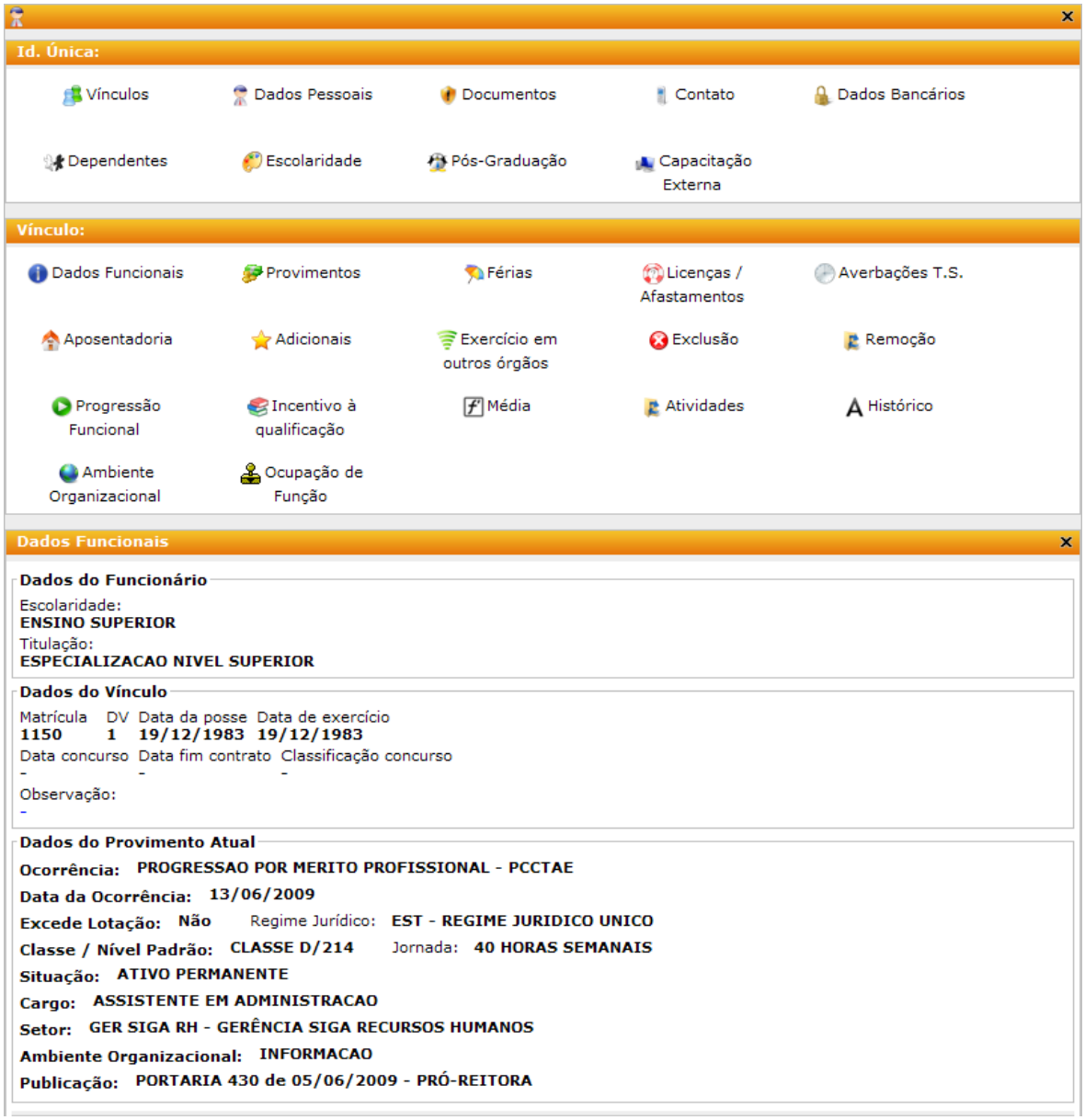


Figura 4.7 - Painel com informações funcionais do servidor

4.3.1.3 Férias

Entre as funcionalidades que podemos destacar como um bom exemplo de administração distribuída dos dados por um sistema integrado está a marcação de férias, onde os secretários das unidades acessam um formulário informando as datas e opções correspondentes às férias dos técnicos administrativos em educação ou docentes do seu, e somente, seu setor. A marcação das férias exige consistência de informações já contidas no banco de dados, regras impostas por força de lei e calendário previamente determinado para facilitar o planejamento da instituição. Vale lembrar que a Pró-Reitoria de Recursos Humanos possui habilidade para alterar a marcação de férias em casos

especiais, evidenciando as diferenças de perfil de acesso. O Anexo 7 – Caso de uso de Férias mostra detalhadamente a descrição do caso de uso do lançamento de férias, incluindo as regras de negócio para lançamento de período aquisitivo que são fundamentais para a usufruição das férias.

A seqüência de figuras a seguir ilustra os pontos importantes do processo de lançamento de férias de um servidor da UFJF.

O primeiro passo consiste na abertura de período aquisitivo para usufruição de férias, um servidor deve ter no mínimo um ano de efetivo exercício para poder fazer adquirir o direito de usufruir suas férias. A Figura 4.8 apresenta um formulário de lançamento de período aquisitivo. Este formulário com a maioria dos campos preenchidos automaticamente com base nos dados do servidor. Deixando somente um campo de observação a critério do usuário. Para maiores detalhes de como os dados desse formulário é influenciado pelas informações já registradas no SIGA deste servidor, consulte o Anexo 8 – Casos de Uso para Período Aquisitivo.

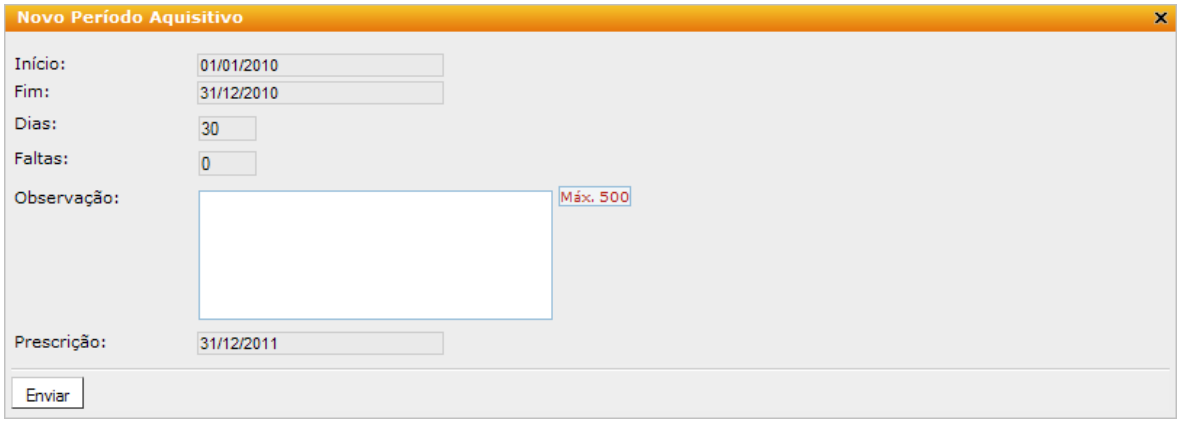
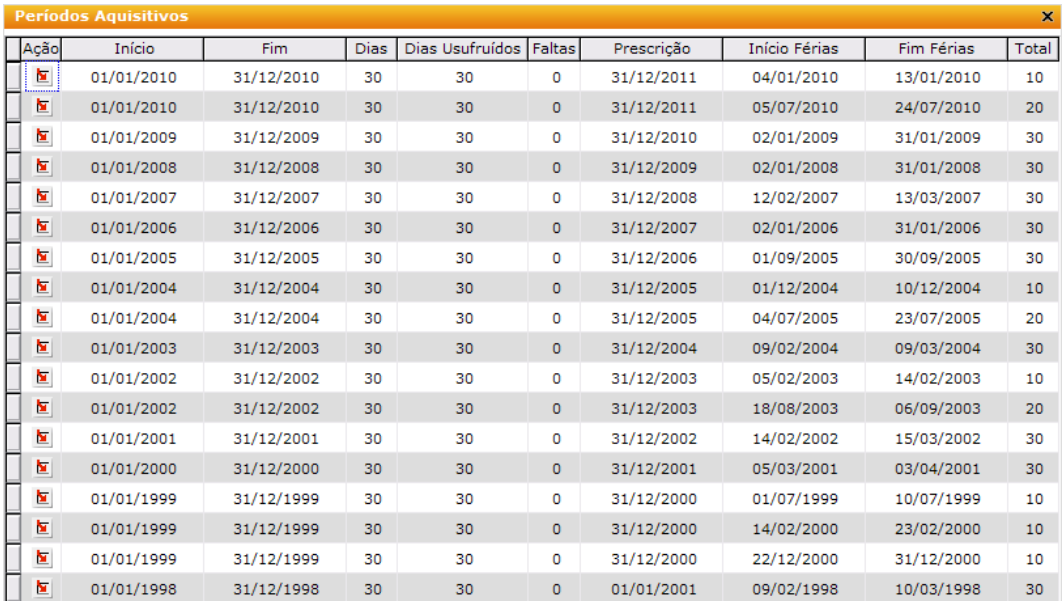


Figura 4.8 - Formulário para registro de período aquisitivo de um servidor

No sistema podemos observar uma grade com as informações relativas ao período aquisitivo e férias dos servidores. A Figura 4.9 apresenta uma parte de um exemplo destes dados, que mesclam em um único ponto dados do período aquisitivo e do parcelamento de férias do funcionário. Esta grade passou por modificações para mesclar essas informações com o objetivo de facilitar a visualização e a conferência dos dados.



Períodos Aquisitivos

Ação	Início	Fim	Dias	Dias Usufruídos	Faltas	Prescrição	Início Férias	Fim Férias	Total
	01/01/2010	31/12/2010	30	30	0	31/12/2011	04/01/2010	13/01/2010	10
	01/01/2010	31/12/2010	30	30	0	31/12/2011	05/07/2010	24/07/2010	20
	01/01/2009	31/12/2009	30	30	0	31/12/2010	02/01/2009	31/01/2009	30
	01/01/2008	31/12/2008	30	30	0	31/12/2009	02/01/2008	31/01/2008	30
	01/01/2007	31/12/2007	30	30	0	31/12/2008	12/02/2007	13/03/2007	30
	01/01/2006	31/12/2006	30	30	0	31/12/2007	02/01/2006	31/01/2006	30
	01/01/2005	31/12/2005	30	30	0	31/12/2006	01/09/2005	30/09/2005	30
	01/01/2004	31/12/2004	30	30	0	31/12/2005	01/12/2004	10/12/2004	10
	01/01/2004	31/12/2004	30	30	0	31/12/2005	04/07/2005	23/07/2005	20
	01/01/2003	31/12/2003	30	30	0	31/12/2004	09/02/2004	09/03/2004	30
	01/01/2002	31/12/2002	30	30	0	31/12/2003	05/02/2003	14/02/2003	10
	01/01/2002	31/12/2002	30	30	0	31/12/2003	18/08/2003	06/09/2003	20
	01/01/2001	31/12/2001	30	30	0	31/12/2002	14/02/2002	15/03/2002	30
	01/01/2000	31/12/2000	30	30	0	31/12/2001	05/03/2001	03/04/2001	30
	01/01/1999	31/12/1999	30	30	0	31/12/2000	01/07/1999	10/07/1999	10
	01/01/1999	31/12/1999	30	30	0	31/12/2000	14/02/2000	23/02/2000	10
	01/01/1999	31/12/1999	30	30	0	31/12/2000	22/12/2000	31/12/2000	10
	01/01/1998	31/12/1998	30	30	0	01/01/2001	09/02/1998	10/03/1998	30

Figura 4.9 - Grade de informações de períodos aquisitivos e férias

O processo de agendamento de férias, assim como o processo de abertura de período aquisitivo, leva em consideração o histórico de informações do servidor.

A Figura 4.10 apresenta o formulário padrão para agendamento de férias. Vale ressaltar que os dias são calculados com base no saldo de dias e no cargo que o servidor exerce. Por exemplo, docentes tem a possibilidade de agendar até 45 dias de férias enquanto técnicos administrativos em educação podem agendar até 30 dias de férias. O parcelamento das férias obedece a legislação vigente previsto do regime jurídico único. Já no caso das pessoas regidas pelos contratos temporários, o embasamento está diretamente vinculado à CLT (Consolidação das Leis Trabalhistas), portanto, podem agendar 30 dias de férias. No Anexo 7 – Casos de Uso de Férias, as regras de negócio aplicáveis ao lançamento de férias são apresentas, incluindo as notas de implementação.

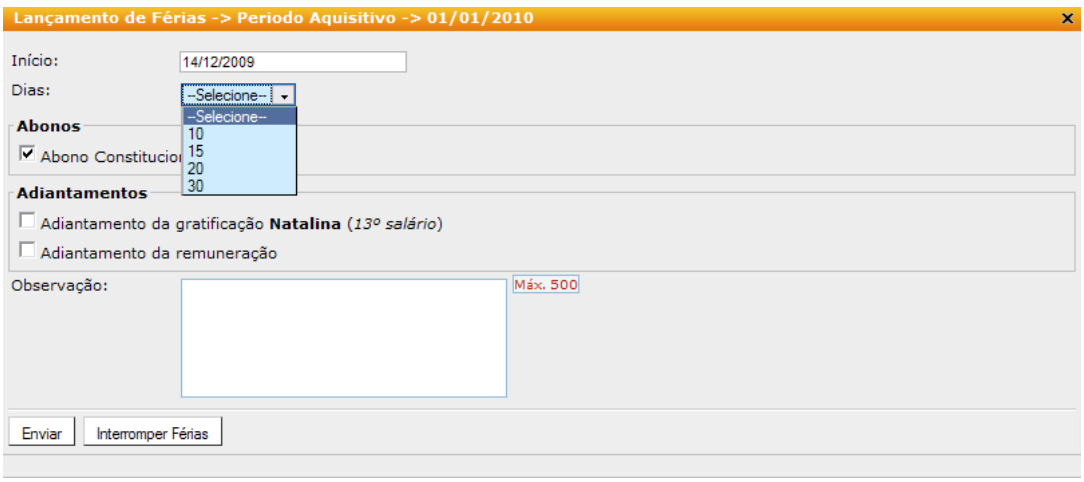


Figura 4.10 - Formulário de "lançamento" de férias

As Figuras 4.11, 4.12 e 4.13 ilustram mensagens enviadas ao usuário do sistema quanto ao sucesso da marcação de uma parcela das férias de um servidor, um erro quando um usuário tentar abrir um período aquisitivo para o ano cujo período já existe. Por último uma mensagem de confirmação quando da exclusão de uma parcela de férias.

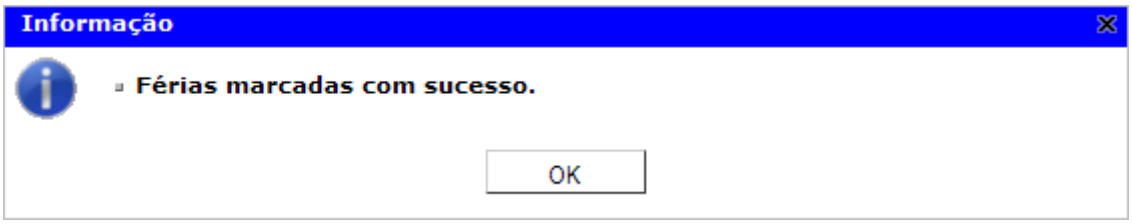


Figura 4.11 - Mensagem de confirmação de marcação de férias do sistema

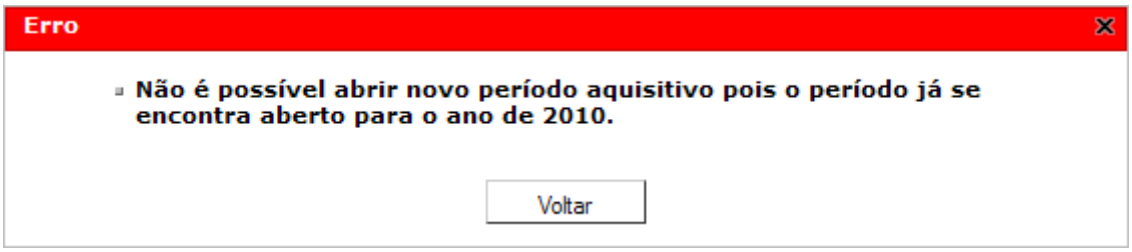


Figura 4.12 - Mensagem de erro ao tentar gerar período aquisitivo já existente

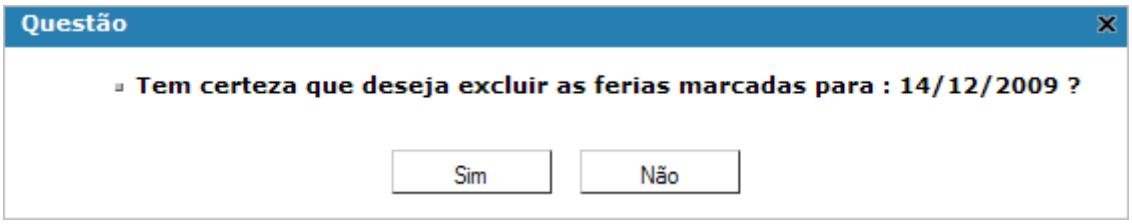


Figura 4.13 - Mensagem de confirmação para exclusão de férias de servidor

4.3.1.4 Licenças e Afastamentos

Segundo MAULER (2008) a gerência de Licenças e Afastamentos provê meios para tratar licenças médicas, férias e outras licenças/afastamentos previstos no regime jurídico único e outras leis aplicáveis.

Quando acessado o *menu* de opções, referentes ao vínculo do servidor e escolhido o *link* referente a Licença/Afastamento uma grade como a da Figura 4.14 deverá aparecer exibindo um histórico em ordem cronológica das licenças e afastamentos pertencentes ao servidor.

Figura 4.14 - Grade de Licenças/Afastamento de servidor MAULER (2008)

No processo de lançamento de licença/afastamento o primeiro passo é escolher o tipo (ocorrência) que gerou tal licença/afastamento, como mostra a Figura 4.15. As ocorrências serão exibidas como base no histórico de informações do servidor. Por exemplo, uma licença maternidade nunca deverá ser opção de lançamento para um servidor do sexo masculino.

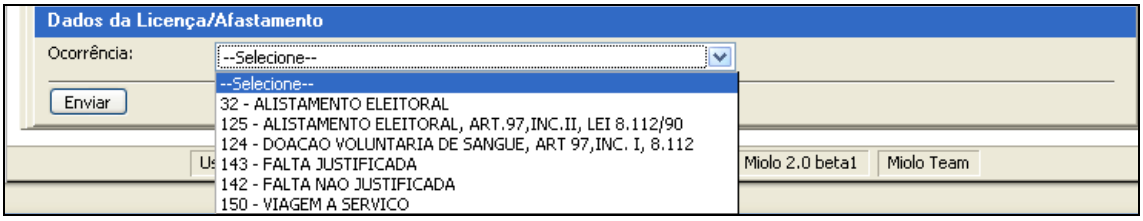


Figura 4.15 - Ocorrência que gerou a licença ou afastamento MAULER (2008)

O segundo passo para o lançamento de uma licença/afastamento é preencher o formulário com as datas de início e fim da licença ou afastamento, o motivo da licença ou afastamento, e a publicação referente aquela licença ou afastamento. A Figura4.16 apresenta os campos a serem preenchidos pelo servidor que estiver lançando a licença/afastamento.

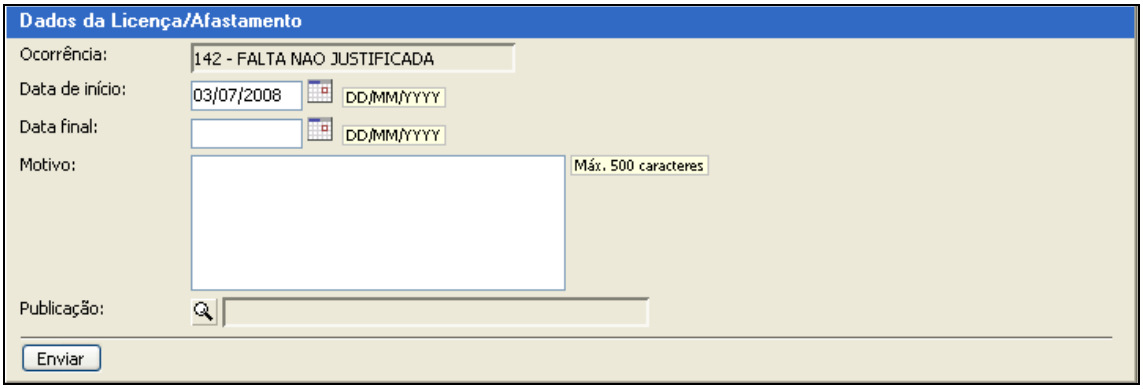


Figura 4.16 - Formulário dos dados da licença ou afastamento MAULER (2008)

A Figura 4.17 apresenta uma mensagem de confirmação positiva para o lançamento de uma Licença/Afastamento.

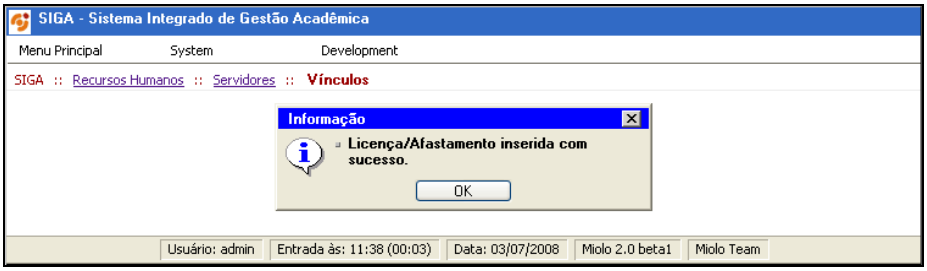


Figura 4.17 - Confirmação de "lançamento" de lic/afast MAULER (2008)

A “FALTA NÃO JUSTIFICADA” aparece na grade do sistema imediatamente após o lançamento e está disponível para qualquer módulo que precise fazer uma verificação se um servidor está em licença ou afastamento, ou até mesmo contabilizar dias de licença parar conceder benefícios ao servidor, como no caso de progressões. A Figura 4.18 mostra a falta não justificada lançada ao servidor por intermédio da grade de informações.

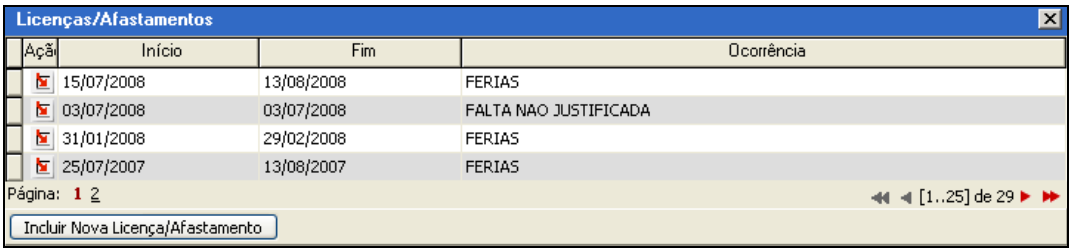


Figura 4.18 - *Grid* (grade) de lic./afast. de servidor MAULER (2008)

4.3.1.5 Relatórios

O SIGA-RH é composto por uma série de documentos e relatórios que auxiliam nos processos de gestão humana, como podemos notar pelo menu de opções (visão do administrador) na Figura 4.19.

Figura 4.19 - Opções de relatórios do SIGA-RH

MAULER (2008) enumera alguns relatórios disponibilizados pelo módulo:

– Servidores cedidos: lista os servidores que se encontram em exercício em outros órgãos em determinada data;

– Licenças/Afastamentos: lista os servidores licenciados da unidade, com opção de relatório geral ou por setor;

– Adicionais: lista os servidores da unidade que recebem adicional de insalubridade, periculosidade ou raios-x;

– Aniversariantes: lista os servidores da unidade que fazem aniversário no mês ou dia considerado;

– Pensionistas: relatório de pensionistas de servidores da unidade;

– Dados Bancários: relatório de dados bancários de servidores da unidade, por setor.

Como exemplo, os aniversariantes do dia podem ser consultados e exibidos através de relatório em PDF de maneira simples, e representa objetivamente a geração de arquivos PDF no sistema. A seguir podemos observar pelas Figuras4.20 e 4.21 respectivamente o formulário de obtenção e uma “foto” do arquivo PDF contendo a listagem solicitada, embora a geração de arquivos CSV também esteja disponível, conforme o botão da Figura 4.20 nos indica.

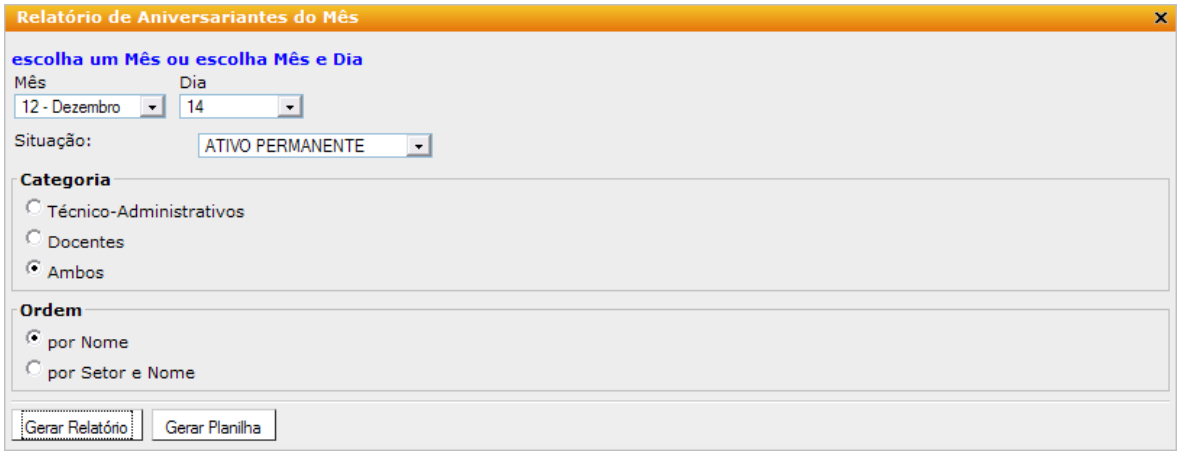


Figura 4.20 - Parâmetros de geração do relatório de aniversariantes

UFJF – Universidade Federal de Juiz de Fora
CGCO – Centro de Gestão do Conhecimento Organizacional
SIGA – Sistema Integrado de Gestão Acadêmica – Módulo Recursos Humanos

Página: 1/1

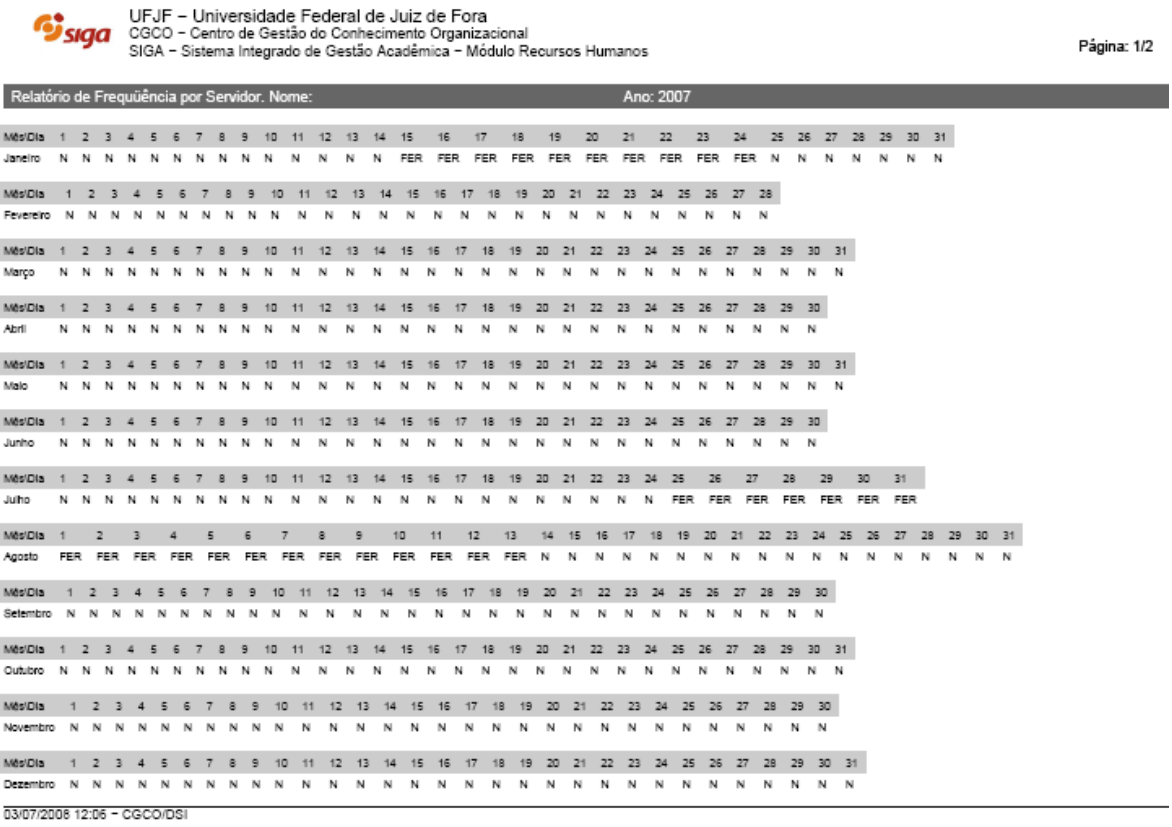
Aniversariantes no dia de 14 de dezembro. (Ativo permanente). Total: 6.

Matrícula	Nome	Setor	Telefone	Email
3312	KATIA VALERIA BASTOS DIAS BARBOSA	ÁREA CENTRO TRATAMENTO INTENSIVO		
1421	LUCIANA BITTENCOURT VILLELA	DEPTO DE TURISMO / ICH	3232	
114	LUZIA DE FATIMA DE ASSIS PEREIRA	SUBGERÊNCIA COZINHA RU CAMPUS	232	
114	MARCELO SOARES DULCI	DEPTO DE CIENCIAS SOCIAIS /ICH	323	
143	MARGARETE FERRARI COSTA	SETOR DE PEDIATRIA /HOSP UNIV	32 322	
114	OSVALDO DOS SANTOS	GERÊNCIA DE OBRAS CIVIS E REFORMAS	32	

/11/2009 - CGCO/DSI

Figura 4.21 - Exemplo do relatório de aniversariantes

Outro relatório simples e amplamente usado pelos secretários de unidade e Pró-Reitora de Recursos Humanos é o relatório de freqüência. Em formato PDF ele é disponibilizado também através do *menu* de relatórios e é baseado em duas opções, uma que lista a freqüência diária do servidor, nos meses do ano e outro que aponta a freqüência de um determinado mês para os servidores do setor escolhido. A Figura 4.22 apresenta um exemplo deste mapa de freqüência que têm sua descrição de caso de uso apresentada no Anexo 6 – Casos de Uso de Freqüência.



UFJF – Universidade Federal de Juiz de Fora
CGCO – Centro de Gestão do Conhecimento Organizacional
SIGA – Sistema Integrado de Gestão Acadêmica – Módulo Recursos Humanos

Página: 1/2

Relatório de Frequüência por Servidor. Nome: Ano: 2007

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Janeiro	N	N	N	N	N	N	N	N	N	N	N	N	N	N	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	N	N	N	N	N	N	N

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Fevereiro	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Março	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Abril	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Maio	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Junho	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Julho	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	FER	FER	FER	FER	FER	FER	FER

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Agosto	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Setembro	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Outubro	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Novembro	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Mês/Dia	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Dezembro	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

03/07/2008 12:06 - CGCO/DSI

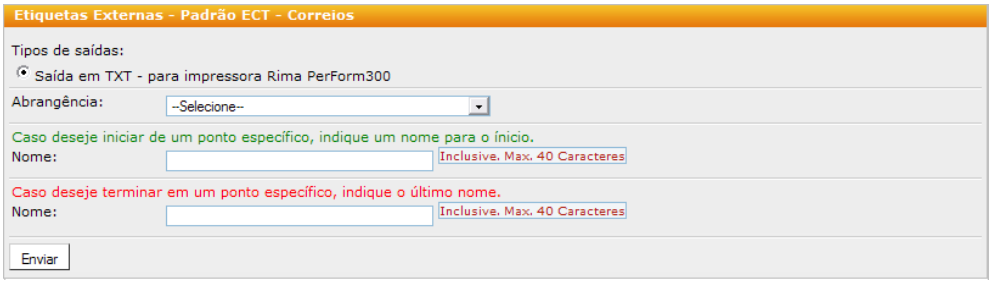
Figura 4.22 - Exemplo do "Mapa de Freqüência" (MAULER, 2008)

Podemos citar como exemplo a geração de arquivos no formato TXT para impressão de etiquetas que são usadas para enviar aos servidores ativos, aposentados e pensionistas folhetos com notícias da UFJF como convites para palestras e afins.

Para servidores ativos, ainda existe a possibilidade das etiquetas serem obtidas com objetivo de encaminhar, cartas e documentos, para o local de trabalho do servidor, ou seja, para as unidades da UFJF.

Pelo fato da UFJF adotar impressoras matriciais para impressão de suas etiquetas, visando a economia de dinheiro público, o SIGA-RH, por este caso específico disponibiliza instruções que ajudam o usuário a controlar a impressão.

Podemos conferir parte da funcionalidade de etiquetas nas Figuras 4.23 e 4.24:



Etiquetas Externas - Padrão ECT - Correios

Tipos de saídas:

Saída em TXT - para impressora Rima PerForm300

Abrangência: --Selecione--

Caso deseje iniciar de um ponto específico, indique um nome para o início.

Nome: Inclusive. Max. 40 Caracteres

Caso deseje terminar em um ponto específico, indique o último nome.

Nome: Inclusive. Max. 40 Caracteres

Enviar

Figura 4.23 - Formulário para geração de etiquetas externas (padrão correios)

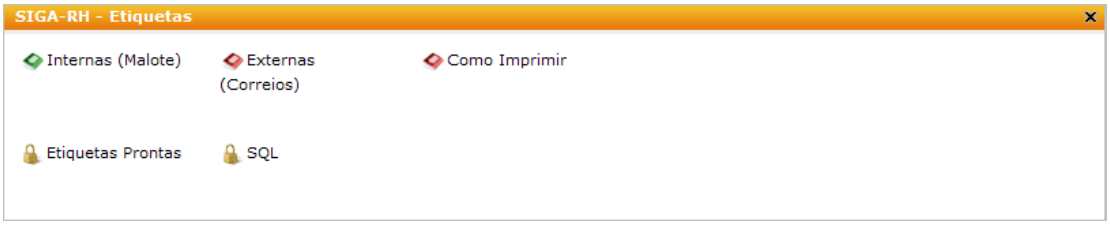


Figura 4.24 - *Menu* de opções para geração de etiquetas e instruções de impressão

Existem dentro da opção de relatórios alguns recursos gerenciais para o próprio sistema. Estes relatórios procuram facilitar o processo de desenvolvimento e manutenção do próprio módulo. Um exemplo claro é apresentado através da Figura 4.25 que ilustra a apresentação do conteúdo que informa as transações pertencentes aos grupos de acesso, como visto anteriormente, esta relação, define o perfil de acesso do usuário do sistema.

	RH_BENEFICIO	RH_BESPE	RH_CADASTRO	RH_CAPACITACAO	RH_CAPACITACAOCURSOS	RH_CAPACITACOESEXTERNAS	RH_CID	RH_CONTRATOS	RH_ENQUADRAMENTO	RH_FUNCAOALTERACAO	RH_FUNCAOCONSULTA	RH_GERAL	RH_GESTOR	RH_IMAGEM	RH_PERIODICOS	RH_PROADESADMIN	RH_PROADESCONSULTA	RH_PROADESPRIMEIRAFASE	RH_PROADESSEGUNDAFASE	RH_RECRUTAMENTO	RH_RELATORIOS	RH_REMUNERACAO	RH_SAUDE	RH_SECRETARIA	RH_SEGURANCA
RELTERCEIRIZADOS								•																	
RH	•		•	•			•	•				•	•	•						•		•	•	•	•
RH_ADICIONAL																									•
RH_AMBIENTE			•	•																					
RH_APOSENTADORIA			•																						
RH_AVERBACAO			•																						
RH_BESPE		•																							
RH_CADASTRO_SERVIDOR			•																						
RH_CAPACITACAO				•																					
RH_CAPACITACAOCURSOS					•																				
RH_CAPACITACOESEXTERNAS						•																			
RH_CESSAO																				•					
RH_CID							•																•		
RH_ETIQUETAS														•											

Figura 4.25 - Identificação de perfis de acesso no sistema

Ainda existem formulários dados para consulta ocasional e não disponibilizam nenhum tipo de saída em arquivo, porém sintetizam diversos dados no sistema envolvendo até mesmo dados que são importados periodicamente do Sistema Integrado de Administração Pessoal (SIAPE), como é o caso do relatório de quantitativos que faz um resumo geral do numerário do quadro de servidores da universidade como ilustra a Figura 4.26. No campo Ano/Mês/Registro, o servidor pode consultar a situação do

numerário em meses anteriores, o número seguinte ao Ano/Mês, separado por um ifem indica o número total de registro advindos do SIAPE.

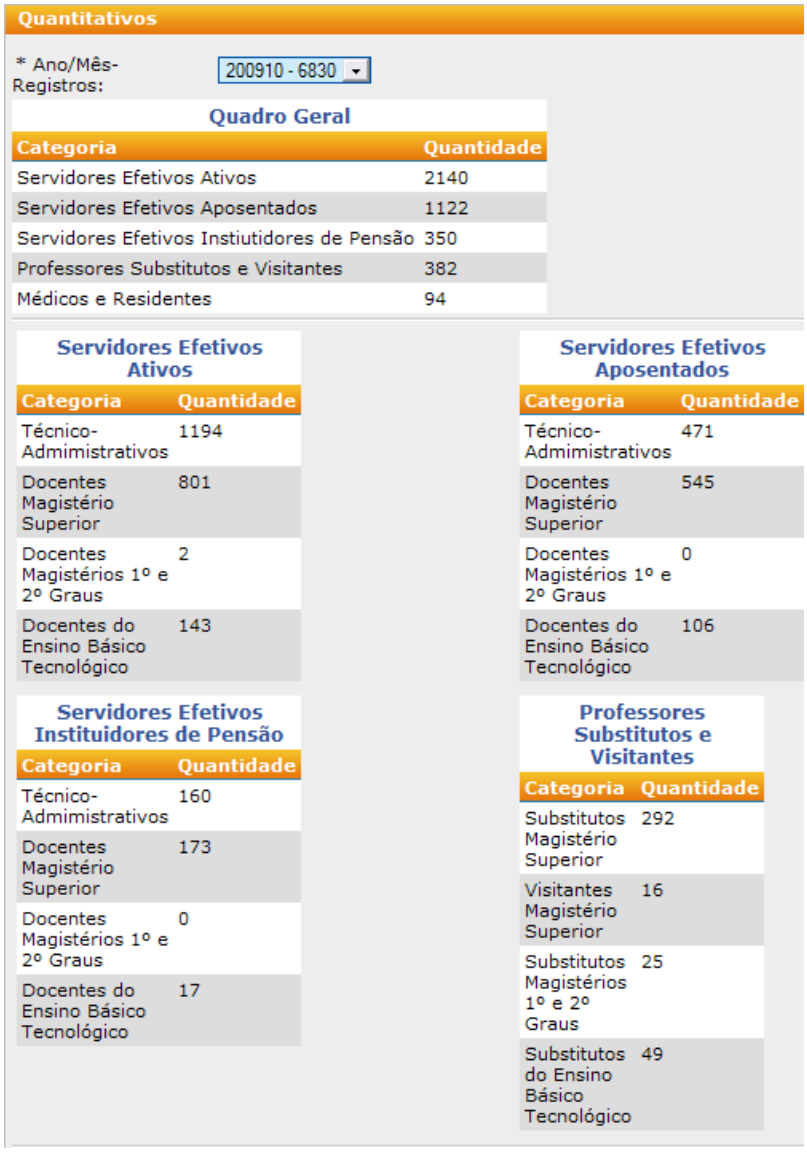


Figura 4.26 – Painel de visualização de dados institucionais

4.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o SIGA e algumas funcionalidades do SIGA-RH procurando mostrar o resultado final do processo de um desenvolvimento baseado em tecnologias de software livre adotados pela UFJF.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A realização deste trabalho possibilitou apresentar uma pequena parte das questões inter-relacionadas ao desenvolvimento de um sistema de informações capaz de gerenciar, com qualidade num ambiente de grande diversidade, a gestão de pessoal em uma instituição de ensino superior como a UFJF.

Devido ao grande número de acessos ao sistema de informação da UFJF, que possui papel fundamental no controle administrativo da instituição e no auxílio ao aprendizado, garantir a segurança sem perder eficiência, através de redes de computadores, especialmente a internet, nos tempos atuais, é uma demanda imperativa.

Com o auxilio de diversas tecnologias o projeto MIOLO busca criar soluções para desafios de desenvolvimento auxiliando a criação e manutenção de novas aplicações web. Tecnologias bem consolidadas como servidor web Apache, linguagem de programação PHP e Javascript, linguagem de marcação XML, definições de estilo como CSS, foram úteis para garantir o sucesso do projeto.

Novas tecnologias são constantemente aplicadas, mesmo que signifiquem apenas mudanças de abordagem ou paradigma, como por exemplo, uso de AJAX, *Tableless*, SPI e persistência de objetos.

Mas para que essas tecnologias trabalhem em harmonia e tenham o resultado final desejado, sem desvio do ideal inicial da construção de sistemas, há que se basear em pontos, algumas vezes, abstratos, mas tão importantes quanto qualquer outra tecnologia. Os padrões de projetos, de forma estrutural e preempitivamente, indicam o caminho a se tomar e os contornos aos desafios já conhecidos, a modelagem de dados classifica regras importantes as informações.

Os modelos de casos de usos, e os diagramas da UML facilitam o entendimento global do sistema e ajudam a documentação. O paradigma da programação orientada a objetos é executado, sem maiores problemas, mesmo em equipes com número de desenvolvedores elevando, pois conta com o auxílio de ferramenta de controle de versão como o Subversion.

Com tantas questões embutidas ao SIGA, o desenvolvimento de novos módulos e áreas acabam sendo tarefas contínuas. Nas áreas ensino, administração e recursos humanos a dinâmica e renovação é uma premissa permanente no dia a dia dos que trabalham na manutenção e para as centenas de pessoas que usam diariamente o sistema.

A área de Recurso Humano apresenta condições e funcionalidades de forma a possibilitar um controle do histórico funcional de servidores e outras pessoas que estão vinculadas à universidade.

Algumas dessas funcionalidades, como por exemplo, obtenção de dados pessoais e funcionais para transmitir informações aos funcionários, marcação de férias, controles de licenças e afastamentos são usados praticamente diariamente. A geração de relatório de apoio é importante quando o número de pessoas para se controlar é elevado.

Como este trabalho mostrou apenas partes das tecnologias e padrões de desenvolvimento, vale deixar como sugestão e incentivar novas pesquisas que apontem outros aspectos desta mesma área, enriquecendo o assunto e servindo de base para outros acadêmicos que se dediquem a construção de sistemas semelhantes.

REFERÊNCIAS BIBLIOGRÁFICAS

AGROPIC. **Rastre**. Disponível em: <http://d35568.tinf28.tuganet.info/seccoes.aspx?id_seccao=42&ord=2>. Último acesso em: 5 dez. 2009.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 2. Ed. Rio de Janeiro : Editora Campus/Elsevier. 2007.

BIRRER, Andreas; EGGENSCHWILER, Thomas. **Frameworks in the financial engineering domain: An experience report**. Alemanha: [S.n.]. 1993.

BRASIL. Portaria Normativa N.º 2, de 27 de outubro de 2000. **Dispõe sobre o SIASG e Instruções Gerais e Procedimentos.** [Online]: http://www.comprasnet.gov.br/legislacao/portarias/p02_00.htm.

COLLINS-SUSSMAN, Ben; FITZPATRICK, Brian W.; PILATO, C. Michael. **Version Control with Subversion: For Subversion 1.4: (Compiled from r2866)**. California. TBA. 2007. 348p.

DEITEL, H. M.; DEITEL, P. J. **Java, como programar**. 3. Ed. Porto Alegre : Bookman. 2001.

DEUTSCH, L. Peter. **Design reuse and framework sin the Smalltalk-80 system. Software Reusability, Volume II: Applications and Experience.** [S.l.] : Addison-Wesley. 1989.

FERNANDES, Leila Maria Pinheiro; OLIVEIRA, Antônio Ricardo de; TAVEIRA, Gilda Aché. **Modelagem de Dados**. 1. Ed. Rio de Janeiro : Senac Nacional. 2000. 80p.

GAMMA, Erich; *et al*. **Padrões de Projeto: soluções reutilizáveis de software orientado a objetos**. Porto Alegre : Bookman, 2005. 364p.

GARRET, J. J. **Ajax: A New Approach to Web Applications**. 2005. Disponível em: <http://adaptivepath.com/ideas/essays/archives/000385.php>. Último acesso em: 28 jun. 2009.

GÄRTNER, Vilson; MATOS, Ely Edison da Silva. **Miolo 2.0 User Guide**. Versão do documento 1.2. [S.l. : s.n.]. 2008.

JOHNSON, Ralph E.; MCCONNELL, Carl; LAKE, I. Michael. **The RTL system: A framework for code optimization**. Alemanha : ACM-Press. 1992.

JOHNSON, Ralph. **ObjectOriented Programming System, Languages, and Applications Conference Proceedings. Documenting frameworks using patterns**. Vancouver : ACM Press. 1992.

JUNIOR, Cleuton Sampaio de Melo. **Guia do Java: Enterprise Edition 5: desenvolvendo aplicações corporativas.** Rio de Janeiro : Brasport. 2007. 182p.

JUNIOR, Nilson de Souza Rego. **Persistência de Dados**. Disponível em: http://www.ietf.org/rfc/rfc3875.txt. Último acesso em: 29 nov. 2009.

LALLI, Felipe Micoroni; BUENTO, Felippe Franco; ZACHARIAS, Guilherme Keese. **Evolução da programação *web***. Campinas. 2008. Trabalho de conclusão de curso (Cientista da Computação) – Unidade III da rede Anhanguera Educacional.

LAUDON, K. C., LAUDON, J. P. **Sistemas de Informação Gerencias: Administrando a empresa digital**. 5. Ed. São Paulo : Prentice-Hall. 2004. 584p.

LEITE, Mário. **Técnicas de Programação: uma abordagem moderna**. Rio de Janeiro : Brasport. 2006. 405p.

MALHEIROS, Marcelo de Gomensoro. **Sistema de Controle de Versão.** Apresentação no V Seminário de desenvolvimento em software livre. Univates. 2005.

MARCELO, Antônio. **Apache: configurando o servidor WEB para Linux**. 3. Ed. Rio de Janeiro : Brasport. 2005.

MATOS, Ely Edison da Silva. **Framework MIOLO 2.5**. 2009. Disponível em: <svn://svnext.ufjf.br/miolo/trunk>. Último acesso em: 28 jun. 2009.

MATOS, Ely Edison da Silva. **Sistema Integrado de Gestão Acadêmica: a experiência da UFJF**. Disponível [Online] em: <svn://svnext.ufjf.br/miolo/trunk> , Juiz de Fora. 2009.

MATOS, Ely Edison da Silva; RIBEIRO, Carlos Alberto. **Sistema Integrado de Gestão Acadêmica: a experiência da UFJF**. Disponível [Online] em: http://www.viadigital.org.br/index.php?option=com_docman&task=doc_download&gid=27, Juiz de Fora. 2007.

MAULER, Mauro José Alvim. **Educação à Distância Aplicada a Capacitação de Recursos Humanos: Uma proposta de curso à distância para os secretários das unidades da Universidade Federal de Juiz de Fora.** Juiz de Fora. 2003. 43p. Monografia (Especialista em gestão da educação à distância) - Coordenação de Pós-Graduação, Universidade Federal de Juiz de Fora.

MAULER, Mauro José Alvim. **Manual SIGA-RH Versão Secretários.** Universidade Federal de Juiz de Fora. Juiz de Fora. 2008. 20p.

NETCRAFT. **About NetCraft.** Disponível em: <http://news.netcraft.com/about-netcraft>. Último acesso em: 16 nov. 2009.

RFC 793: **Transmission Control Protocol**. DARPA INTERNET PROGRAM, Disponível [Online]: www.ietf.org/rfc/rfc0793.txt. Setembro 1981.

RUDOY, Artyom. Persistence Layer. Disponível em: <http://artyomr.narod.ru> Último acesso em: 5 dez. 2009.

SCOTT W. Ambler. The fundamental of Mapping Objects to Relational Databases. Disponível em: <http://www.agiledata.org/essays/mappingObjects.html> Último acesso em: 5 dez. 2009.

SIGA. **SIGA 2.0 – Módulos.** Versão do documento 3. Documento interno. 2009.

SILVA, Gleydson Mazioli da. **Guia Foca GNU/Linux. Capítulo 12 – Apache.** Versão 6.42. 2007. Disponível em: <http://focalinux.cipsga.org.br/guia/avancado/ch-s-apache.htm>. Último acesso em: 4 dez. 2009.

TEIXEIRA, Thiago Nery. **Modelo SPI (Single Page Interface) para desenvolvimento de aplicações web.** Juiz de Fora. 2009. 85p. Monografia (Bacharel em Ciência da Computação – Exatas) - Departamento de Ciência da Computação, Universidade Federal de Juiz de Fora.

TITTEL, Ed. **XML Teoria e problemas de XML**. [S. l.] : Bookman. 2002. 202p.

VASCO, Furtado. **Tecnologia e Gestão da Informática na Segurança Pública**. [S.l.] : Garamond Ltda. 2002.

VLISSIDES J. M.; LINTON M. A. **A framework for building domain-specific graphical editors**. [S.l. : s.n.]. 1990.

W3C. **HTML 4.01 Specification**. 1999a. Disponível em: <http://www.w3.org/TR/REChtml40>. Último acesso em: 28 jun. 2009.

Apêndice 1 – Árvore do organograma administrativo da UFJF

Visão hierárquica contendo parte das unidades do organograma administrativo da UFJF disponibilizada através de recursos do MIOLO no SIGA-RH.

Figura A1.1 Setores do organograma administrativo da UFJF

Continuação da visão de parte do organograma administrativo da UFJF.

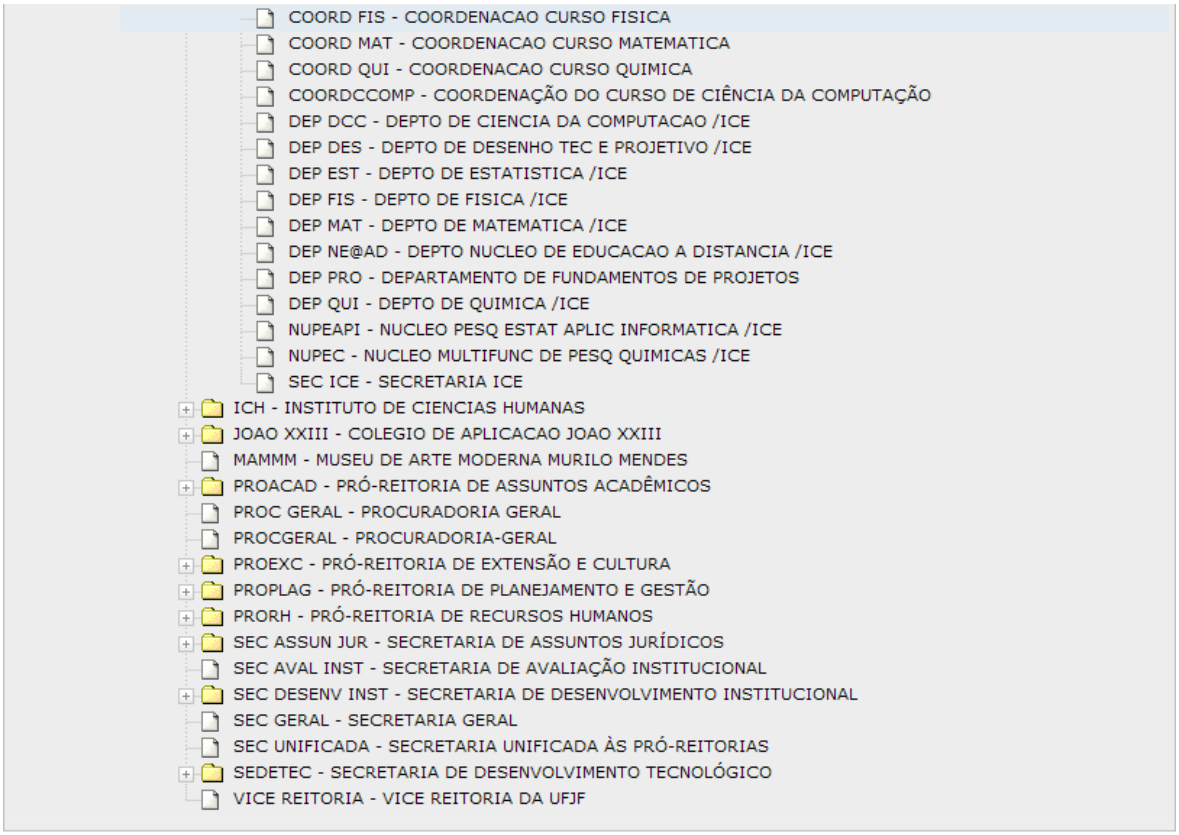


Figura A1.2 Continuação dos Setores do organograma administrativo da UFJF

Anexo 1 – Árvore de Controles do MIOLO

Apresentação da hierarquia da árvore de controles do MIOLO.

```
          +--MStyle
          +--MAttributes
          +--MdragDropControl
          +--+--MDraggable
          +--+--MDroppable
          +--MComponent
          +--+--MControl
          +--+--+--MPage
          +--+--+--MDiv
          +--+--+--+--MHr
          +--+--+--+--MCSSBox
          +--+--+--+--MBoxTitle
          +--+--+--+--MModuleHeader
          +--+--+--+--MEditor
          +--+--+--+--MSeparator
          +--+--+--+--MSpacer
          +--+--+--+--MContainerControl
          +--+--+--+--+--MBox
          +--+--+--+--+--MAccordion
          +--+--+--+--+--MContainer
          +--+--+--+--+--+--MHContainer
          +--+--+--+--+--+--MVContainer
          +--+--+--+--+--+--MBasePanel
          +--+--+--+--+--+--+--MPanel
          +--+--+--+--+--+--+--+--MActionPanel
          +--+--+--+--+--+--MBaseGroup
          +--+--+--+--+--+--+--MRadioButtonGroup
          +--+--+--+--+--+--+--MCheckBoxGroup
          +--+--+--+--+--+--+--MLinkButtonGroup
          +--+--+--+--+--+--+--MToolBar
          +--+--+--+--+--MTabContainer
          +--+--+--+--+--MBaseGrid
          +--+--+--+--+--+--MGrid
          +--+--+--+--+--+--+--MDataGrid
          +--+--+--+--+--+--+--+--MDataGrid2
          +--+--+--+--+--+--+--MLookupGrid
          +--+--+--+--+--+--+--+--MLookupObjectGrid
          +--+--+--+--+--+--+--+--MLookupQueryGrid
          +--+--+--+--+--+--+--MObjectGrid
          +--+--+--+--+--+--+--MPDFReport
          +--+--+--+--+--+--+--MGridAjax
          +--+--+--+--+--MBaseForm
          +--+--+--+--+--+--MForm
          +--+--+--+--+--+--+--MFormAjax
          +--+--+--+--+--+--+--MCompoundForm
          +--+--+--+--+--+--+--MCSSForm
          +--+--+--+--+--+--+--MCSSPForm
          +--+--+--+--+--+--+--MIndexedForm
          +--+--+--+--+--+--+--MTabbedForm
          +--+--+--+--+--MTheme
          +--+--+--+--+--MThemeElement
          +--+--+--+--+--MThemeBox
          +--+--+--MTabbedFormPage
          +--+--+--MTableCell
```

```
          +--+--+--MTableRow
          +--+--+--MTable
          +--+--+--+--MSimpleTable
          +--+--+--+--+--MTableRaw
          +--+--+--+--+--+--MTableXML
          +--+--+--MContent
          +--+--+--+--MFileContent
          +--+--+--MContentHeader
          +--+--+--MAreaContainer
          +--+--+--MIFrame
          +--+--+--MImageFormLabel
          +--+--+--MGridColumn
          +--+--+--+--MGridHyperlink
          +--+--+--+--+--MDataGridHyperlink
          +--+--+--+--+--MObjectGridHyperlink
          +--+--+--+--MGridControl
          +--+--+--+--+--MDataGridControl
          +--+--+--+--+--MObjectGridControl
          +--+--+--+--+--MPDFReportControl
          +--+--+--+--MDataGridColumn
          +--+--+--+--MObjectGridColumn
          +--+--+--+--MPDFReportColumn
          +--+--+--MGridAction
          +--+--+--+--MGridActionIcon
          +--+--+--+--MGridActionText
          +--+--+--+--MGridActionSelect
          +--+--+--+--MDataGridAction
          +--+--+--+--MObjectGridAction
          +--+--+--MGridFilter
          +--+--+--+--MGridFilterText
          +--+--+--+--MGridFilterSelection
          +--+--+--+--MGridFilterControl
          +--+--+--MGridNavigator
          +--+--+--MJavascriptCode
          +--+--+--MFormControl
          +--+--+--+--MInputControl
          +--+--+--+--+--MActionControl
          +--+--+--+--+--+--MButton
          +--+--+--+--+--+--+--MButtonAjax
          +--+--+--+--+--+--+--MInputButton
          +--+--+--+--+--+--+--+--MButtonFind
          +--+--+--+--+--+--+--MButtonImage
          +--+--+--+--+--+--+--MButtonWindow
          +--+--+--+--+--+--+--MDropDownButton
          +--+--+--+--+--+--MLink
          +--+--+--+--+--+--+--MActionHyperLink
          +--+--+--+--+--+--+--MLinkBack
          +--+--+--+--+--+--+--MOpenWindow
          +--+--+--+--+--+--+--MGridHeaderLink
          +--+--+--+--+--+--+--MLinkButton
          +--+--+--+--+--+--+--+--MButtonClose
          +--+--+--+--+--+--+--+--MButtonHelp
          +--+--+--+--+--+--+--+--MButtonMinimize
          +--+--+--+--+--+--+--MImageLink
          +--+--+--+--+--+--+--+--MImageLinkLabel
          +--+--+--+--+--+--+--+--MImageButton
          +--+--+--+--+--+--+--+--+--MImageButtonLabel
          +--+--+--+--+--MTextField
          +--+--+--+--+--+--MCalendarField
          +--+--+--+--+--+--MCurrencyField
```

```
          +--+--+--+--+--+--MPasswordField
          +--+--+--+--+--+--MFileField
          +--+--+--+--+--+--MHiddenField
          +--+--+--+--+--+--MMultiLineField
          +--+--+--+--+--+--+--MmultiTextField3
          +--+--+--+--+--+--+--+--MmultiTextField4
          +--+--+--+--+--+--MmultiTextField2
          +--+--+--+--+--+--MLookupField
          +--+--+--+--+--+--+--MLookupTextField
          +--+--+--+--+--+--+--MLookupFieldValue
          +--+--+--+--+--MChoiceControl
          +--+--+--+--+--+--MCheckBox
          +--+--+--+--+--+--MRadioButton
          +--+--+--+--+--MListControl
          +--+--+--+--+--+--MSelection
          +--+--+--+--+--+--+--MComboBox
          +--+--+--+--+--+--MMultiSelection
          +--+--+--+--+--+--MMultiSelectionField
          +--+--+--+--+--+--MOrderedList
          +--+--+--+--+--+--MUnOrderedList
          +--+--+--+--+--MOption
          +--+--+--+--+--MInputGridColumn
          +--+--+--+--+--MInputGrid
          +--+--+--+--+--MPrompt
          +--+--+--+--+--MIndexedControl
          +--+--+--+--MOutputControl
          +--+--+--+--+--MSpan
          +--+--+--+--+--+--MHint
          +--+--+--+--+--MBaseLabel
          +--+--+--+--+--+--MPageComment
          +--+--+--+--+--+--MSeparator
          +--+--+--+--+--+--MLabel
          +--+--+--+--+--+--+--MRawText
          +--+--+--+--+--+--+--MFieldLabel
          +--+--+--+--+--+--+--MTextHeader
          +--+--+--+--+--+--+--MText
          +--+--+--+--+--+--+--+--MTextLabel
          +--+--+--+--+--+--+--+--MGDText
          +--+--+--+--MImage
          +--+--+--+--+--MImageLabel
          +--+--+--+--MDragDrop
          +--+--+--+--MHtmlForm
          +--+--+--+--MTextTable
          +--+--+--+--MtoolBarButton
          +--+--+--+--MTreeMenu
          +--+--+--MValidator
          +--+--+--MWindow
          +--+--+--MStatusBar
          +--+--+--MOptionGroup
          +--+--+--MOptionListItem
          +--+--+--MOptionList
          +--+--+--+--MDhtmlMenu2
          +--+--+--+--MMenu
          +--+--+--+--+--MNavigationBar
```

Anexo 2 – Árvore de diretórios do MIOLO

Apresentação da árvore de diretórios criada após a instalação do MIOLO. Esses diretórios contêm as classes do framework, bem como extensões, contribuições, arquivos de configurações e documentação.

```
-----<diretório-base>(p.ex. /usr/local/miolo)
|
+--- classes
|      +--- contrib
|      +--- database
|      +--- doc
|      +--- etc
|      +--- extensions
|      +--- ezpdf
|      +--- flow
|      +--- model
|      +--- persistence
|      +--- pslib
|      +--- security
|      +--- services
|      +--- ui
|            +----+--- controls
|            +----+--- painter
|            +----+--- report
|            +----+--- themes
|                        | +--- miolo2
|                        | +--- system
|                        | +--- clean
|                        | +--- .....
|      +--- utils
|
+--- docs
+--- etc
|      +--- miolo.conf
|      +--- passwd.conf
|      +--- mkrono.conf
+--- html
|      +--- downloads
|      +--- images
|      +--- reports
|      +--- scripts
+--- locale
+--- modules
|      +--- admin
|      +--- tutorial
|      +--- helloworld
|      +--- hangman
|      +--- modulo1
|      +--- ....
+--- var
+--- db
+--- log
+--- report
+--- trace
```

Anexo 3 – Árvore de diretório de módulos do MIOLO

A seguir, podemos observar um exemplo da estrutura de diretórios formado por um módulo típico do MIOLO.

```
-----<diretório-base>(p.ex. /usr/local/miolo)
|
+--- modules
|      +--- module1
|      |     +--- classes
|      |     +--- forms
|      |     +--- menus
|      |     +--- sql
|      |     +--- handlers
|      |     +--- grids
|      |     +--- reports
|      |     +--- inc
|      |     +--- etc
|      |     +--- html
|      |     |     +--- images
|      |     |     +--- files
|      |     +--- ui
|      |     +--- controls
|      |     +--- themes
|      |
|      +--- module2
|      |
|      +--- module3
|      .
|      .
|      .
.
.
.
```

Anexo 4 – Modelagem de algumas tabelas/classes do MIOLO

Este anexo ilustra o trecho de código para exemplo do uso de persistência conforme apresentado no capítulo 3.

O diagrama de classe da Figura A4.1, apresenta a descrição que precisamos para as classes *User* e *Sector*, usadas no exemplo à persistência.

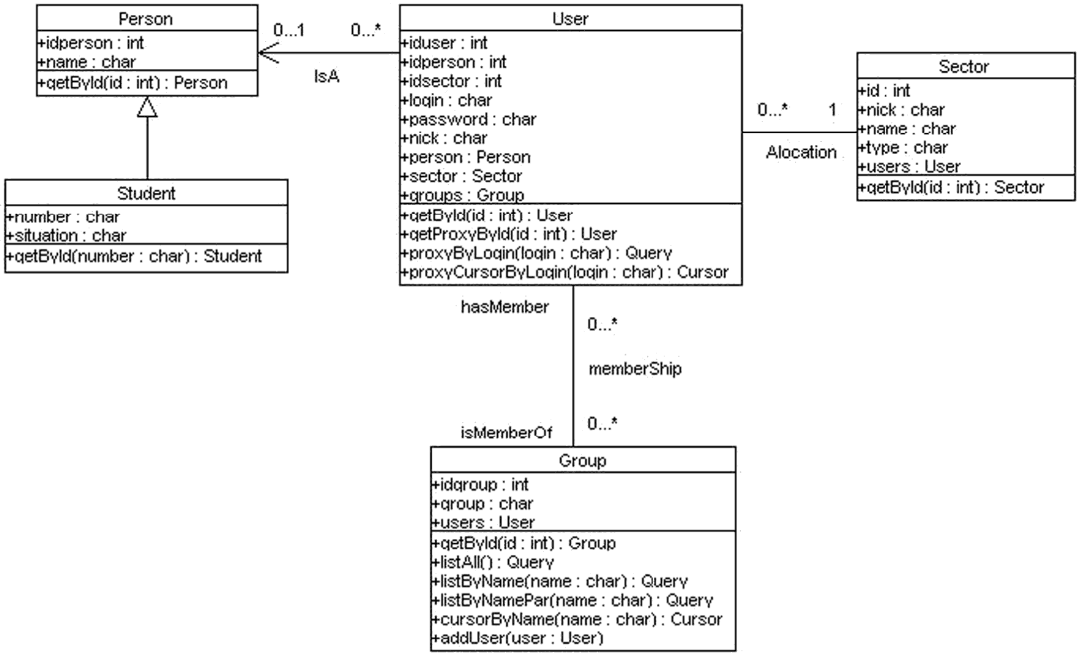


Figura A4.1 - Diagrama de Classe

O DTR da Figura A4.2 apresenta a forma como os objetos são persistidos no banco de dados relacional.

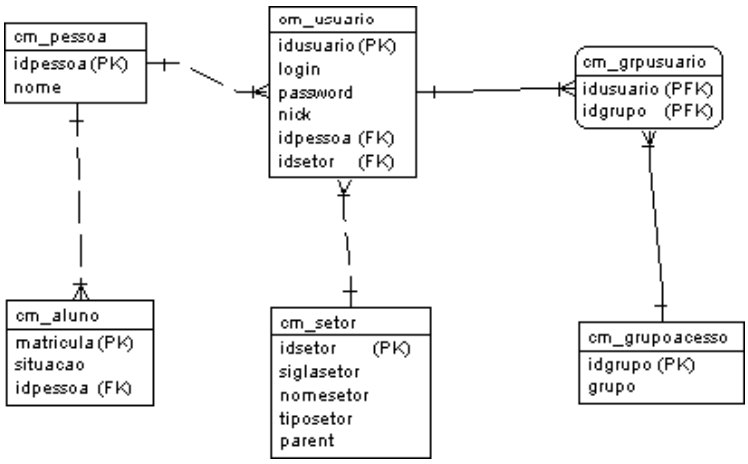


Figura A4.2 - Esquema do banco de dados relacional

O mapeamento dos objetos para posterior persistência em banco de dados relacional é parte fundamental no desenvolvimento de aplicações com MIOLO, então, abaixo apresentamos o mapeamento das classes *User* e *Sector* usadas no exemplo de uso de persistência de objetos no capítulo 3 deste trabalho.

Classe *User*:

```
<map>
   <moduleName>persistence</moduleName>
   <className>user</className>
   <tableName>cm_usuario</tableName>
   <databaseName>admin</databaseName>
   <attribute>
      <attributeName>iduser</attributeName>
      <columnName>idusuario</columnName>
      <key>primary</key>
      <idgenerator>seq_cm_usuario</idgenerator>
   </attribute>
   <attribute>
      <attributeName>login</attributeName>
      <columnName>login</columnName>
      <proxy>true</proxy>
   </attribute>
   <attribute>
      <attributeName>password</attributeName>
      <columnName>password</columnName>
   </attribute>
   <attribute>
      <attributeName>nick</attributeName>
      <columnName>nick</columnName>
   </attribute>
   <attribute>
      <attributeName>idperson</attributeName>
      <columnName>idpessoa</columnName>
      <proxy>true</proxy>
   </attribute>
   <attribute>
      <attributeName>idsector</attributeName>
      <columnName>idsetor</columnName>
      <proxy>true</proxy>
   </attribute>
   <attribute>
      <attributeName>person</attributeName>
   </attribute>
   <attribute>
      <attributeName>groups</attributeName>
   </attribute>
   <attribute>
      <attributeName>sector</attributeName>
   </attribute>

   <association>
      <toClassModule>persistence</toClassModule>
      <toClassName>person</toClassName>
      <cardinality>oneToOne</cardinality>
      <target>person</target>
      <retrieveAutomatic>true</retrieveAutomatic>
```

```
            <saveAutomatic>true</saveAutomatic>
            <entry>
               <fromAttribute>idperson</fromAttribute>
               <toAttribute>idperson</toAttribute>
            </entry>
         </association>
         <association>
            <toClassModule>persistence</toClassModule>
            <toClassName>group</toClassName>
            <associativeClassModule>persistence</associativeClassModule>
            <associativeClassName>groupuser</associativeClassName>
            <cardinality>manyToMany</cardinality>
            <target>groups</target>
            <retrieveAutomatic>false</retrieveAutomatic>
            <saveAutomatic>false</saveAutomatic>
            <direction>
               <fromAttribute>users</fromAttribute>
               <toAttribute>groups</toAttribute>
            </direction>
          </association>

          <association>
            <toClassModule>persistence</toClassModule>
            <toClassName>sector</toClassName>
            <cardinality>oneToOne</cardinality>
            <target>sector</target>
            <retrieveAutomatic>true</retrieveAutomatic>
            <saveAutomatic>false</saveAutomatic>
            <entry>
               <fromAttribute>idsector</fromAttribute>
               <toAttribute>id</toAttribute>
            </entry>
         </association>
      </map>
```

Classe *Sector*:

```
      <map>
         <moduleName>persistence</moduleName>
         <className>sector</className>
         <tableName>cm_setor</tableName>
         <databaseName>admin</databaseName>
         <attribute>
            <attributeName>id</attributeName>
            <columnName>idsetor</columnName>
            <key>primary</key>
         </attribute>
         <attribute>
            <attributeName>nick</attributeName>
            <columnName>siglasetor</columnName>
         </attribute>
         <attribute>
            <attributeName>name</attributeName>
            <columnName>nomesetor</columnName>
         </attribute>
         <attribute>
            <attributeName>type</attributeName>
            <columnName>tiposetor</columnName>
         </attribute>
```

```
            <attribute>
               <attributeName>users</attributeName>
            </attribute>
            <association>
               <toClassModule>persistence</toClassModule>
               <toClassName>user</toClassName>
               <cardinality>oneToMany</cardinality>
               <target>users</target>
               <retrieveAutomatic>true</retrieveAutomatic>
               <saveAutomatic>false</saveAutomatic>
               <deleteAutomatic>true</deleteAutomatic>
               <inverse>true</inverse>
               <entry>
                  <fromAttribute>idsector</fromAttribute>
                  <toAttribute>id</toAttribute>
               </entry>
            </association>
         </map>
```

Anexo 5 – Diagrama de Casos de Uso

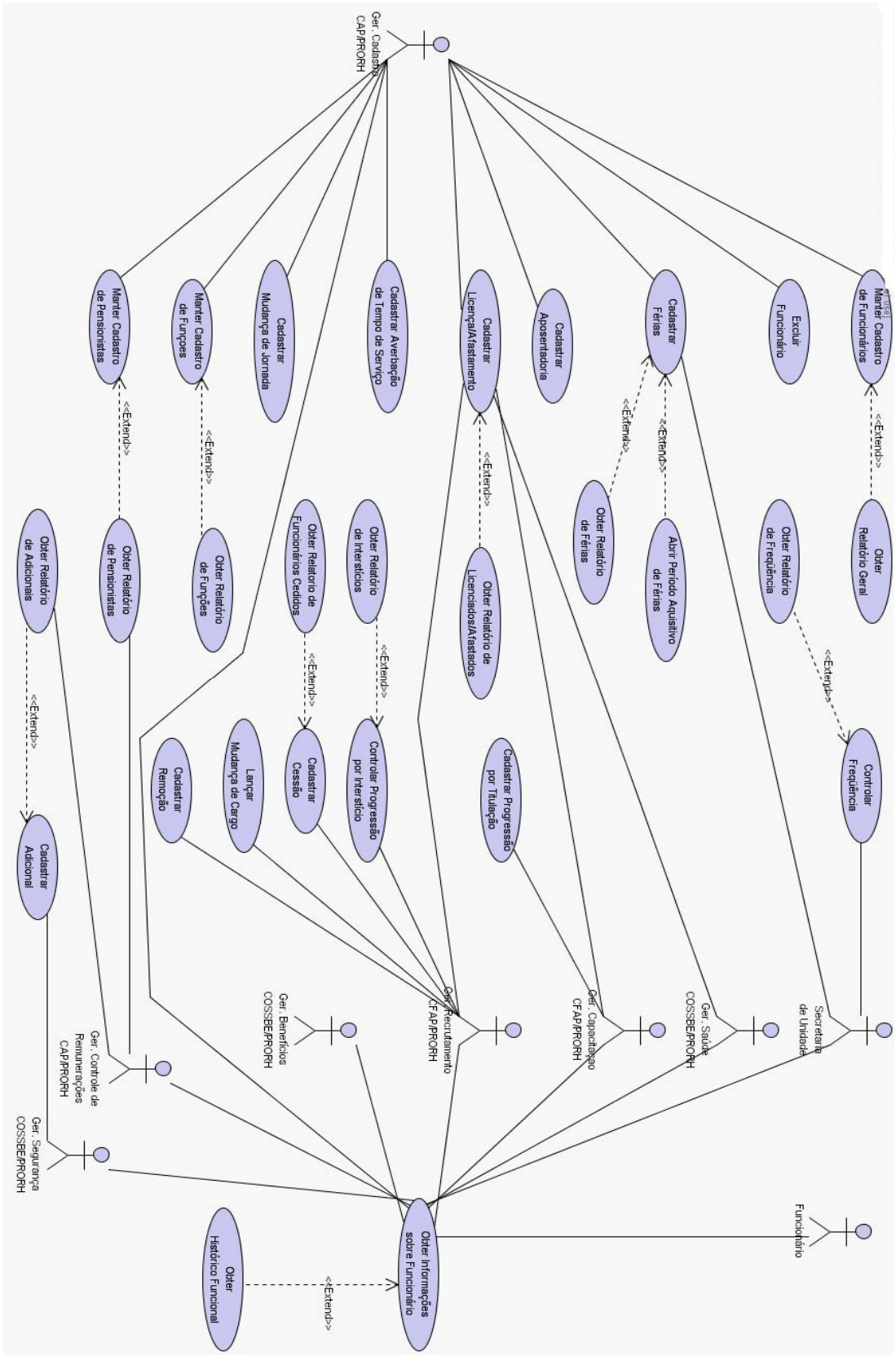


Figura A5.1 - Diagrama de Classe do SIGA-RH

Anexo 6 – Casos de Uso de Freqüência

Na descrição das funcionalidades do módulo de recursos humanos do SIGA da UFJF no capítulo 4 deste trabalho são abordados os diversos relatórios propiciados pelo sistema e entre eles é citado o "Mapa de Freqüência". Abaixo segue documento interno descrevendo o caso de uso (seção 4.3.1.3) referente a esta implementação.

	Controlar Freqüência

Sumário:
Controle de freqüência dos funcionários.

Ator:
Procedimento de responsabilidade das Secretarias de Unidade.

Fluxo:

1. Usuário solicita visualização de freqüência anual de determinado funcionário, informando matrícula e ano de referência.
2. Sistema exibe mapa de freqüência do funcionário, relativa ao ano escolhido, facultando ao usuário requisitar sua impressão.
3. Usuário solicita ou não a impressão do mapa de freqüência e encerra o caso de uso.

Nota de Implementação:

- O mapa de freqüência anual consiste em um demonstrativo baseado no calendário do ano escolhido pelo usuário, conforme exemplo abaixo, exibindo os mnemônicos associados às ocorrências de freqüência do funcionário (licenças/afastamentos), para cada dia do ano, sendo que, para os dias em que não houver ocorrências registradas, o mnemônico será 'N' (freqüência normal).

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
JANEIRO	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.
FEVEREIRO	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	FER	FER	FER	FER	FER	FER	FER	
	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER		
MARÇO	FER	FER	FER	FER	FER	FER	FER	FER	FER	.N.	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.
ABRIL	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
MAIO	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.
JUNHO	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
JULHO	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.
AGOSTO	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.
SETEMBRO	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
OUTUBRO	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.
NOVEMBRO	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	
DEZEMBRO	FER	FER	FER	FER	FER	FER	FER	FER	FER	FER	.N.	.N.	.N.	.N.	.N.	
	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.	.N.

Sumário:
Obtenção de relatório mensal contendo as ocorrências de freqüência dos funcionários.

Ator:
Procedimento de responsabilidade das Secretarias de Unidade.

Fluxo:

.

1. Usuário solicita geração do relatório de freqüência de sua Unidade.
2. Sistema exibe relatório de freqüência, facultando ao usuário requisitar sua impressão e/ou gravá-lo em seu computador.
3. Usuário escolhe uma das opções (ou as duas) e/ou encerra o caso de uso.

Nota de Implementação:

O relatório mensal de freqüência da Unidade deve consistir em uma lista dos funcionários nela lotados, devendo conter campos relativos aos dias do mês, cada qual contendo o mnemônico relativo à respectiva ocorrência de freqüência (ou indicação de freqüência normal, com o mnemônico 'N'). Na prática, todas as informações serão extraídas dos registros de licenças/afastamentos de cada funcionário. O relatório deverá, ainda, conter legendas com os significados dos mnemônicos exibidos. Abaixo, segue sugestão de *layout*:

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
MAPA DE CONTROLE DE FREQÜÊNCIA
MÊS/ANO: NOVEMBRO/2004
UNIDADE: CGCO/DSI

Matrícula	Nome	1	2	3 4 5 6 7 8 9 10 11 12 13 14 15 16	28	29	30	31
1150942	Mauro José Alvim Mauler	N	N	.....	N	N	N	N
.....		.....						

LEGENDA:

N = FREQÜÊNCIA NORMAL
FER = FÉRIAS
.....

Assinaturas:

_____ _____ _____

Secretário Diretor PRORH

Anexo 7 – Casos de Uso de Férias

Diante do disposto no capítulo 4, com relação à implementação das telas referentes ao lançamento de férias, destinada aos secretários de unidades da UFJF, podemos observar nos casos de uso abaixo a documentação do processo de desenvolvimento, inclusive com notas de codificação.

	Cadastrar Férias
Sumário: Inclusão/alteração/exclusão de registros relativos às férias anuais dos funcionários.	
Ator: Procedimento de responsabilidade da Gerência de Cadastro/PRORH, descentralizado para as Secretarias de Unidades no que se refere aos funcionários nelas lotados.	
Fluxo: . 1. Usuário solicita opção de lançamento de férias para funcionário. 2. Sistema abre ficha de férias do funcionário, contendo o histórico de usufruição (períodos já lançados), bem como os períodos aquisitivos em aberto (em relação aos quais não houve usufruição), e permitindo a inclusão de período(s), se for o caso. 3. Se houver período(s) aquisitivo(s) em aberto, usuário inclui período(s) de usufruição de férias, informando se o funcionário optou pelo recebimento de metade do 13º salário e/ou adiantamento da remuneração e encerra o caso de uso.	
Regras do Negócio: RN01 – As férias devem ser marcadas com antecedência mínima de 60 (sessenta) dias do mês previsto para usufruição. RN02 – O funcionário que opera direta e permanentemente com Raios-X ou substâncias radioativas deve obrigatoriamente usufruir 20 (vinte) dias de férias a cada 6 (seis) meses. RN03 – As férias de funcionário estatutário cujo direito de usufruição já foi adquirido podem ser acumuladas até o máximo de 2 (dois) períodos (exceto quando houver conflito com a RN02). O limite para início de usufruição (integral ou última parcela) das férias acumuladas de um exercício para o outro é o dia 31 de dezembro. RN04 – A usufruição de férias de funcionário estatutário pode ser parcelada em até 3 (três) etapas, de acordo com o seguinte critério:	

- Docentes
  - 15 – 15 – 15
  - 30 – 15
  - 15 – 30
  - 20 – 25
  - 25 – 20
- Técnico-Administrativos
  - 10 – 10 – 10
  - 15 – 15
  - 20 – 10
  - 10 – 20

RN05 – No caso de funcionário celetista, as férias devem ser usufruídas dentro dos 12 (doze) meses subseqüentes ao vencimento do período aquisitivo, em um só período, podendo ser excepcionalmente parceladas em dois períodos, sendo que um deles não pode ser inferior a 10 (dez) dias. Se as férias forem concedidas após o vencimento do prazo acima previsto, o empregador pagará em dobro a respectiva remuneração.

RN06 – Ao ser marcado o primeiro período de férias parceladas, devem também ser programados o(s) período(s) restantes do parcelamento.

RN07 – Quando houver parcelamento de férias, o pagamento do abono contitucional de férias (1/3 sobre a remuneração relativa ao período total de direito para cada período aquisitivo) deverá ser efetuado por completo quando da usufruição da primeira parcela, exceto no caso de funcionário abrangido pela RN02, que deverá receber o abono em cada período de usufruição e proporcionalmente aos dias correspondentes a este período.

RN08 – No caso de férias a serem usufruídas até o mês de junho de cada ano, o funcionário pode optar por receber metade da Gratificação Natalina (“13º salário”) no mês anterior ao de usufruição (do período integral ou da 1ª parcela).

RN09 – O funcionário pode optar por receber o adiantamento de 70% da remuneração relativa ao período em que usufruir férias. No caso de férias parceladas, o adiantamento pode ser requerido em qualquer das parcelas e é pago proporcionalmente ao número de dias programados no respectivo período.

RN10 – As férias só podem ser interrompidas por motivo de calamidade pública, comoção interna, convocação para júri, serviço militar ou eleitoral, ou por necessidade do serviço declarada pela autoridade máxima da Instituição. Neste caso, o período interrompido deverá ser posteriormente usufruído de uma só vez, respeitadas as regras RN03 e RN05.

RN11 – O funcionário celetista pode solicitar a conversão de 1/3 (um terço) do período de férias a que faz jus em abono pecuniário igual ao valor da remuneração que lhe seria devida nos dias correspondentes.

RN12 – As férias são consideradas como uma ocorrência de licença/afastamento.

RN13 – As férias não podem ser marcadas concomitantemente com outra ocorrência de licença/afastamento, exceto quando se tratar de afastamento para participação em programa de pós-graduação.

Notas de Implementação:

- O sistema deverá permitir aos Secretários de Unidade a marcação de férias para determinado funcionário apenas se a usufruição tiver início no segundo mês subseqüente ao de marcação (RN01). Exemplo: férias a serem usufruídas em dezembro só poderão ser marcadas no sistema até 31 de outubro. A marcação de férias sem observância deste prazo só poderá ser realizada por usuários com nível de acesso da Gerência de Cadastro/CAP/PRORH.

- As férias só poderão ser marcadas se houver período aquisitivo registrado e em aberto. Desta forma, o sistema deverá listar para o usuário os períodos aquisitivos disponíveis para o funcionário (não prescritos e cujos dias de direito de usufruição ainda não tenham sido integralmente utilizados). Se houver mais de um período aquisitivo em aberto, deverá ser permitida a marcação apenas em relação ao período aquisitivo mais antigo.

- No caso de funcionário estatutário, o início de usufruição das férias corresponderá a qualquer data a partir do início do período aquisitivo (e nunca antes disso). Para o celetista, o início de usufruição será obrigatoriamente uma data posterior à de vencimento (data fim) do período aquisitivo.

- O sistema deverá promover o controle do saldo de dias a serem usufruídos, não permitindo a marcação de período superior ao saldo (total de dias na tabela de períodos aquisitivos) e garantindo, no caso de parcelamento, a observância dos limites estabelecidos nas RN04 e RN05. O controle do saldo será feito mediante incrementação do campo DIASUSUFRUIDOS da tabela rh_periodoaquisitivo. O incremento corresponderá ao número de dias de usufruição relativo às férias marcadas.

- Em qualquer marcação de férias, o sistema deverá atribuir valores iguais aos campos relativos à data fim e à data fim prevista da tabela de férias.

- Quando houver necessidade de alteração de informações relativas às datas de início e fim de períodos de férias já marcados, o usuário deverá excluir o registro anterior e lançar novo registro. O registro já gravado somente poderá ser alterado no caso de interrupção de férias (vide abaixo).

- No caso de funcionário que estiver operando Raios-X (registro na tabela de adicionais com data fim em aberto), o sistema deverá permitir a marcação de 20 (vinte) dias de férias a cada 6 (seis) meses, independentemente do saldo vinculado ao período aquisitivo. No caso de professor efetivo (grupo de cargos = ‘060’; situação = ‘01’), este período pode ser de 25 (vinte e cinco) dias, desde que haja saldo (RN02).

- O sistema deverá facultar o lançamento de interrupção de períodos de férias já lançados, nos termos da RN16. Neste caso, o usuário deverá lançar a data de interrupção, anterior à data fim anteriormente informada, que será substituída, porém mantendo-se a data fim prevista já registrada, pois a diferença entre as duas datas servirá como identificação de férias interrompidas. O período

interrompido (a ser posteriormente usufruído de uma só vez) será deduzido da informação de DIASUSUFRUIDOS do respectivo período aquisitivo.

- Em qualquer marcação de férias, o sistema deverá verificar se há férias interrompidas para o funcionário (checando, no último período lançado, se há diferença entre a data fim e a data fim prevista), forçando o lançamento integral do período restante, a não ser que haja ocorrência de prescrição ou conflito com a RN02.

- Quando houver a marcação do primeiro período de férias parceladas de determinado funcionário, o sistema deverá atribuir automaticamente o valor 'S' ao campo relativo ao abono constitucional, na tabela de férias (RN07). Além disso, deverá forçar a marcação (previsão) das demais parcelas, conforme RN06, neste caso não permitindo a atribuição do abono constitucional (que assumirá o valor 'N'), a não ser no caso de operador de Raios-X, que terá a atribuição do valor 'S' ao abono nas duas parcelas obrigatórias de férias.

- Na marcação de férias de funcionário que opera Raios-X (possui adicional de Raios-X registrado no sistema, com data fim em aberto), o sistema deve permitir a marcação apenas dos 20 (vinte) dias obrigatórios a cada seis meses. No caso de professor (grupo de cargos = '060') que opera Raios-X, vale a regra de usufruição de no mínimo 20 (vinte) dias a cada seis meses, garantido o direito aos 45 (quarenta e cinco) dias de usufruição para cada período aquisitivo (parcelamento 20 – 25 ou 25 – 20).

- Tendo em vista as RN03 e RN05, o sistema não deve permitir a marcação de férias relativas a períodos aquisitivos prescritos (checagem através da informação de "data de prescrição" contida na tabela de períodos aquisitivos).

- O lançamento da opção de recebimento de metade da gratificação natalina (13º salário) deve ser permitido apenas no caso de marcação de férias integrais ou 1ª parcela, cujo início de usufruição se dará até o mês de junho (RN08).

- O lançamento da opção pelo recebimento do adiantamento da remuneração (RN09) deve ser permitidao ao usuário em qualquer marcação de férias, inclusive quando se tratar de um dos períodos de parcelamento.

- Apenas no caso de funcionário celetista (regime jurídico = 'CLT' ou 'CDT'), o sistema deverá permitir que haja a opção pela conversão de 1/3 das férias em abono pecuniário, com atribuição do valor 'S' ao campo ABONO da tabela de férias. Nos demais casos o campo será sempre alimentado com o valor 'N' (RN11)

- Além do registro na tabela própria, a marcação de férias também devem ser registradas na tabela de licenças/afastamentos – ocorrência '045', '144', '507' ou '508', dependendo da situação funcional – respectivamente, CLT20, EST01, CDT12 ou EST04 (RN12).

- Ao ser solicitada pelo usuário a marcação de férias para o funcionário, o sistema

deverá verificar se o mesmo já não se encontra com registro de outro tipo de licença/afastamento abrangendo o mesmo período (checagem na tabela de licenças/afastamentos), não permitindo, neste caso, a referida marcação, exceto no caso das ocorrências de afastamento ‘130’, ‘400’, ‘402’, ‘403’, ‘405’, ‘406’, ‘500’, ‘502’, ‘503’, ‘505’, ‘506’ (casos excepcionais em que pode ocorrer a sobreposição da marcação de férias).

Anexo 8 – Casos de Uso para Períodos Aquisitivos

	Abrir Período Aquisitivo de Férias

Sumário:
Registro do período de exercício do funcionário pelo qual ele fará jus à usufruição de férias.

Ator:
Procedimento exclusivo da Gerência de Cadastro/CAP/PRORH.

Fluxo Principal:

.

1. Usuário solicita opção de lançamento de período aquisitivo de férias para funcionário.
2. Sistema exibe períodos aquisitivos já lançados, permitindo sua edição e/ou lançamento de novo período.
3. Usuário edita períodos e/ou inclui novo período aquisitivo (neste caso, informando início e fim do novo período).
4. Sistema verifica validade dos dados e, se são válidos, grava as alterações/inclusão. Caso contrário, solicita correção e repete a verificação.

Fluxo Alternativo: abertura de período aquisitivo para todos os funcionários

4. Usuário solicita opção de lançamento de período aquisitivo de férias para todos os funcionários.
5. Sistema requisita informação de início e fim do novo período aquisitivo.
6. Usuário informa período aquisitivo.
7. Sistema promove verificações, gravando o novo período aquisitivo para funcionários que de fato façam jus ao mesmo e para os quais o referido período ainda não tenha sido lançado.
8. Usuário recebe a confirmação de lançamento e encerra o caso de uso.

Regras do Negócio:

RN01 – O funcionário técnico-administrativo faz jus a 30 (trinta) dias de férias para cada período aquisitivo (12 meses de efetivo exercício).

RN02 – O funcionário docente (professor) faz jus a 45 (quarenta e cinco) dias de férias para cada período aquisitivo, exceto no caso de professores substitutos e visitantes (contratos temporários).

RN03 - No caso do funcionário estatutário, o período aquisitivo coincide com o ano civil (1º de janeiro a 31 de dezembro), sendo que o resíduo de tempo correspondente ao

período decorrido entre o início de exercício (data de ingresso) do funcionário até o final do ano em que ingressou na Instituição não é considerado para a usufruição de férias. Para o celetista, cada ano de exercício (independentemente do ano civil) corresponde a um período aquisitivo.

RN04 – Para que o funcionário estatutário usufrua as férias relativas ao seu primeiro período aquisitivo, são exigidos 12 (doze) meses de exercício. A partir de então, as férias podem ser usufruídas dentro do próprio período aquisitivo. Os funcionários celetistas só podem usufruir cada período de férias após o vencimento do respectivo período aquisitivo.

RN05 – No caso de funcionário estatutário que passa a ocupar novo cargo e que tenha saído de outro com vacância decorrente de posse em outro cargo inacumulável, não é exigido o cumprimento dos 12 (doze) primeiros meses de exercício para usufruição de férias, desde que esta exigência tenha sido cumprida no cargo anterior.

RN06 – As férias de funcionário estatutário cujo direito de usufruição já foi adquirido podem ser acumuladas até o máximo de 2 (dois) períodos (exceto quando houver conflito com a RN04). O limite para início de usufruição (integral ou última parcela) das férias acumuladas de um exercício para o outro é o dia 31 de dezembro.

RN07 – No caso de funcionário celetista, as férias devem ser usufruídas dentro dos 12 (doze) meses subseqüentes ao vencimento do período aquisitivo, em um só período, podendo ser excepcionalmente parceladas em dois períodos, sendo que um deles não pode ser inferior a 10 (dez) dias. Se as férias forem concedidas após o vencimento do prazo acima previsto, o empregador pagará em dobro a respectiva remuneração.

RN08 – O funcionário estatutário que usufruir de licença ou afastamento fará jus às férias relativas ao exercício em que retornar.

RN09 – O funcionário que não tenha ainda completado os primeiros 12 (doze) meses de exercício para usufruição de férias e que entre em licença por um dos motivos abaixo, deverá, quando do retorno, completar o referido período.

- Licença para tratamento de saúde de pessoa da família;
- Licença para atividade política, a partir do registro da candidatura e até o décimo dia seguinte ao da eleição, somente pelo período de três meses;
- Licença para tratamento da própria saúde que exceder o prazo de 24 (vinte e quatro) meses;
- Licença por motivo de afastamento do cônjuge.

RN10 – O funcionário celetista tem direito à usufruição de férias na seguinte proporção:

- 30 dias corridos, quando tiver no máximo 5 (cinco) faltas durante o período aquisitivo;
- 24 dias corridos, quando tiver de 6 a 14 faltas durante o período aquisitivo;
- 18 dias corridos, quando tiver de 15 a 23 faltas durante o período aquisitivo;
- 12 dias corridos, quanto tiver de 24 a 32 faltas durante o período aquisitivo.

Para este efeito, não serão considerados como faltas os dias de suspensão preventiva para responder a inquérito administrativo ou de prisão preventiva, em ambos os casos desde que o funcionário tenha sido impronunciado ou absolvido.

RN11 – O funcionário celetista que permanecer em licença por 30 (trinta) dias durante o período aquisitivo, com percepção de salário, não fará jus a férias.

RN12 – No âmbito da UFJF, o professor efetivo celetista faz jus às mesmas regras de férias do funcionário estatutário.

Notas de Implementação:

- A abertura de período aquisitivo consiste no registro das datas início e fim, do total de dias de direito à usufruição de férias, da data prevista para prescrição do direito e, apenas no caso de funcionário celetista, do número de faltas ocorrido durante o período.

- Ao ser solicitado pelo usuário a abertura de período aquisitivo (para um ou todos os funcionários), o sistema deverá verificar:
  - A data de início de exercício (campo DATAEXERCICIO na tabela de vínculos).
  - Se o funcionário é professor ou técnico-administrativo (professor: grupo de cargos = '060').
  - Se o funcionário opera Raios-X (registro de adicional de Raios-X na tabela de adicionais com data fim em aberto).
  - Se o funcionário é celetista ou estatutário (celetista: regime jurídico = 'CLT' ou 'CDT'; estatutário: regime jurídico = 'EST').
  - Caso o funcionário seja celetista, o número de faltas ocorridas durante o período aquisitivo (checagem na tabela de licenças/afastamentos, ocorrências '043', '044', '995', '996').

- Da contagem dos primeiros 12 meses de exercício de funcionário estatutário (RN04), deverão ser descontados os dias de usufruição das licenças cujos códigos de ocorrência são:
  - '013', '019', '031', '113', '117' e '999', descontando apenas o que exceder a 24 meses;
  - '012', '016', '100', '101', '133', '135', '137';

  (RN09)

- No caso de funcionário estatutário, a abertura de período aquisitivo deverá ser implementada da seguinte forma:

  - Verificar o último período aquisitivo registrado para o funcionário. O período a ser aberto corresponderá ao ano seguinte, desde que se trate do ano em curso ou do próximo ano, em relação ao lançamento. Caso ainda não haja período(s) aquisitivo(s) registrado(s), o primeiro período corresponderá ao ano em que o mesmo completar 12 (doze) meses de exercício (RN03; RN04).
  - A data início será sempre o dia 1º de janeiro e a data fim o dia 31 de dezembro (RN03).
  - A data de prescrição será o dia 1º de janeiro do segundo ano posterior ao

período aquisitivo. Ex.: período aquisitivo = 01/01 a 31/12/2004; data de prescrição = 01/01/2006 (RN06).
  - O total de dias de direito à usufruição de férias será igual a 45 no caso de funcionário professor (grupo de cargos = '060') e 30 nos demais casos (RN01; RN02).
  - A abertura de período aquisitivo de professor efetivo, mesmo que seja celetista (grupo de cargos = '060'; regime juridico = 'CLT'; Situação = '01'), deverá seguir as mesmas regras aqui definidas para o funcionário estatutário.

- Se o funcionário for celetista, a abertura do período aquisitivo seguirá as seguintes orientações:
  - Verificar o último período aquisitivo registrado para o funcionário. O período a ser aberto corresponderá aos 12 meses subseqüentes. Caso ainda não haja período(s) aquisitivo(s) registrado(s), o novo período corresponderá aos primeiros 12 meses de exercício do funcionário. Em qualquer hipótese, o novo período só poderá ser aberto após seu vencimento, tendo em vista a necessidade de apuração das faltas ocorridas (RN03; RN04).
  - A data de prescrição será o dia imediatamente posterior ao último dia dos 12 meses subseqüentes ao vencimento do período aquisitivo. Ex.: período aquisitivo = 18/11/2003 a 17/11/2004; data de prescrição = 18/11/2005.
  - O sistema deverá apurar a registrar, na tabela de períodos aquisitivos, o número de faltas (ocorrências de licença/afastamento: '043', '044', '047', '995', '996') do funcionário no decorrer do período a ser aberto. Com esta informação, o total de dias de direito à usufruição de férias será:
    - 5 faltas -> 30
    - 6 a 14 faltas -> 24
    - 15 a 23 faltas -> 18 (RN10)
    - 24 a 32 -> 12
  - O sistema deverá apurar (na tabela de licenças/afastamentos) se o funcionário esteve em licença especificada através da ocorrência '029', durante o período aquisitivo, por 30 dias ou mais. Neste caso, ele não fará jus às férias e o total de dias de direito relativo ao período aquisitivo será 0 (zero). (RN11).

- O período aquisitivo não deverá ser aberto se funcionário se encontrar em licença ou afastamento na data de solicitação da abertura. Se o funcionário não usufruiu férias em determinado ano e, no mesmo ano, iniciou usufruto de licença/afastamento com término no ano seguinte, o período aquisitivo a ser aberto corresponderá ao ano de retorno da licença/afastamento (RN08). As ocorrências de licença/afastamento abrangidas por esta regra são: '035', '102', '104', '105', '106', '107', '109', '128', '135', '166', '175', '176'.

- Na abertura de período aquisitivo para um único funcionário (fluxo principal), caso o funcionário estatutário não tenha completado os primeiros 12 meses de exercício (RN04), o sistema deverá alertar o usuário desta restrição, porém permitindo forçar a abertura, tendo em vista a RN05, neste caso atribuindo automaticamente ao campo "OBS' da tabela de períodos aquisitivos o texto

"Funcionário com ocorrência de vacância por posse em cargo inacumulável".

- Quando da geração de qualquer período aquisitivo, o sistema deverá atribuir o valor inicial 0 (zero) ao campo DIASUSUFRUIDOS da tabela rh_periodoaquisitivo (este campo será atualizado quando houver marcação de férias relativas ao período aquisitivo gerado).