Utilização do Framework PHP CodeIgniter no Desenvolvimento de um Sistema Integrado de Gestão

Daniel Michelon De Carli[1], Márcio Vinissius Fernandes Furtado[1], Pedro Conrad Júnior[1], Sérgio Antônio Martini Bortolin Júnior[1]

[1]Núcleo de Tecnologia da Informação e Comunicação - NTIC
Universidade Federal do Pampa - UNIPAMPA
Av. Tiarajú, 810 - Alegrete, RS - 97546-550 - Brasil

{danielcarli, marciofurtado, pedrojunior, sergiojunior}@unipampa.edu.br

Resumo: *Este artigo apresenta os aspectos envolvidos na utilização do framework de desenvolvimento PHP CodeIgniter para desenvolvimento de um Sistema Integrado de Gestão de forma modular, no que diz respeito a padrões de projeto e arquitetura de sistemas, visando a flexibilidade, agilidade de desenvolvimento e a uniformidade do processo de desenvolvimento de sistemas institucionais da Universidade Federal do Pampa (UNIPAMPA). Desta forma, mantém-se o foco na qualidade do software desenvolvido dando ênfase aos objetos e entidades envolvidos, tendo em vista a não duplicação de dados institucionais, através do uso de uma base de dados centralizada e da reutilização de código, por meio de um conjunto de bibliotecas comum aos módulos, o que impacta diretamente na produtividade da equipe de desenvolvimento de software, e também beneficia o trabalho dos usuários finais através da manutenção da qualidade do software e da redução do tempo de entrega do produto.*

1. INTRODUÇÃO

Um dos grandes projetos em que a UNIPAMPA vem trabalhando é o desenvolvimento de um sistema de gestão baseado em módulos, integrado à base de dados institucional existente na universidade, com o objetivo de fornecer recursos baseados na web, complementares às funcionalidades do SIE (Sistemas de Informações para o Ensino), sistema de gestão em vigor na universidade no momento atual. Este sistema contempla características semelhantes às de um sistema ERP (*Enterprise Resource Planning*).

Um sistema ERP é um *software* integrado de gestão, que engloba e atua em todas as áreas da organização na qual ele é implantado, fornecendo uma base de dados centralizada para armazenamento e recuperação de informações. Segundo Audy, Andrade e Cidral (2005), "o ERP promete resolver uma grande gama de desafios empresariais através da integração dos processos de negócio em uma única arquitetura integrada de informação, o que exige mudanças ma estrutura da organização, no processo de gerenciamento, na plataforma tecnológica e na capacidade de negócios".

Ou seja, um ERP pode ser considerado um sistema macro formado por vários subsistemas interconectados, os quais são definidos pela divisão de funcionalidades com base em assuntos ou áreas, o que visa proporcionar uma estrutura flexível e minimamente acoplada.

Devido ao seu porte, *softwares* de gestão ERP perduram por vários anos nas organizações, podendo ser consideradas extremamente complexas as etapas do seu ciclo de vida, tanto nas fases de concepção (análise e projeto), construção (codificação e testes) e implantação.

Para a solução desenvolvida na UNIPAMPA, o *framework* de desenvolvimento CodeIgniter foi escolhido levando em conta a análise de diversos critérios definidos pela equipe de desenvolvimento relativos à sua construção e à velocidade de aprendizagem da equipe, bem como a possibilidade de integrar os sistemas desenvolvidos de forma modular.

2. CRITÉRIOS PARA ESCOLHA

A escolha do *framework* que promove a estrutura básica para o desenvolvimento deste trabalho baseou-se na avaliação de seis critérios que variam a nota de 0 à 5. Avaliados pela equipe de analistas do NTIC, baseando-se nas experiências profissionais prévias. Os critério são instalação (facilidade e dependência de outras tecnologias), produtividade, performance, curva de aprendizagem, segurança e perspectiva de continuidade. Os principais *frameworks* PHP foram avaliados, podendo ser observadas as avaliações na tabela 1.

Tabela 1 - Comparativo entre os *Frameworks*.

	Instalação	Produtividade	Performance	Curva de Aprendizagem	Segurança	Continuidade	Total
CackePHP	5	5	3	5	4	4	26
CodeIgniter	5	5	5	5	5	4	29
Symfony	5	5	3	3	5	4	25
Zend Framework	4	4	4	4	5	5	26

De maneira geral todos os *frameworks* tiveram notas similares. Contudo, o codeingniter se destacou no somatório dos quesitos. Além disso, foram analisados os seguintes: suporte a bancos de dados, facilidade para geração de funções CRUD (Create, Retrieve, Update & Delete), suporte a PHP5, estrutura orientada a objetos, presença de componentes reusáveis e documentação oficial. Dessa forma, optou-se pelo uso do Codeigniter como base para o desenvolvimento do *framework* utilizado para as aplicações desenvolvidas no NTIC da UNIPAMPA.

Ainda podemos destacar como ponto fundamental a facilidade de aprendizagem e a estrutura bem definida do *framework*, além de sua frequência de atualização e presença de uma comunidade de desenvolvimento ativa. Além desses fatores, o CodeIgniter apresenta um desempenho muito satisfatório ao comparado com os demais *frameworks* do mundo PHP.

Soma-se a isso, considerar a integração proporcionada pelo *framework* entre seus módulos, através da extensão de MVC (*Models, Views e Controllers*) Hierárquico (HMVC), provida por um componente de terceiros, possibilitando a criação de uma estrutura modular padrão para ser usada como base para todos os módulos que sejam futuramente desenvolvidos. O padrão HMVC e os aspectos estruturais do *framework* serão detalhados nas seções seguintes deste artigo.

3. O FRAMEWORK CODEIGNITER

O CodeIgniter é um *framework* PHP baseado no padrão MVC, que contém um conjunto de bibliotecas reutilizáveis, comuns a qualquer sistema *web*, o que elimina a necessidade de reescrevê-las “do zero”. O *framework* preza por fatores como facilidade de uso, ao ponto que poucas horas são necessárias para um desenvolvedor PHP entender sua estrutura e criar um sistema simples, devido a sua baixa curva de aprendizagem.

O CodeIgniter é mantido pela equipe *EllisLab*, a qual libera suas versões oficiais, e é distribuído sob a licença *Open Software License* 3.0 (OSL 3.0), sendo gratuito para *download* e modificação. Sua versão inicial foi liberada no ano de 2006, tendo alcançado estabilidade ao longo dos anos, consagrando-se como um dos *frameworks* PHP mais utilizados no mundo.

O *framework* mescla o paradigma orietando a objetos, através de suas *libraries*, com a programação estruturada, através de seus *helpers*. As *libraries* são classes PHP que atendem as necessidades comuns das aplicações web (como upload, envio e-mail, paginação, controle de sessão, validação de formulários, entre outros). Os *helpers* são funções procedurais utilizadas principalmente para formatação de dados para a camada de apresentação, e funcionalidades menos complexas do que as *libraries*. As *libraries* e os *helpers* do CodeIgniter são recursos genéricos à qualquer aplicação, no entanto, o *framework* também fornece a possibilidade de desenvolvê-los para aplicações específicas, estendendo, desta forma, o *core* (núcleo) do sistema.

Além disso, o *core* do CodeIgniter, provê drivers para integração com a maioria dos SGBD's (Sistemas Gerenciadores de Bancos de Dados) do mercado, não necessitando a escrita manual de comandos SQL, para quaisquer operações DML (*Data Manipulation Language*), tais como *INSERT, UPDATE, DELETE* e *SELECT*.

Do ponto de vista técnico, os objetivos da arquitetura do CodeIgniter são: a instanciação dinâmica (componentes são carregados somente quando requisitados), baixo acoplamento (componentes fracamente relacionados) e singularidade de componentes (classes totalmente autônomas), permitindo o máximo de reutilização.

Outro aspecto vantajoso deste *framework* diz respeito às suas atualizações. Além de serem feitas atualizações com frequência, para atualizar qualquer sistema simplesmente se pode substituir o diretório *system* da aplicação pelo diretório atualizado da nova versão, não sendo necessário modificar nada na aplicação em si, uma vez que seu código fica separado do núcleo da ferramenta.

4. ERP UNIPAMPA

Durante a construção do ERP UNIPAMPA, procurou-se levar em conta a necessidade de que o sistema tivesse uma construção sólida no que diz respeito à segurança e ao controle de processos, bem como a possibilidade de que sejam acrescidos módulos à estrutura principal. Por este motivo, foram adotadas duas características marcantes na arquitetura do sistema: a adoção de uma estrutura modular e um suporte a plugins (regras) de integração.

4.1 Estrutura Modular

Diversas tecnologias existem com o intuito de criar uma arquitetura modular e expansível em plugins ou de maneira semelhante. Dentre elas podemos destacar a tecnologia *Java Portlets* (KLAENE, 2012), em sua especificação JSR 168, e a tecnologia Microsoft do Webparts (MICROSOFT, 2012). Contudo, tecnologias padronizadas para a construção de aplicações em PHP não foram encontradas. O que permitiu o desenvolvimento desse projeto dentro do contexto de arquitetura de sistemas ERP. Assim, para a arquitetura modular, adotou-se uma solução baseada no padrão HMVC, uma variação do padrão MVC, conforme consta na figura 1:

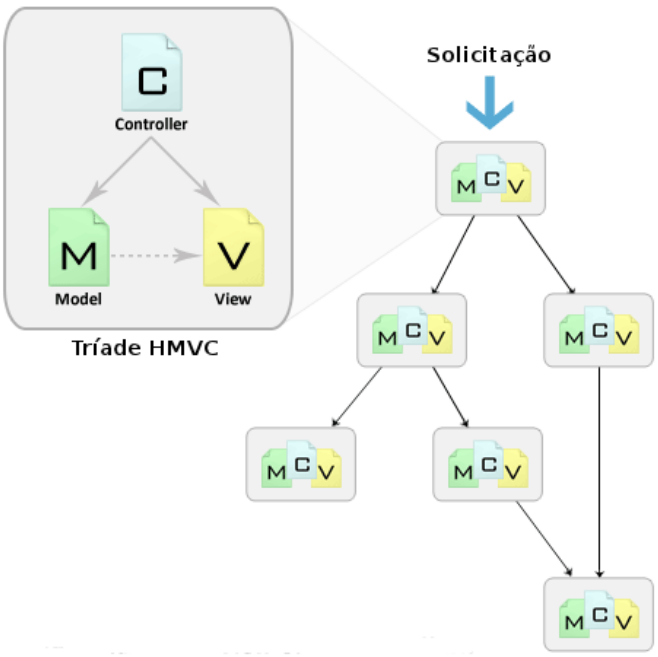

Figura 1 - Esquema HMVC (adaptado de COGAN, Barry)

A aplicação principal possui uma estrutura capaz de receber outros módulos de forma hierárquica, possibilitando separar cada módulo em seu próprio diretório, cada qual destes módulos contendo seus próprios Models, Views e Controllers. Esta estrutura é totalmente transparente ao usuário final e facilita o trabalho dos times de desenvolvimento, permitindo que estas trabalhem em cada módulo separadamente e o integrem em uma estrutura principal, facilitando a utilização de uma base de dados centralizada e a reutilização de código comum aos módulos, tal como fazem os sistemas de ERP existentes atualmente.

4.2 Estrutura de Diretórios

A estrutura de diretórios do sistema é de simples entendimento, conforme mostrado na Figura 2:

erp - diretório raiz do sistema
- **uploads** - *diretório de uploads de qualquer módulo*
- **public** - *arquivos públicos (javascript, css e imagens) da aplicação*
- **system** - *arquivos do CodeIgniter*
- **application** - *arquivos da aplicação*
  - **config** - *parâmetros gerais do sistema (db, e-mail, upload, etc)*
  - **libraries** - *classes genéricas utilizadas por qualquer módulo*
  - **helpers** - *funções procedurais utilizadas por qualquer módulo*
  - **views** - *views genéricas utilizadas por qualquer módulo*
  - **modules** - *pasta que abriga os módulos, identificados pela sigla*
    - **abc** - *contém o módulo "abc" em MVC*
      - **models** - *classes de interação com o db (negócios)*
      - **controllers** - *classes* de *requisições do usuário (negócios)*
      - **views** - *código html (apresentação)*
        - **composite** - *views compostas reutilizáveis*
        - **public** - *arquivos públicos específicos do módulo*
        - **relatórios** - *relatórios do módulo em html*
      - **helpers** - *funções utilitárias específicas do módulo*

Figura 2 - Estrutura de diretórios

A organização de diretórios neste esquema permite uma flexibilidade quanto à adição/remoção de módulos à aplicação, assim como a reutilização de código, sendo que cada módulo usufruirá da mesma instalação do CodeIgniter e da mesma "application", aproveitando os recursos genéricos.

Na interface um recurso do módulo é acessado da seguinte forma: *http://dominio.com/erp/modulo/controller/metodo/*. Utilizando este padrão, teríamos, por exemplo: *http://dominio.com/erp/abc/materiais/novo/*

4.3 Criação de novo módulo

A criação de novos módulos para o sistema ocorre através de um esqueleto de módulo de exemplo (módulo *sample*) contendo funções básicas de CRUD, bem como duas interfaces padrão para listagem de registros e cadastro/edição de registros em uma tabela de dados simples. Assim, a maior parte do tempo de desenvolvimento é dedicada a escrita de código que envolva a lógica de negócio propriamente dita necessária a este novo módulo.

4.4 Suporte a Plugins

No que tange a parte técnica, um ERP deve possuir certa flexibilidade para alocar novos requisitos no seu código-fonte núcleo sem grande esforço de codificação. Por outro lado, também importante, é poder expandir o *software* sem a necessidade de alterar o código-fonte do núcleo do sistema. Uma possível abordagem é a de construir o ERP com suporte a plugins. Plugins normalmente apresentam um baixo acoplamento, o que os torna adequados para construção de programas de forma paralela, por diferentes equipes. Ou seja, a maior parte dos profissionais de programação não necessitarão um conhecimento aprofundado sobre toda a arquitetura do *software*.

Neste sentido, o sistema possibilita o cadastro de plugins via interface, onde é informada a classe responsável por executar a ação. Tais classes, localizadas num repositório de plugins, podem ser disparadas em alguns eventos do sistema, sem afetar o comportamento de tal evento.

4.5 Módulos desenvolvidos

4.5.1 Núcleo do sistema principal

O núcleo do sistema principal é essencial para o funcionamento do sistema, uma vez que os módulos do núcleo fazem a verificação de segurança e disponibilidade dos demais módulos. São eles:

a) Módulo Portal (PTL): É o módulo integrador dos demais módulos, provendo o acesso a todos os módulos através de uma interface padronizada e um menu de opções gerado dinamicamente. Possui recursos de administração do sistema na própria interface, como instalação, desinstalação e exportação de módulos (Figura 3).

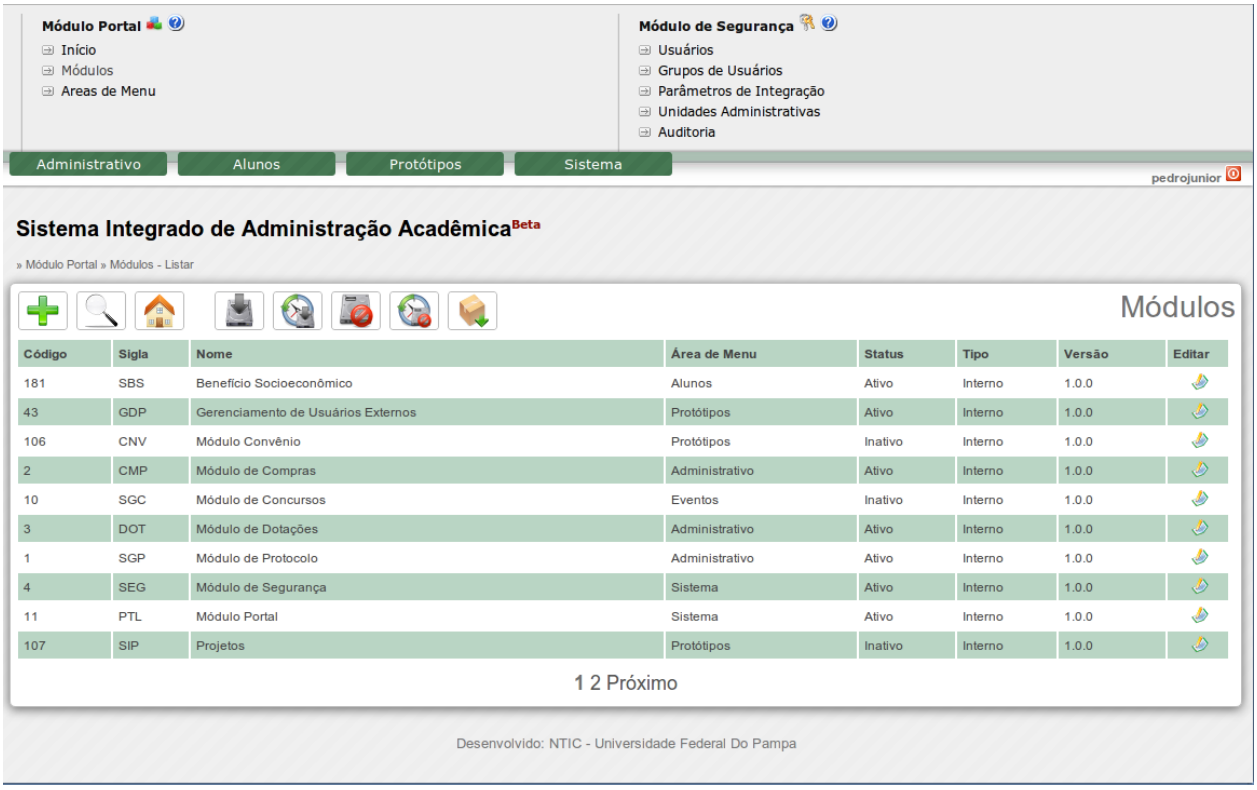

Figura 3 – Tela do módulo integrado de módulos

b) Módulo Segurança (SEG): Compreende o controle de permissões por grupo e a auditoria do ERP.

4.5.2 Módulos secundários

Os módulos secundários são tratados como funções opcionais do sistema, podendo ou não estar todos habilitados. Dentre os módulos já desenvolvidos podemos citar os seguintes:

a) Módulo Compras (CMP): Compreende o módulo responsável por controlar o recebimento e andamento dos processos de compras, materiais e empenho da Universidade de forma centralizada.

b) Módulo Dotações (DOT): É o módulo responsável pelos lançamentos de dotações orçamentárias (receita e despesa) da Universidade. Foi desenvolvido para funcionar de forma integrada com o Módulo Compras.

c) Módulo Protocolo (SGP): Corresponde às funções de protocolo geral e controle de fluxo de processos (workflow) da Universidade. Possui uma funcionalidade de criação de fluxos com tramitação por estados utilizando características obtidas através da importação de diagramas BPM (Business Process Management) via arquivo XPDL (*XML Process Definition Language*) e atribuição de tarefas para grupos de pessoas de acordo com o estado atual do processo.

d) Módulo Solicitação de Benefícios (SBS): É o primeiro módulo de visibilidade pública do sistema, tendo sido construído para receber formulários eletrônicos de solicitação de benefícios socioeconômicos para alunos da Universidade (como bolsas de carência), possibilitando agilizar e padronizar o controle de dados para solicitação de benefícios.

e) Módulo do Sistema de Controle de Usuários Externos (SUE): Ainda em testes, este módulo possibilita o cadastramento e controle de usuários externos à Universidade, possibilitando que estes usuários utilizem a infraestrutura de rede e serviços de internet e plataformas de aprendizagem de

forma temporária, através do fornecimento e controle de validade de credenciais de acesso especiais.

5. CONSIDERAÇÕES FINAIS

Este artigo apresentou o sistema integrado de gestão desenvolvido pela UNIPAMPA. Tal solução foi concebida baseando-se nas necessidades da instituição, uma universidade em construção, carente de funcionalidades de software, as quais o sistema de gestão em vigor na universidade (SIE) não contempla, aliado às necessidades dos desenvolvedores, de uma estrutura organizada e padronizada para atender a demanda, baseada no conceito de reutilização de código e não redundância de dados, utlizando-se de uma base de dados centralizada, integrada ao banco de dados do SIE.

Até o presente momento a estrutura do sistema demonstrou-se viável e consolidada para atender as demandas solicitadas pelos diferentes setores da instituição, ao passo que cada novo sistema pode ser abrigado como um módulo no ERP, aproveitando-se de sua estrutura.

A facilidade de manutenção e a garantia de integridade da estrutura do sistema são parte de um grande desafio aos desenvolvedores, tendo em vista manter a qualidade do que é produzido, ao mesmo tempo que o sistema receberá reformulações, conforme as necessidades da instituição, fator natural no processo de desenvolvimento de software.

Através da utilização deste modelo, tornou-se possível dedicar ainda mais tempo para a efetiva resolução de questões específicas no dia-a-dia da instituição de forma mais clara e objetiva, sem necessidade de reescrever código de recursos já presentes na estrutura. Espera-se ainda que este trabalho reforce a importância da qualidade do desenvolvimento do software utilizando padrões ágeis de desenvolvimento, bem como atender de forma satisfatória as demandas futuras.

6. REFERÊNCIAS

COGAN, Barry. HMVC: an Introduction and Application. Disponível em: http://net.tutsplus.com/tutorials/php/hvmc-an-introduction-and-application/. Acesso em 05/03/2012.
ELLISLAB INC. CodeIgniter User Guide. Disponível em: http://codeigniter.com/user_guide/. Acesso em 05/03/2012.

KLAENE, Michael. Understanding the Java Portlet Specification. Disponível em: http://www.developer.com/java/web/article.php/3366111. Acesso em 06/03/2012.

MICROSOFT CORP. A developer's Introduction to Web Parts. Disponível em: http://msdn.microsoft.com/en-us/library/dd583154(v=office.11).aspx. Acesso em 06/03/2012.