UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA
ESCOLA SUPERIOR DE TECNOLOGIA
ENGENHARIA DE COMPUTAÇÃO

JANDERSON DE MELO ANTUNES

INTEGRANDO OS FRAMEWORKS CAKEPHP E FLEX PARA APLICAÇÕES WEB

Manaus

2010

JANDERSON DE MELO ANTUNES

INTEGRANDO OS FRAMEWORKS CAKEPHP E FLEX PARA APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Computação.

Orientador: Prof. M. Sc. Jucimar Maia da Silva Júnior

Manaus

2010

Universidade do Estado do Amazonas - UEA
Escola Superior de Tecnologia - EST

Reitor:
José Aldemir de Oliveira
Vice-Reitor:
Marly Guimarães Fernandes Costa
Diretor da Escola Superior de Tecnologia:
Mário Augusto Bessa de Figueirêdo
Coordenador do Curso de Engenharia de Computação:
Danielle Gordiano Valente
Banca Avaliadora composta por: *Data da Defesa: 07/12/2010.*
Prof. M.Sc. Jucimar Maia da Silva Júnior *(Orientador)*
Prof. M.Sc. Flávio José Coelho
Prof. M.Sc. Ricardo Barboza

CIP - Catalogação na Publicação

A636i Antunes

Integrando os Frameworks CakePHP e Flex para Aplicações Web/ Janderson de Melo Antunes; [orientado por] Prof. MSc. Jucimar Maia da Silva Júnior - Manaus: UEA, 2010.

79 p.: il.; 30cm

Inclui Bibliografia

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação). Universidade do Estado do Amazonas, 2010.

CDU: 004

JANDERSON DE MELO ANTUNES

Integrando os Frameworks CakePHP e Flex para Aplicações Web

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Computação.

Aprovado em: 07/12/2010

BANCA EXAMINADORA

Prof. Jucimar Maia da Silva Júnior, Mestre
UNIVERSIDADE DO ESTADO DO AMAZONAS

Prof. Flávio José Coelho, Mestre
UNIVERSIDADE DO ESTADO DO AMAZONAS

Prof. Ricardo Barboza, Mestre
UNIVERSIDADE DO ESTADO DO AMAZONAS

Aos meus pais, minha filha e minha esposa por toda dedicação, carinho e renúncias.

Agradecimentos

Agradeço primeiro a Deus. Aos meus pais, minha filha e a minha esposa pelo apoio e carinho durante minha caminhada na graduação. Ao professor Jucimar Jr., pelas idéias e incentivos assim como pela sua orientação em meu trabalho. A todos os meus amigos da UEA pelo companheirismo. Um agradecimento também ao Lenier pelo template em Latex da monografia.

Agradeço também aos amigos de trabalho, especialmente Rodrigo Diniz, Juci Frank e Orange Marques pelos incentivos e apoio que me deram.

Resumo

O propósito deste trabalho é demonstrar através de um Estudo de Caso (uma aplicação real) o uso e a integração de duas poderosas tecnologias altamente conhecidas pela maioria dos programadores orientados para aplicações Web, CakePHP e Flex. Essas tecnologias são conhecidas como frameworks. Sendo que o primeiro, CakePHP, aplica o padrão arquitetural MVC (Model, View e Controller) e o segundo, Flex, aplica a tecnologia RIA (Rich Internet Application). O estudo de caso implementado neste trabalho está baseado em um problema real e conhecido pela EST (Escola Superior de Tecnologia).

PALAVRA-CHAVE: Frameworks.Padrão de projetos.Internacionalização e Localização.Aplicações orientada Web.Sistemas de Cursos Livres.

Abstract

The purpose of this study is to demonstrate through a case study (a real application) the use and integration of two highly powerful technologies known by most programmers oriented Web applications, CakePHP and Flex. These technologies are known as frameworks. Since the first, CakePHP, applies the architectural pattern MVC (Model, View, Controller) and the second, Flex applies the technology RIA (Rich Internet Application). The case study implemented in this work is based on a real problem and known by the EST (Escola Superior de Tecnologia).

KEYWORD: Frameworks.Desing Pattern.Internacionalization and Localization.Web Aplications. Free Courses System.

Sumário

Lista de Tabelas **xi**

Lista de Figuras **xii**

1 Introdução **1**

1.1 Descrição do Problema 2

1.2 Objetivos 2

1.2.1 Objetivo Geral 2

1.2.2 Objetivos Específicos 3

1.3 Justificativa 3

1.4 Metodologia 4

1.5 Organização do Trabalho 4

2 Fundamentação Teórica **6**

2.1 Padrões de Projeto e o MVC 6

2.2 O MVC para Aplicações *WEB* 8

2.3 *Framework* CakePHP 10

2.3.1 Instalação e Configuração do CakePHP 12

2.3.2 Convenções do CakePHP 15

2.3.3 Configuração do Controle de Acesso e Autenticação 17

2.3.4 Configurações para a língua portuguesa 17
2.3.5 I18N e L10N 19
2.4 *Framework* Flex 21
2.4.1 Flex e Tecnologia RIA (*Rich Internet Applications*) 21
2.4.2 Linguagem ActionScript 3.0 22
2.4.3 *Action Message Format* - AMF 22
2.5 FDD (*Feature Driven Development*) 23
2.6 Modelagem em Cores 25

3 Integração dos *Frameworks* CakePHP e Flex 28
3.1 Instalação e configuração do *plugin* Cpamf 30
3.1.1 Parâmetro de Codificação de Caracteres 31
3.1.2 Mapeamento de Objetos 34
3.1.3 Configuração dos Serviços Remotos no Flex 35
3.1.4 Classes de serviços em AS3 36

4 Estudo de caso: Sistema de Controle Acadêmico 38
4.1 Modelagem dos Objetos do domínio do problema 38
4.2 Casos de Uso 40
4.3 Lista de Funcionalidades 40
4.4 Modelo Entidade-Relacionamento 42
4.5 Arquitetura da Aplicação 43
4.6 Gerenciamento de Usuários e Grupos 45
4.7 Gerenciamento de Cursos 47
4.8 Gerenciamento de Turmas 49
4.9 Gerenciamento de Alunos 52
4.10 Gerenciamento de Turmas Ministradas 53

4.11 Relatórios 57
4.12 Configuração do CakePHP para o SCL 57
4.12.1 Geração do CRUD básico do SCL 57
4.12.2 Melhorando a Interface com o Usuário 59

5 Conclusão e Trabalhos Futuros **61**

Referências Bibliográficas **63**

Lista de Tabelas

2.1 Exemplo de reescrita de URL. 12

2.2 Estrutura de diretórios do CakePHP. 13

2.3 Estrutura de diretórios da pasta `app`. 13

Lista de Figuras

2.1 Modelo e suas visualizações alternativas. 8

2.2 Processamento de requisições CGI. 9

2.3 O padrão de projeto Front Controller. 10

2.4 Como o CakePHP implementa o MVC. 11

2.5 Uma visão geral de FDD. 24

2.6 DNC - *Domain Neutral Componente*. 27

3.1 Funcionamento esquemático da comunicação Flex-PHP. 29

3.2 *Gateway* do Cpamf. 31

3.3 Configuração do *Service Browser*. 32

3.4 Janela para explorar Controles e métodos 33

3.5 Hierarquia Servico/UsuarioServico. 37

4.1 Modelo Gerenciamento de Turmas. 39

4.2 Caso de Uso Gerenciamento de Usuários. 40

4.3 Caso de Uso Gerenciamento de Turmas Ministradas. 41

4.4 Caso de Uso Gerenciamento de Turmas e Matrículas. 41

4.5 Detalhamento do Gerenciamento de Turmas. 42

4.6 Detalhamento do Gerenciamento de Turmas Ministradas. 42

4.7 Detalhamento do Gerenciamento de Usuários e Grupos. 42

4.8 Modelo Entidade-Relacionamento 43
4.9 Arquitetura geral do SCl. 44
4.10 Tela Adicionar Usuário. 45
4.11 Tela Alterar Usuário. 46
4.12 Detalhe da Tela Listar Usuário. 47
4.13 Tela Adicionar Grupo. 47
4.14 Tela Alterar Grupo. 48
4.15 Tela Listar Grupo. 48
4.16 Tela Adicionar Curso. 49
4.17 Tela Alterar curso. 49
4.18 Tela Listar curso. 50
4.19 Tela Adicionar Turma. 51
4.20 Tela Alterar Turma. 52
4.21 Tela Listar Turmas. 52
4.22 Tela Adicionar Alunos. 53
4.23 Tela Editar Alunos. 54
4.24 Tela Listar Alunos. 54
4.25 Tela Área Professor. 55
4.26 Tela Pesquisa Turma. 55
4.27 Tela de Resultados de Pesquisa de Turma. 56
4.28 Tela Gerenciar Turma. 56
4.29 Relatório de Notas da Turma. 57
4.30 Menu de opções ao executar o *bake* com a opção *all*. 58
4.31 Tela Alterar Turma sem Melhorias 60
4.32 Tela Alterar Turma com Melhorias 60

Lista de Códigos

2.1 Configuração do Apache. 14
2.2 Configuração de banco de dados. 14
2.3 *Security salt* padrão do*CakePHP*. 15
2.4 Os componentes Auth e ACL adicionados a aplicação. 17
2.5 Configuração para formação de singular e plural. 18
2.6 Configuração de um campo descritivo para a classe Curso. 19
2.7 Trecho de código não internacionalizado. 20
2.8 Exemplo do conteúdo gerado pelo i18n 20
2.9 Trecho corrigido dos arquivos de tradução 20
2.10 Configuração de linguagem padrão. 21
3.1 Configuração do arquivo *routes.php*. 30
3.2 Parâmetro de codificação de caracteres. 32
3.3 Parâmetro de codificação de caracteres. 33
3.4 Componentes disponibilizados para uso em um Controle. 35
3.5 Trecho do arquivo *services-config.xml* 36
4.1 Campo preferencial para exibição na camada de visualização 59

Capítulo 1

Introdução

Diversas tecnologias têm surgido com o intuito de facilitar e agilizar a programação de sistemas de médio e grande porte. Tais tecnologias têm suas peculiaridades, que podem se tornar um problema para os programadores menos experientes.

Este trabalho visa demonstrar através de uma aplicação real o uso e integração de duas poderosas tecnologias que facilitam e agilizam a construção desse sistema, CakePHP e Flex.

O sistema o qual iremos construir para demonstrar o uso e a integração das tecnologias MVC e RIA é baseado em um problema real e conhecido pela EST (Escola Superior de Tecnologia). A EST oferece diversos cursos de extensão para a comunidade. Estes cursos são conhecidos como cursos livres e são voltados para o aprendizado e aperfeiçoamento de conhecimentos técnicos e capacitação profissional.

Os cursos livres também atendem as empresas de Manaus que tenham demanda em treinar e capacitar seus colaboradores. Esses são oferecidos pela EST, os quais consistem de módulos de curta duração.

1.1 Descrição do Problema

Atualmente o controle acadêmico dos cursos livres da EST é realizado "manualmente", ou seja, o esforço humano é muito alto, e ainda demanda muito tempo para organizar e manipular as informações pertencentes aos cursos livres, mesmo utilizando ferramentas de auxílio, tais como *Excel* e *Word*.

Um problema crítico ao utilizar essas ferramentas de auxílio é a inconsistência de dados devido a existência de redundância de dados. Isto ocorre quando temos a informação em dois ou mais documentos diferentes. Por exemplo, um aluno matriculado em mais de um curso livre poderá ter mais de um cadastro. Essas informações naturalmente serão duplicadas e poderão ser inconsistentes no momento em que se fizer necessária a atualização de dados, pois os responsáveis pela organização dos cursos livres podem não se atentarem de atualizar os demais cadastrados existentes. Também, há problemas com relação as planilhas de notas e de frequências de alunos matriculados nos cursos livres, as quais as informações nelas contidas se tornam vulneráveis, pois não há a devida segurança. Por exemplo, não há regras de restrições de dados, podendo todos terem acesso.

Através da construção desse sistema será gerado uma documentação específica sobre as principais dificuldades ao se utilizar tais *frameworks* e como contorná-las. Dessa forma, profissionais interessados em usar esse tipo de tecnologia teriam um documento-guia que os auxiliaria na construção de sistemas semelhantes.

1.2 Objetivos

1.2.1 Objetivo Geral

Integrar duas tecnologias modernas e documentar as dificuldades encontradas e as possíveis soluções dadas a tais dificuldades. Especificamente, para este usaremos as tecnologias *framework* MVC CakePHP e *framework* RIA Flex. Através dessa integração construiremos os fundamentos do sistema de controle acadêmico de cursos livres da EST. Tendo como

resultado a automatização de parte do processo de gestão de cursos livres oferecidos pela EST. A priori, o escopo da criação desse sistema computacional (Sistema de Cursos Livres - SCL) abrangerá a gestão de informações referentes aos cursos livres, alunos e professores.

1.2.2 Objetivos Específicos

- Documentar o processo de adaptação as regras específicas ao *framework* CakePHP para a língua portuguesa. Pois, uma das dificuldades de usar os *frameworks* é o fato de serem otimizados em sua língua nativa, a qual geralmente é a língua inglesa. Sendo oneroso adaptar determinadas regras para língua portuguesa. Mediante tal problema, um dos objetivos específicos é documentar o processo de adaptar as regras específicas ao *framework* CakePHP para a língua portuguesa.

- Realizar a comunicação entre um Sistema de Gerenciamento de Banco de Dados. Neste trabalho usaremos o MySQL.

- Tornar o *framework* CakePHP capaz de entender o protocolo AMF (*Action Message Format*) para poder se comunicar com a aplicação em Flex. Essa comunicação é um ponto importante para a integração dos *frameworks*.

1.3 Justificativa

Mediante os fatos relatados na seção Descrição do Problema e o aumento da demanda por reuso de *software* através de *frameworks* faz-se necessário documentar as principais dificuldades encontradas na adoção dos *frameworks* CakePHP e Flex e suas possíveis soluções.

Visto que o *framework* CakePHP e Flex, assim como a maioria dos *frameworks*, são otimizados na língua inglesa é relevante levantar e apresentar algumas dificuldades de adaptá-los para sistemas de língua portuguesa.

1.4 Metodologia

Para a realização desde trabalho, optamos por usar a abordagem de reuso de *software* e padrões projetos. Para isto, escolhemos dois *frameworks* para integrá-los, ou seja, fazer com elas se comuniquem, tais ferramentas são CakePHP e FLex. Com relação aos padrões de projetos, estes já se encontram incorporados aos *frameworks* escolhidos.

A integração dos *frameworks* comtemplará a comunicação com o Sistema de Gerenciamento de Banco de Dados, MySQL.

E para demonstrar que alcançamos nossos objetivos, construiremos um sistema com escopo pouco abrangente, porém funcional e capaz de demonstrar a integração dos *frameworks*.

1.5 Organização do Trabalho

Este trabalho é composto por cinco capítulos organizados da seguinte forma:

O Capítulo 2 se encontra a fundamentação teórica que é a base conceitual para o bom entendimento deste. Neste, damos a visão geral de *frameworks* e algumas importantes definições tais como: PHP, RIA, Linguagem ActionScript 3.0 e Action Message Format - AMF. Além disso, este abrange a instalação e a configuração do *framework* CakePHP e convenções específicas a esse *framework*.

O Capítulo 3 pode ser considerado o capítulo principal deste trabalho, pois nele se encontra a Integração dos *Frameworks* Cakephp e Flex, foco principal desde trabalho.

O Capítulo 4 apresenta os principais artefatos de modelagem, o qual consideramos a UML (*Unified Modeling Language*), onde nesta são apresentados os diagramas de Classes e o Modelo Lógico Entidade-Relacionamento (projeto de banco de dados). Em seguinda são apresentadas as funcionalidades e telas principais do SCL (Sistema de Cursos Livres). Por fim, na seção 4.12 são apresentadas configurações e melhorias na interface com o usuário usados neste trabalho.

O Capítulo 5 apresenta as conclusões e trabalhos futuros que possam ser realizados com a documentação-guia sobre integração dos *frameworks* CakePHP e Flex.

Capítulo 2

Fundamentação Teórica

Neste capítulo descrevemos as tecnologias aplicadas neste trabalho, tais como: *Frameworks* MVC, onde definimos *Frameworks*, Padrões de Projetos, *Frameworks* MVC e o *Frameworks* MVC aplicado a *web*. Em seguida, descrevemos sobre *Frameworks* de aplicações *web*, os quais são pontos principais desde trabalho, por isso nos concentramos somente nos dois que foram utilizados: CakePHP e Flex.

2.1 Padrões de Projeto e o MVC

Segundo, (SOMMERVILLE, 2010) *Frameworks* e Padrões de Projetos são duas abordagens para o reuso de *software*. Um *framework* (ou *framework* de aplicação) é um projeto de subsistema composto por um conjunto de classes abstratas e concretas e as interfaces entre elas. Conforme o nome sugere é uma estrutura genérica que pode ser ampliada para criar um subsistema ou aplicações mais específicas.

Segundo (GAMMA et al., 1995), o projeto de *software* orientado a objetos é difícil e a construção de *software* reusável, orientado a objetos, é ainda mais. Quando nos deparamos com um problema, geralmente a primeira abordagem é buscar a solução por meio de analogias existentes em problemas já resolvidos. Neste contexto, os Padrões de Projetos

foram criados para catalogar as soluções para problemas recorrentes de desenvolvimento de *software*, de modo que não precisemos reinventar a roda.

O padrão de projeto é uma descrição do problema e a essência da sua solução, dessa forma a solução pode ser reusada em aplicações diferentes. O padrão não é uma especificação detalhada. Pode-se pensar nele como uma descrição de conhecimento e experiência acumulados, uma solução comprovada para um problema comum.

O *Framework* MVC, Modelo-Visualização-Controle (*Model-View-Controller*), foi concebido no final da década de 1970 por Trygve Reenskaug com o intuito de permitir interfaces gráficas capazes de manipular diferentes visualizações (*Views*) de dados. Mais tarde surgiu uma versão modificada do *framework* que foi implementada na linguagem *Smalltalk-80*. Tal versão era usada para construir interfaces de usuário.

O *Framework* MVC tal como implementado no *SmallTalk-80* consiste de três objetos: o Modelo (*Model*), a Visualização (*View*) e o Controle (*Controller*). O Modelo é o objeto da aplicação onde se encontra os dados a serem manipulados ou visualizados. A Visualização corresponde a tela de apresentação do modelo. Os componentes GUI (Graphical User Interface) tais como, botões, campos de textos, etc, usados nas tela da aplicação também fazem parte da Visualização. O Controle define a maneira de como a interface gráfica reage aos dados de entrada do usuário.

A separação entre apresentação e modelo é alcançada pelo o estabelecimento de um protocolo em que um objeto (Visualização) cadastra-se para receber notificações de outro objeto (Modelo). Quando o estado do modelo se altera, o modelo notifica todas as visualizações interessadas. Então, As visualizações têm oportunidades de se atualizarem. Desta forma, um modelo pode ter múltiplas visualizações, tais como ilustrado na Figura 2.1. Essa abordagem é baseada no padrão de projeto Observador (*Observer*).

A Visualização usa uma instância particular do objeto Controle para implementar uma estratégia de resposta aos estímulos dos usuários. Para mudar a forma que uma interface responde as entradas dos usuários basta que se escolha um objeto Controle diferente. Por exemplo, em tempo de execução uma tela poderia ser configurada para um estado inativo através da substituição de um Controle capaz de tratar eventos (entrada de dados) por um

Controle que ignorasse qualquer estímulo vindo do usuário. Essa abordagem é baseada no padrão de projeto Estratégia (*Strategy*).

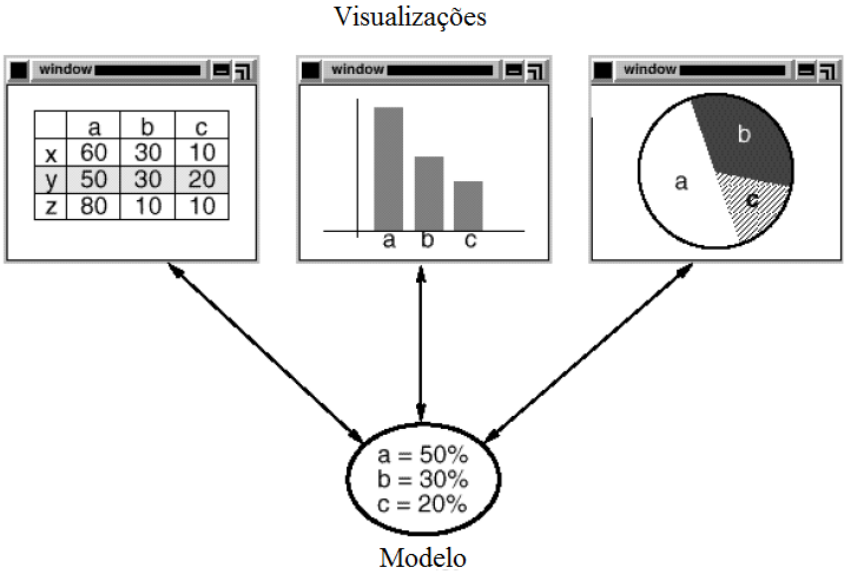

Figura 2.1: Modelo e suas visualizações alternativas.
Fonte: GAMMA et al em Padrões de Projetos, 1995.

2.2 O MVC para Aplicações *WEB*

O MVC, aplicado a Web, surgiu junto com a adoção da Programação Orientada a Objetos (POO) aplicada a *Web*. Antes disso, os sistemas *Web* eram construídos usando CGI (*Common Gateway Interface*). Estes sistemas consistem de códigos interpretados ou compilados e eram instanciados por um servidor *Web*, em um processo separado, para realizar o processamento de requisições HTTP (*Hypertext Transport Protocol*). A Figura 2.2 ilustra o funcionamento básico dos programas CGI.

No ano de 1997, a empresa *Sun Microsystems* publicou a especificação do *Java Servlet 1.0* cujo objetivo era melhorar o desempenho de aplicações CGI por meio da execução do processamento de requisições em *threads* em uma máquina virtual Java. Em 1999, a *Sun Microsystems* lança a especificação 1.0 do *Java Server Pages* (JSP), uma tecnologia similar ao *Active Server Pages*(ASP) da *Microsoft*. O JSP consiste de um arquivo contendo código

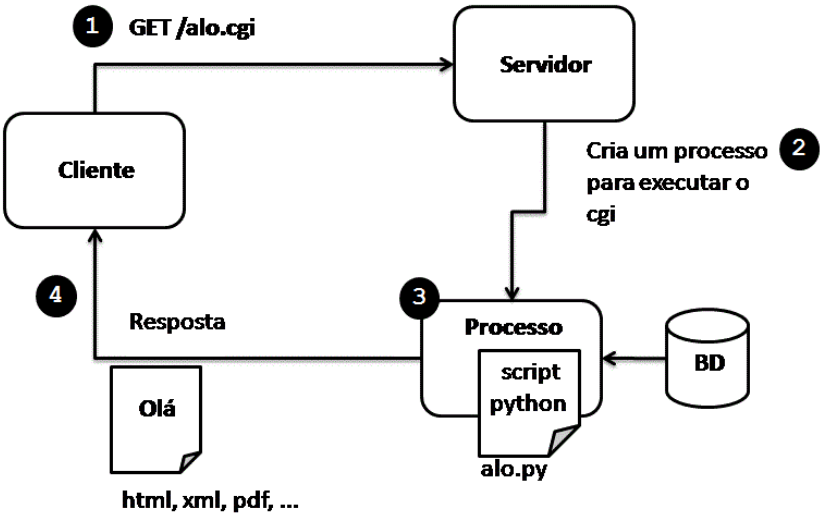

Figura 2.2: Processamento de requisições CGI.

HTML (*Hypertext Markup Language*) e pequenos blocos de código escritos em linguagem Java, os *Scriptlets*. As requisições HTTP são processadas por servlets que interagem com os modelos em Java são chamados de JavaBeans, e depois transferem o controle para uma página JSP a qual é responsável pela apresentação do conteúdo no *browser*. Essa abordagem era referenciada como *Model 2* pela *Sun Microsystems* e foi considerada como sendo a implementação MVC no lado servidor.

Apesar da empresa *Sun Microsystems* ter criado o *Model 2*, a ampla adoção deste padrão arquitetural em aplicações *Web* é devida a Craig R. McClanahan, criador do *framework* Struts. Tal *framework* tornou-se um projeto de código aberto no ano 2000, quando McClanahan doou o código para a *Apache Software Foundation*. O *framework* Struts implementa uma variação do MVC conhecida como padrão *Front Controller*. Este padrão é a estrutura mais comum de aplicações MVC para *Web* . A Figura 2.3 ilustra os componentes do padrão *Front Controller*.

O objeto *Front Controller* é responsável por centralizar o tratamento e processamento de atividades comuns. Por exemplo, segurança, gerenciamento da sessão e processamento de requisições HTTP.

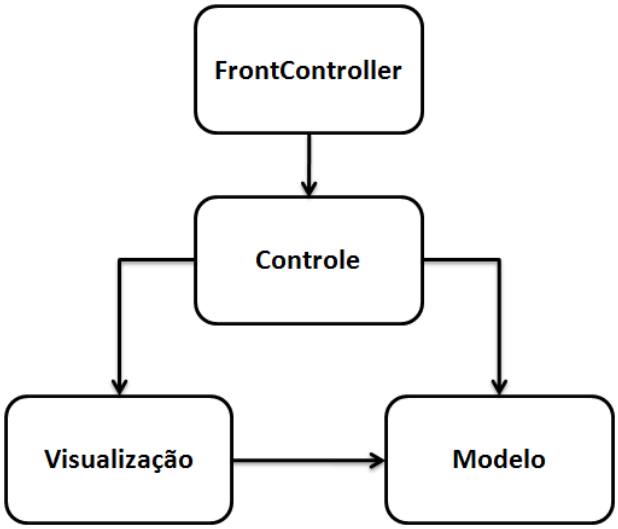

Figura 2.3: O padrão de projeto Front Controller.

As requisições que chegam no *Front Controller* são encaminhadas para os Controles apropriados. Os Controles são os componentes que respondem as entradas do usuário. Os Controles *Web* diferem dos Controles do Smalltalk-80, pois estes recebiam sinais de dispositivos de *hardware* tais como teclados, *mouse*, etc. Entretanto, os Controles *Web* processam apenas as requisições HTTP que o *Front Controller* delega.

A Visualização é o conteúdo de uma página HTML, geralmente. Podem ser implementadas através de *templates* que são entrada para um *View Processor*, um programa que interpreta o *template* e gera HTML. Em algumas implementações o *template* é compilado em um objeto que encapsula o conteúdo a ser apresentado.

O Modelo é o encapsulamento dos dados derivados ou permanentes podendo, também, encapsular algumas regras de negócio.

2.3 *Framework* CakePHP

A linguagem PHP (Processador de Hipertexto) é uma linguagem interpretada e orientada a objetos destinada a programação *web*. Segundo (DOYLE, 2009), os programas PHP (conhecidos como *scripts* PHP) são executados em um servidor por meio de um motor

PHP quando uma URL é requisitada por um usuário.

O *framework* CakePHP é totalmente implementado na linguagem PHP e é um *framework* para simplificar o desenvolvimento de aplicações *web*. Este foi projetado tendo como inspiração a filosofia de desenvolvimento ágil do *Ruby On Rails*, a qual é também um *framework web*, mas escrito na linguagem *Ruby*.

Segundo (GOLDING, 2008), o *framework* CakePHP implementa o MVC, ou seja, separa as operações típicas em três camadas específicas: *Modelos* para interações com o banco de dados, *Visualizações* para apresentação dos dado e *Controle* para os comandos de fluxo do programa.

Para entender como o CakePHP implementa o MVC, veja a Figura 2.4.

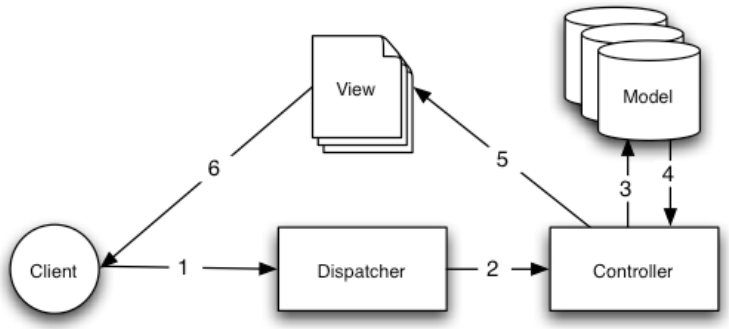

Figura 2.4: Como o CakePHP implementa o MVC.
Fonte: `http://cakephp.org`

No passo 1: Uma requisição HTTP feita pelo *Cliente* é intermediada pelo *Despachante*.

No passo 2: O *Despachador* faz uma análise léxica da requisição HTTP, extraí os parâmetros e os repassa para uma ação(método) no *Controle* apropriado.

No passo 3: O *Controle* executa alguma lógica, por exemplo verifica se o usuário conectado está logado e em seguida usa um *Modelo* para acessar os dados armazenados.

No passo 4: O *Modelo* recupera os dados do banco de dados[1] e os disponibiliza para o

[1]De acordo com (FOUNDATION, 2010b), os *Modelos*, geralmente, são associados com tabelas em banco de dados, mas também podem ser qualquer fonte de dados existentes. Por exemplo, um arquivo XML(*eXtensible Markup Language* - Linguagem de Marcação Extensível) ou um objeto *JSON*.

Controle.

No passo 5: O *Controle* repassa os dados recebidos do *Modelo* para a *Visualização* apropriada.

No passo 6: A *Visualização* apresenta os dados para o *Cliente*.

2.3.1 Instalação e Configuração do CakePHP

O primeiro passo da configuração do ambiente de trabalho é a instalação de um servidor, neste caso usamos o servidor Apache, também conhecido por *Httpd*. O servidor Apache é um *software* de código aberto distribuído pela *Apache Software Foundation*. Atualmente, o Apache é um dos servidores *HTTP* mais usados na internet e está disponível em diversas plataformas: *Unix*, *Windows*, *MAC OS X* e outros.

Neste trabalho foi usado um módulo para mapear URLs (*Universal Resource Locator*) em outra URL mais amigável ao usuário. Este módulo é chamado de *mod_rewrite.c*, o qual é um módulo do servidor Apache que usa expressões regulares[2] para reescrever URLs. Este módulo atua durante a fase de processamento de uma requisição HTTP. Nesta fase, toda ou parte de uma URL é substituída e mapeada em outra URL ou em um endereço de arquivo local. A Tabela 2.1 mostra o mapeamento de uma determinada URL para outra. Note que a URL reescrita é mais compacta, portanto facilita memorizá-la.

url normal	www.uea.edu.br/cursos-livres/index.php?url=cursos/view/1
url reescrita	www.uea.edu.br/cursos-livres/cursos/view/1

Tabela 2.1: Exemplo de reescrita de URL.

O Sistema Operacional (SO) utilizado foi o Windows Seven Starter com o servidor Apache, o interpretador PHP e o Sistema de Gerenciamento de Banco de Dados (SGBD) Mysql instalados e configurados em uma máquina Pentium Dual Core T4400 e 2 Gigabytes de memória RAM.

[2]Uma expressão regular é uma forma concisa e flexível de identificar cadeias de caracteres de interesse, como caracteres particulares, palavras ou padrões de caracteres (WIKIPEDIA, 2010)

Para a obtenção do *framework* CakePHP visite o endereço[3]. A instalação do *framework* CakePHP é bem simples bastando descompactar em seu *Document Root*[4]. Ao descompactar o CakePHP, nos deparamos com a seguinte estrutura de diretórios e arquivos da Tabela 2.2.

app	pasta onde a maior parte do desenvolvimento é realizada
cake	pasta central do *framework*
vendors	biblioteca de terceiros
plugins	pacotes no formato *plugin*
.htaccess	arquivo de configuração do apache
index.php	o *FrontController* do CakePHP
README	instruções de uso do *framework*

Tabela 2.2: Estrutura de diretórios do CakePHP.

Duas pastas do *CakePHP* merecem nossa atenção: `cake` e `app`. A pasta `cake` contém o núcleo do CakePHP. Nela também encontram-se diversos utilitários que auxiliam no desenvolvimento, incluindo um gerador de código. É recomendável não alterar esta pasta para manter compatibilidade com versões futuras do CakePHP.

A pasta `app` é a mais utilizada durante o desenvolvimento. A Tabela 2.3 mostra a estrutura de diretórios da pasta `app`.

config	contém arquivos de configuração
controllers	contém os controles da aplicação
locale	contêm arquivos relacionados a I18N (Ver Seção 2.3.5)
models	contêm os modelos usados na aplicação
plugins	pacotes no formato *plugin*
tmp	pasta para armazenamento temporário
views	arquivos relacionados a camada de visualização
webroot	contêm arquivos CSS[5], imagens e javascript da aplicação

Tabela 2.3: Estrutura de diretórios da pasta `app`.

Em termos de abreviação, o endereço do servidor Apache 2.2 será referenciado por APACHE_HOME. Após a instalação deve-se configurar o servidor Apache para que este

[3]`http://github.com/cakephp/cakephp/archives/1.3.4`

[4]Para o SO Windows o *Document Root* está localizado em `C:/Program\Files/Apache\Software\Foundation/Apache\2.2/htdocs`

reconheça os arquivos de configurações *.htaccess*[6] do CakePHP. Na instalação padrão do servidor Apache o caminho completo para este arquivo de configuração é `APACHE_HOME/conf/httpd.conf`. A configuração deve ser realizada inserindo as linhas conforme a listagem 2.1, no arquivo de configuração do servidor Apache (httpd.conf):

```
<Directory APACHE_HOME/htdocs>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
</Directory>

<Directory APACHE_HOME/htdocs/cursos>
    Allow from All
    AllowOverride All
</Directory>
```

Listagem 2.1: Configuração do Apache.

Configurando o Acesso ao Banco de Dados no CakePHP

As configurações de banco de dados devem ser feitas no arquivo `CAKE_HOME/app/config/database.php`. O CakePHP disponibiliza o arquivo `CAKE_HOME/app/config/database.php.default` para ser usado como modelo. Basta preencher esse arquivo de configuração conforme os dados do banco de dados a ser utilizado.

```
class DATABASE_CONFIG {
    var $default = array(
        'driver' => 'mysql',
        'persistent' => false,
        'host' => 'localhost',
        'login' => 'usuário',
        'password' => 'senha',
        'database' => 'nome do banco de dados',
        'prefix' => '',
```

Listagem 2.2: Configuração de banco de dados.

[6]Um arquivo de configuração do apache que é localizado dentro do diretório de uma aplicação Web. No CakePHP, este arquivo encontra-se no diretório raiz, no diretório `app` e `app/webroot`.

Configurando do Security Salt

O *Security salt* é um código usado internamente para gerar chaves de espalhamento que são usadas durante a criação de sessões de usuários. É uma boa prática mudar esta chave no ambiente de produção. O arquivo encontra-se no caminho `CAKE_HOME/app/config/core.php`.

```
Configure::write('Security.salt',
                 'DYhG93b0qyJfIxfs2guVoUubWwvniR2G0FgaC9mi'
                );
```

Listagem 2.3: *Security salt* padrão do*CakePHP*.

2.3.2 Convenções do CakePHP

O CakePHP é considerado um *framework* de desenvolvimento rápido. Uma das características que contribuem para este fim são as convenções de código. Por meio das convenções o *framework* é capaz de gerar algumas funcionalidades sem que o programador escreva uma linha sequer de código.

Por ser um *framework* em língua inglesa algumas convenções de código acabam favorecendo o idioma em inglês. Como veremos nas seções 2.3.4 e 2.3.5 do capítulo 4, o CakePHP oferece meios para se beneficiar das convenções de código através do mecanismo de *Inflactors*[7] e da *Internacionalização*(I18N e L11N).

Nomes de Tabelas

Os nomes de tabelas devem estar no plural. Por exemplo, uma tabela que guarda as informações relativas aos cursos oferecidos pode se chamar de *cursos*.

[7]*Inflactors* são regras configuradas para por palavras no singular ou plural.

Nomes para Controles

Para nomear as classes *Controles*, usa-se o nome da tabela justaposto com a palavra *Controller*. Por exemplo, considerando a tabela *cursos*, a classe *Controle* seria nomeada como *CursosController*. E, também, o nome do arquivo da classe obedece a uma convenção, onde o nome é formado a partir do nome da tabela justaposto ao texto *_controller.php*. Logo, o nome resultante seria `cursos_controller.php`, sendo o caminho completo deste arquivo `CAKE_HOME/app/controllers/cursos_controller.php`.

Nomes para Modelos

A regra de formação dos nomes das classes *Modelos* é o nome da tabela no singular com a primeira letra capitulada. Mantendo nosso exemplo, teríamos a classe *Cursos*. Esta seria arquivada no diretório `$CAKE/app/models/cursos.php`. Vale observar que estas regras de formação de nomes têm um propósito. Internamente o CakePHP usa essas convenções para extrair informações úteis. Por exemplo, a partir do nome do *Controle* é possível determinar a classe *Modelo* e a tabela do banco de dados ou extrair associações entre tabelas.

Nomes para Visualização

A convenção para a *Visualização* segue uma abordagem diferente. O padrão do CakePHP é apresentar as informações em uma página HTML. O nome da página corresponde a uma ação\método no *Controle*. Considerando o nosso exemplo, estas páginas de visualização são armazenadas no caminho `CAKE_HOME/app/views/cursos`. Considere que *CursosController* define uma ação\método para listar os cursos oferecidos pela UEA digamos, `listar_cursos`. Quando a ação\método `listar_cursos` for chamada, o CakePHP apresentará os dados para o usuário através do arquivo guardado em `CAKE_HOME/app/views/cursos/listar_cursos.ctp`. Note que o nome do método corresponde ao nome do arquivo de visualização sem a extensão.

2.3.3 Configuração do Controle de Acesso e Autenticação

Em (FOUNDATION, 2010a), ensina que o controle de acesso e autenticação é realizado por dois componentes do CakePHP: *Auth* e ACL. Para utilizar estes dois componentes em uma aplicação CakePHP basta adicioná-los ao *array components*, na classe AppController (Veja Listagem).

```
<?php
   class AppController extends Controller {
        var $paginate = array('limit' => 10, 'page' => 1);
        var $components = array('Util', 'Acl', 'Auth', 'Session',
    }
```

Listagem 2.4: Os componentes Auth e ACL adicionados a aplicação.

.

O componente *Auth* é o componente usado para verificar se um usuário possui uma conta no sistema válida. A verificação ocorre após um usuário enviar seu *login* e senha por meio de um formulário de *login*. O componente *Auth* pode ser usado em conjunto com o componente ACL.

O ACL (Access Control List), Lista de Controle de Acesso, define regras mais complexas que são aplicadas quando do acesso de um usuário a um recurso do sistema. Por exemplo, é possível através do ACL permitir que todos os usuários (anônimos ou não) acessem a página principal de um site mas restringir que usuário anônimos acessem um página administrativa.

2.3.4 Configurações para a língua portuguesa

Uma das dificuldades de quem usa um *framework* otimizado para a língua inglesa é adaptá-lo para outra língua, no nosso caso, a língua portuguesa. Embora a tecnologias I18N e L10N já existam a algum tempo é sempre maçante ter que adaptar um *framework* para nossa língua. Como foi comentado brevemente na seção 2.3.2, o CakePHP faz uso de determinadas convenções de codificação para gerar código automático.

A regra de nomeação de tabelas (ver seção 2.3.2) é um exemplo de convenção usada pelo o *framework*. Essa regra, assim como as outras existentes, não existem apenas por convenção. O *bake* [8] ao gerar as classes MVC baseia os nomes das classes a partir do nome das tabelas. Por exemplo, a tabela *professores* dá origem a dois arquivos: ProfessoresController (o controle) e Professor (o modelo). Quanto a visualização, é gerada uma pasta *professores* contendo as telas em HTML de uma aplicação CRUD, a saber: add.ctp, edit.ctp, index.ctp e view.ctp.

O problema surge quando uma regra linguística do inglês não encontra um precedente em nossa língua portuguesa. Veja o caso da tabela *professores*. Se tentarmos gerar as classes MVC a partir desta tabela obteremos os seguintes resultados: a classe modelo *Professore* e o erro informando que a tabela *professore* não foi encontrada. Este problema está relacionado com as regras para formação de plural e singular das palavras.

Existe duas maneiras de resolver este problema, entretanto optaremos pela maneira rápida e eficiente, as regras de *inflectors*. Estas regras consistem de um *array* associativo onde podemos mapear as palavras no singular para sua versão no plural e vice-versa. As regras de *inflactor* usadas para os nomes *professor* e *frequência* estão ilustradas na Listagem 2.5.

```
Inflector::rules('singular',array(
  'rules' => array('/professores$/i' => 'professor',
                   '/frequencias$/i' => 'frequencia'
)));

Inflector::rules('plural',array(
  'rules' => array('/professor$/i'  => 'professores',
                   '/frequencia$/i' => 'frequencias'
)));
```

Listagem 2.5: Configuração para formação de singular e plural.

Um recurso interessante do código gerado pelo CakePHP é a apresentação de certos campos nas telas de visualização com valores mais amigáveis aos seres humanos. Por exemplo, em uma caixa de combinação (*combobox*) contendo a lista de cursos oferecidos por uma instituição, os valores apresentados serão os nomes dos cursos ao invés de IDs. Isso

[8] O *bake* é um utilitário com interface linha de comando.

é possível pois o CakePHP busca por atributos descritivos nos objetos. Quando o CakePHP não encontra um campo descritivo, o padrão é usar a chave primária. Certamente é mais fácil lembrar-se dos nomes dos cursos do que de seus códigos. O problema é que tais campos são buscados em inglês. Isso geralmente frustra os iniciantes, pois se aprende em inúmeros tutorias pela internet que esse comportamento é automático e, no entanto, não funciona como o esperado.

Aos iniciantes no CakePHP recomenda-se recorrer primeiramente ao manual e só após esgotar-se todas as buscas por informações, pesquisar no Google ou qualquer outra máquina de busca. O problema anteriormente relatado tem uma solução bem simples. Basta informar o campo descritivo na classe modelo. Para o nosso exemplo, a classe em questão é Curso. A Listagem 2.6 ilustra como configurar esse campo.

```
<?php
class Curso extends AppModel {
        var $displayField = 'nome';
```

Listagem 2.6: Configuração de um campo descritivo para a classe Curso.

2.3.5 I18N e L10N

Os termos I18N e L10N referem-se, respectivamente a Internacionalização (*Internationalization e Localization*). A internacionalização refere-se a capacidade de uma aplicação ser localizada. Por localizada, queremos dizer que uma aplicação adápta-se a um idioma específico. O abreviação I18N e L10N vem do fato de que existem 18 (dezoito) e 10 (dez) caracteres, respectivamente, entre a primeira e última letras dos termos *Internationalization e Localization*.

O CakePHP oferece a seus usuários meios de internacionalizar as aplicações contruídas com o *framework*. Para realizar isto é necessário passar as *string* que se deseja traduzir para a função de tradução do CakePHP. Na Listagem 2.7 podemos ver duas versões da mesma *string*, a primeira não é fixa e segunda é passada para a função tradutora do CakePHP.

Para a localização funcionar é necessário criar o arquivo do idioma no caminho `CAKE_`

```
<h2>Posts</h2>

<h2><?php __('Posts') ?></h2>
```

Listagem 2.7: Trecho de código não internacionalizado.

`HOME/app/locale/<Códigodeidioma>/LC_MESSAGES/default.po`. No caso do português falado no Brasil precisamos de uma arquivo no seguinte caminho: `CAKE_HOME/app/locale/pt_br/LC_MESSAGES/default.po`. Vale destacar que o CakePHP disponibiliza o utilitário i18n para criar este arquivo automaticamente. Ao invocar o utilitário i18n este varre todas as classes do diretório especificado e procura por chamadas a função tradução do CakePHP e gera a entrada correspondente no arquivo *default.po*. Um trecho do arquivo gerado encontra-se na Listagem 2.8.

```
#: \controllers\alunos_controller.php:23;38
msgid "The aluno has been saved"
msgstr ""
```

Listagem 2.8: Exemplo do conteúdo gerado pelo i18n

Neste trecho, destacam-se dois elementos importantes, *msgid* e *msgstr*. O primeiro corresponde a string que será traduzida, ou seja a *string* que será passada como parâmetro da função de tradução do CakePHP. O *msgstr* guarda a *string* traduzida. Note que as *strings* geradas precisam de alguns ajustes devido ao nosso idioma. A Listagem 2.9 mostra a correção que fizemos e a respectiva tradução.

```
#: \controllers\alunos_controller.php:23;38
msgid "The student has been saved"
msgstr "O aluno foi salvo"
```

Listagem 2.9: Trecho corrigido dos arquivos de tradução

Uma vez tendo criado o arquivo de tradução e ter traduzido as mensagens para o português, basta configurar a linguagem padrão do CakePHP para o português. Isso pode ser feito no arquivo `CAKE_PHP/app/config/bootstrap.php`. A configuração está ilustrada

na Listagem

```
Configure::write('Config.language', 'pt-br');
```

Listagem 2.10: Configuração de linguagem padrão.

2.4 *Framework* Flex

O Adobe Flex[9] é um *framework* de código-aberto para desenvolvimento de aplicações RIA (*Rich Internet Application*) multiplataforma.

O Flex é baseado na plataforma Adobe Flash e faz uso da linguagem ActionScript para o ambiente de programação.

O Flex SDK(*Software Development Kit*) dispõe de uma ampla gama de aplicações *widgets* tais como: botões, campos de entrada de dados, caixas de seleção, formulários, grades (*grids*), gráficos (*charts*), diversos controles de texto e *layouts*.

2.4.1 Flex e Tecnologia RIA (*Rich Internet Applications*)

Segundo (MOORE; BENSON; BUDD, 2007), o termo RIA refere-se a melhoria da interação com o usuário por meio de diversos recursos visuais, tais como: animações, comportamentos de "*arrastar* e *soltar*"e outros recursos que geralmente só encontramos em aplicações Desktop.

Aplicações RIA separam o projeto de uma aplicação *web* em duas partes: lado cliente e lado servidor, onde o primeiro apresenta uma interface com capacidades muito próximas das que encontramos em aplicações Desktop, e a segunda realiza o processamento de dados do lado servidor.

De acordo com (HARVEY, 2008), uma aplicação RIA feita em Flex é executada dentro de um *browser* por meio do *plugin* FlashPlayer. Por segurança, o FlashPlayer implementa

[9] `http://www.adobe.com/devnet/flex,13/08/2010`

uma camada de segurança (*sandbox*) que limita a visualização e o acesso ao sistema de arquivos do lado cliente.

Uma vantagem de usar Flex é ter o desempenho otimizado, pois uma aplicação RIA evita o tempo de latência da rede através do processamento de informações na máquina cliente. Por exemplo: a validação das informações de um formulário no lado cliente pode ser feita no próprio *browser*. Outra vantagem, é o fato de não haver mais a instalação do *software*, apenas o acesso ao *software*, isto propicia a ter sempre a versão atualizada dele. E, por último, o Flex pode ser executado em modo *offline* desde que tenha sido feito pelo menos um acesso a aplicação e esteja na memória *cache* do *browser*.

2.4.2 Linguagem ActionScript 3.0

O ActionScript, também conhecido simplesmente pelo acrônimo AS3[10], é uma linguagem de *script*, baseada na linguagem ECMAScript, utilizada para programação de aplicações RIA.

A AS3 possui muitas semelhanças de sintaxe com o *Javascript*, pois ambas se baseiam no padrão *ECMAScript*. Os *scripts* escritos em AS3 são executados em uma máquina virtual denominada AVM (*ActionScript Virtual Machine*). Esta máquina virtual está presente no plugin *Adobe Flash Player*, disponível para diversos *browser* de internet e plataformas de sistemas operacionais no mercado.

2.4.3 *Action Message Format* - AMF

O AMF é um formato binário compacto que é usado para serializar grafos de objetos ActionScripts. Os dados codificados em AMF podem ser usados para persistir ou recuperar o estado de uma aplicação entre sessões ou permitir que dois pontos terminais se comuniquem através da troca de dados fortemente tipados.

O AMF é uma especificação de código aberto promovido pela *Adobe Systems Inc* e

[10] `http://pt.wikipedia.org/wiki/ActionScript,13/08/2010`

pode ser adquirida através do endereço eletrônico *http://opensource.adobe.com*.

2.5 FDD (*Feature Driven Development*)

O FDD (*Feature Driven Development*), é uma processo de desenvolvimento ágil desenvolvido por Peter Coad e Jeff De Luca (PALMER, 2010). O FDD foi concebido em outubro de 1997 em um projeto para um grupo Bancário Regional do sudeste da Ásia e tem sido usado desde então em diversos projetos ao redor do mundo.

O projeto FDD típico é organizado em cinco fases:

- Desenvolva um modelo abrangente do domínio do problema.
- Construa uma lista de funcionalidades (*Feature List*).
- Planeje por funcionalidade.
- Projete por funcionalidade.
- Construa por funcionalidade.

Apesar da lista anterio dar a entender se tratar de um processo de desenvolvimento de software sequencial, o FDD é um processo de desenvolvimento iterativo como mostra a Figura 2.5.

A primeira fase do FDD é a construção de um modelo do domínio do problema. Este modelo deve ser pensado de maneira abragente de tal forma que capture os conceitos chaves, os relacionamentos e interações entre os objetos do domínio do problema. Ao final desta fase não teremos um modelo completo. Na verdade, esse modelo será incrementado ao longo das fases do projeto. Embora não seja uma obrigação, o modelo é tipicamente construído usando as técnicas de modelagem em cores desenvolvidas por Peter Coad (PALMER, 2010). Na próxima seção discutiremos alguns pontos dessa abordagem.

A próxima fase é a criação de uma lista de funcionalidades (*Features List*). O FDD fornece uma definição formal de uma funcionalidade como sendo uma pequena função de

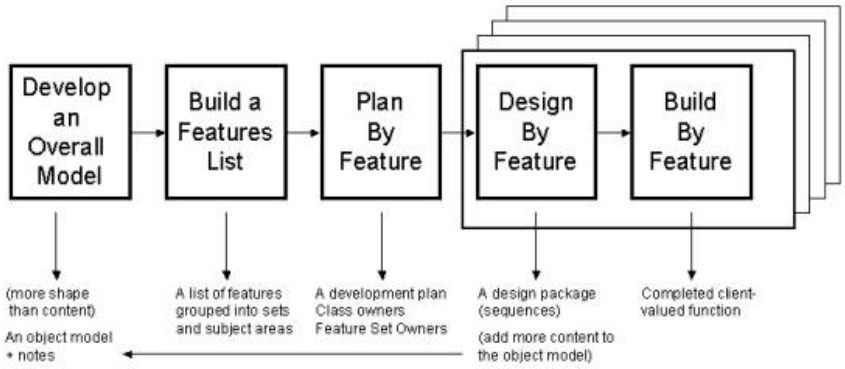

Figura 2.5: Uma visão geral de FDD.
Fonte: `http://www.step-10.com/SoftwareProcess/FeatureDrivenDevelopment/introduction.html`

valor para o cliente e pode ser expressada na forma: <Ação> <Resultado> <Objeto>. Por exemplo, *calcular o total de produtos comprados*. O significado de *pequena* refere-se ao tempo de implementação da funcionalidade cuja duração típica é de um a três dias, eventualmente uma semana mas nunca dez ou mais dias para implementar. A lista de funcionalides é possui a forma de uma árvore de três níveis de hierarquia.

O planejamento por funcionalidade que é a fase seguinte a lista de funcinalidades compreende a criação de um cronograma inicial e atribuição de responsabilidades a cada membro da equipe de desenvolvimento. Nessa etapa, também ocorre a distribuição de funcionalidades entre os membros da equipe. A sequência de implementação das funcionalidades não é aleatória, mas é planejada cuidadosamente levando em consideração as dependências e riscos.

Na fase seguinte, projetar por funcionalidade, as funcionalidades serão detalhadas. Caso uma funcionalidade seja complexa, um especialista de negócio pode conduzir um novo estudo dirigido para esclarecer e tirar dúvidas a respeito da funcionalidade. A saída desta fase são alguns diagramas de sequência e a inclusão de atributos e métodos no modelo abrangente da fase 1.

Na fase de construção por funcionalidade, as funcionalidades são codificadas e testadas. Em seguinda, o código implementado passa por uma inspeção no código. Segundo [McConnell, Code Complete, Microsoft], através da inspeção de código é possível identificar

mais defeitos e mais tipos diferentes de defeitos do que técnicas de teste.

2.6 Modelagem em Cores

Segundo (PALMER, 2009), a Modelagem em Cores é um conjunto de padrões e estratégias que ajudam a produzir modelos e uma análise orientada a objetos melhores. A história da modelagem em cores está estreitamente ligada a história do FDD. Foi criada por Peter Coad em 1997 mas tarde aprimorada por ele e por seus colegas de trabalho.

Para endender a Modelagem em Cores é preciso entender antes o que é um arquétipo de classe. Segundo o dicionário da Priberam da Língua Portuguesa, um arquétipo é um modelo ideal, inteligível, do qual se copiou toda a coisa sensível. Os arquétipos de classe são resultado da observação em diversos projetos na área de software de classes de domínio do problema recorrentes. Um exemplo, seria os sistemas empresarias em que existem classes que representam **transações** ou **iterações** de negócio de diferentes tipos, tais como vendas, aluguéis, reservas, eventos programados, entregas, submissões, aprovações, etc. Além disso, há geralmente classes que modelam as **partes**, os **locais** e **coisas** que participam nestas transações ou iterações. Por exemplo há pessoas ou organizações comprando ou vendendo. Por conseguinte há produtos e serviços sendo comprados. Há também lugares onde tudo isso acontence tais como lojas, web sites, bancos, etc. O exemplo acima pode ser ampliado para outros tipos de sistemas como sistemas industriais. Nos sistemas industriais temos geralmente classes que modelam **eventos** ou *intervalos de tempo*, por exemplo o estado de um sensor ou chave, ou período de tempo em que foram realizadas certas medições. As classes partes, locais e coisas podem representar operadores humanos de equipamentos, a localização de partes de um equipamento e os sensores e dispositivos, respectivamente.

Segundo, (PALMER, 2009) As semelhanças entre todas as classes de uma categoria geralmente não são suficientes para generalizá-las usand a herança da programação orientada a objetos. Entretanto, estas semelhanças são próximas o bastante para servir como guia para escolha e definição de classes de uma modelagem. Na modelagem em cores, estas categorias de classe são chamadas de arquétipos.

Em (COAD; LEFEBVRE; DELUCA, 1999) temos definidos quatro arquétipos de classes: Momento-Intervalo (*Moment-Interval*), Parte (*Party*) , Papél (*Role*) e Descrição (Description). O nome Modelagem em Cores deve-se as cores aplicadas a estas classes quando representadas no UML, que são, respectivamente, rosa, verde, amarela e roxa.

Segundo (COAD; LEFEBVRE; DELUCA, 1999), o Momento-Intervalo refere-se a algo que ocorre em um momento no tempo ou em um intervalo de tempo. Esse instante ou intervalo de tempo precisa ser gravado. O cadastro de um usuário é um exemplo de algo que ocorre em um instante de tempo. Um exemplo de intervalo é a criação de uma turma para determinado curso. Outros exemplos de Momento-Intervalo são aluguéis, reservas, reuniões, etc.

O *Papel* modela como as classes irão participar nos momentos-intervalos. Exemplos de papéis em nossa sistemas de cursos livres são professores, alunos, funcionários e administradores.

As *Partes* modelam alguém ou alguma coisa que pode assumir diferentes papéis. O arquétipo *Parte* pode ser decomposto em três: Parte, Locais e Coisas. Exemplos de classes Locais são escritório, lojas, aeroportos, agência bancária, sala de aula, etc. Exemplos de classes Partes são pessoas e organização. A classe Coisas, incluem objetos como carros, aviões, DVDs, livros, etc.

A classe *Descrição* modela uma descrição do tipo entradas em um catálogo.

Quando juntamos todas as classe arquetípicas através de relacionamentos entre seus objetos obtemos o que Peter Coad chamou de *Domain Neutral Component* (Componente de Domínio Neutro) ou abreviadamente DNC. O DNC é pode ser visto na Figura 2.6 localizada na página 27, com seus atributos e relacionamentos típicos.

O DNC é usado como para auxiliar a construção de modelos do domínio do problema que se tem interesse. Segundo (PALMER, 2009) Os passos básicos para usar o DNC são:

- Identifiquer uma mais classes do domínio do problema que sejam Momento-Intervalos.
- Substitua os Momento-Intervalos no DNC pelas classes encontradas.
- Repita o procedimento para outras classes do DNC.

- Ignore as classes arquetípicas que não tem equivalência em nenhuma classe do domínio do problema.

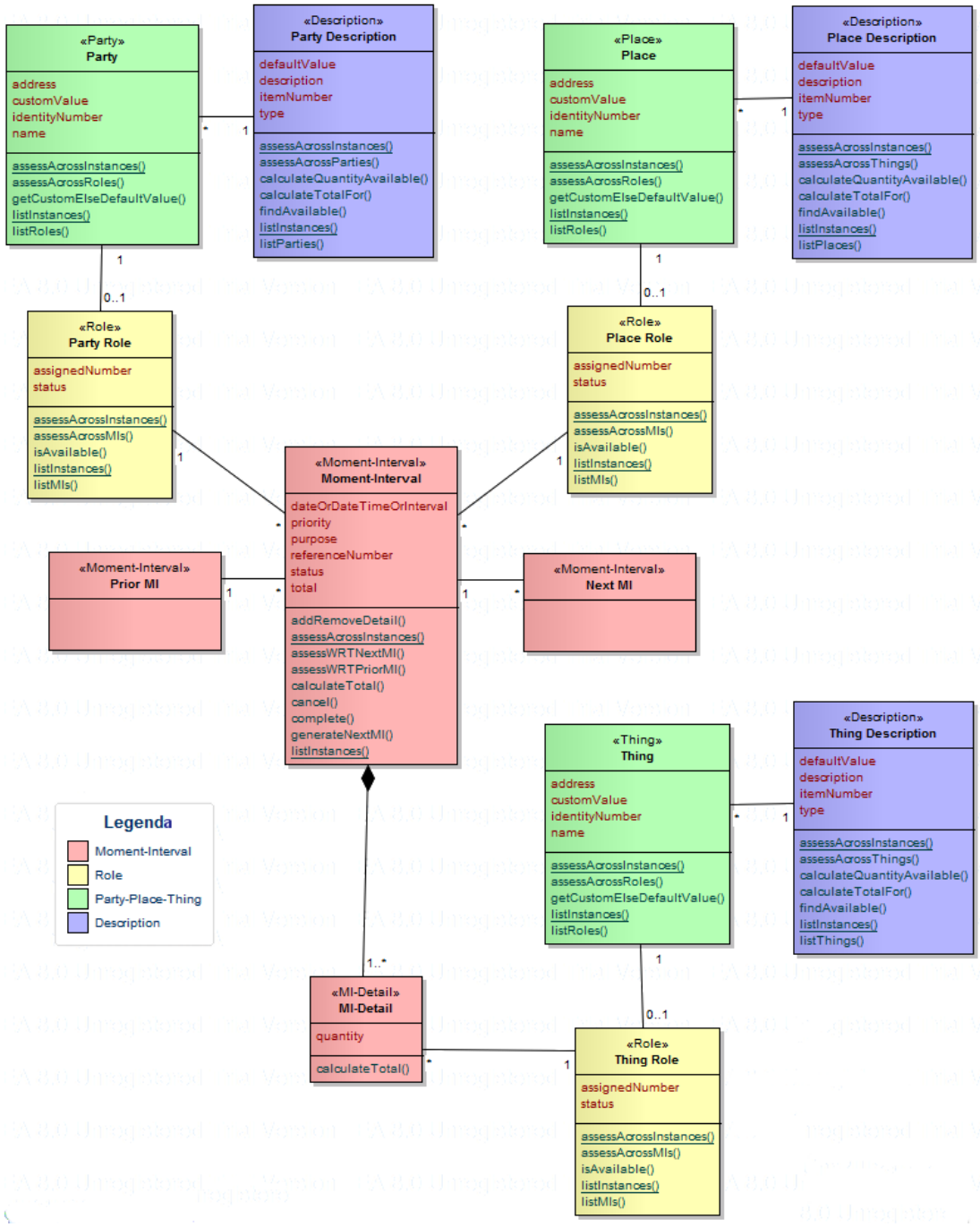

Figura 2.6: DNC - *Domain Neutral Componente*.
Fonte: `www.step-10.com`

Capítulo 3

Integração dos *Frameworks* CakePHP e Flex

O *framework* Flex inclue três componentes RPC (*Remote Procedure Call*) que permitem a integração de aplicações Flex com aplicações do lado servidor. Todos os três componentes comunicam-se com o servidor através do protocolo HTTP. Neste trabalho, concentra-se apenas em um componente, o *RemoteObject*.

O componente *RemoteObject* codifica as chamadas de métodos remotos no formato AMF3. Como discutido no capítulo anterior, o AMF3 é o formato no qual os objetos da linguagem ActionScript são serializados. No entanto, o AMF pode ser usado para codificar chamadas a procedimentos remotos.

A vantagem de se utilizar AMF3 para troca de dados em relação ao XML, é o tamanho do pacote final que será transmitido através da rede. A economia em tamanho de pacotes é devida as informações estarem codificadas em formato binário e comprimidas. O protocolo AMF3, também, trata múltiplas ocorrências de um mesmo objeto. Por exemplo, toda cadeia de caracteres (*String*) é mapeada para um identificador. Desta forma, um ou mais objetos poderão referenciar a cadeia de caracteres por meio do seu identificador evitando a duplicação.

Para realizar a integração entre Flex e o CakePHP precisaremos de um servidor de objetos AMF3 para mapear os objetos PHP em objetos AMF3. Um servidor bastante documentado e estável para este fim é o AMFPHP.

O AMFPHP[1] é um projeto de código-aberto que oferece uma implementação do protocolo AMF na linguagem PHP. O principal objetivo do AMFPHP é permitir que aplicações leves escritas em Flash ou Flex possam trocar mensagens com aplicações escritas em PHP.

A comunicação entre o cliente e o servidor, por exemplo Flex e PHP, é alcançada através de chamadas de métodos remotos e do intercâmbio de dados serializados.

O cliente (visualização) serializa uma requisição e envia os dados para um programa denominado *Gateway*. O AMFPHP deserializa a requisição e encontra a classe remota correspondente, e em seguida instancia a classe e realiza as verificações de segurança. Depois, o método remoto é invocado usando os parâmetros especificados na requisição. Os dados de retorno são serializados e enviados como resposta a requisição para o cliente. A Figura 3.1 mostra esquematicamente esse processo.

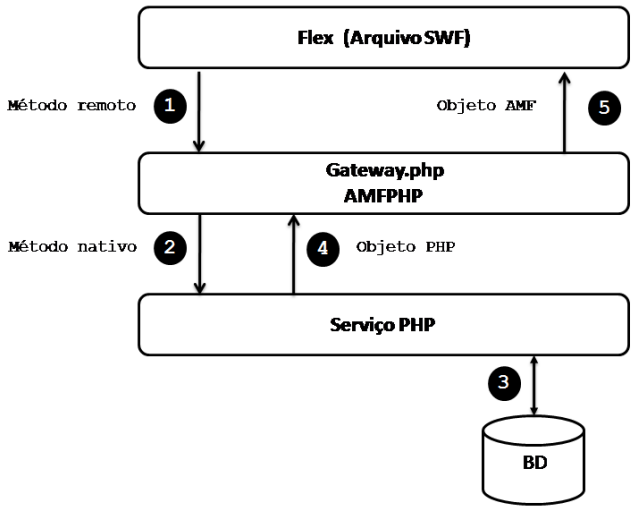

Figura 3.1: Funcionamento esquemático da comunicação Flex-PHP.

[1]http://www.amfphp.org/docs

Há um *plugin* para o CakePHP chamado Cpamf[2] que encapsula o AMFPHP e permite que este seja utilizado dentro do CakePHP. Portanto, utilizaremos esse *plugin* para demonstrar a comunicação com o Flex.

3.1 Instalação e configuração do *plugin* Cpamf

Primeiramente, faça o *download* do arquivo no endereço [3] e depois descompacte-o no diretório `CAKE_HOME/app/plugins`. Segundo a documentação do Cpamf, é necessário programar o CakePHP para que as URLs do Cpamf sejam mapeadas corretamente em ações do controle CpamfController. Isso é alcançado pela configuração de rotas (*Routes*). A Listagem 3.1 mostra como a rota foi configurada para este trabalho. O arquivo de configuração encontra-se no caminho `CAKE_HOME/app/config/routes.php`.

```
Router::connect('/cpamf/:action/*',
                array('plugin'=>'cpamf','controller' => 'cpamf'));
```

Listagem 3.1: Configuração do arquivo *routes.php*.

Para testar a instalação e configuração do CPAMF é suficiente acessar a URL do Cpamf, conforme a Figura 3.2. Uma página HTML será mostrada no navegador informando que o AMFPHP e o *gateway* foram instalados corretamente.

A ferramenta Flex *Service Browser*, a qual vem incorporada dentro do AMFPHP permite ao desenvolvedor testar e realizar *debug* de funções e métodos de serviços implementados na linguagem PHP.

Para utilizar a ferramenta é necessário clicar no *hyperlink Load the service browser* ou acessar o endereço `http://localhost/<APP>/cpamf/gateway`. Ao acessar o *hyperlink* pela primeira vez será exibida a tela de configuração do *Service Browser*, conforme ilustrado na Figura 3.3. Embora a janela de configuração do *Service Browser* sugira um valor para

[2]http://bakery.cakephp.org/articles/vernerd/2009/06/28/flex-remoting-with-cakephp-cpamf-plugin-1.

[3]`http://carrotplant.com/public/files/cpamf/cpamf_v0_12.zip`

Figura 3.2: *Gateway* do Cpamf.

o parâmetro *Gateway location*, o correto é remover a extensão *.php* da URL e clicar no botão *Save*.

Na janela principal do *Service Browser*, Figura 3.4, pode-se observar os controles presentes no diretório `CAKE_HOME/app/controllers` e os métodos da classe *TurmasController*. Para chamar um método de um determinado controle clica-se no método desejado e pressiona-se o botão *Call*. O resultado da chamada é mostrado em um conjunto de abas que é exibido logo abaixo do botão *Call*. Caso um método possua parâmetros, são exibidos campos de texto para cada parâmetro do método.

3.1.1 Parâmetro de Codificação de Caracteres

A codificação de caracteres padrão do AMF3 é o UTF-8, por isso é necessário garantir que as cadeias de caracteres nativas do servidor sejam codificadas em UTF-8 antes de

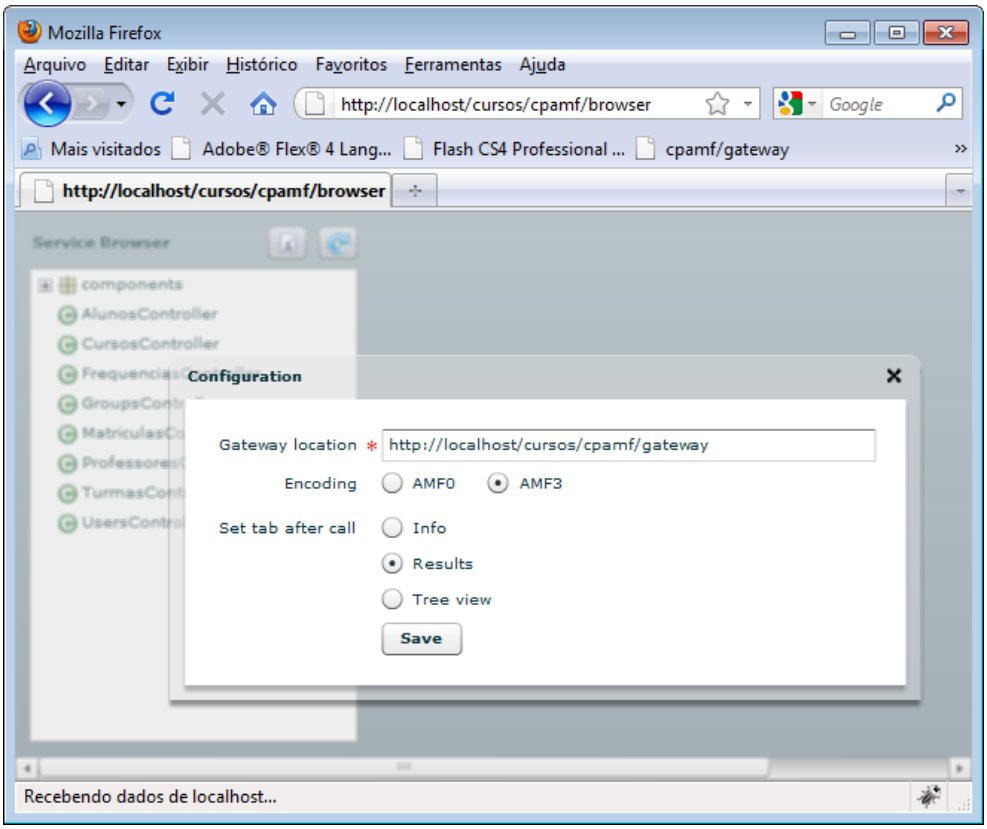

Figura 3.3: Configuração do *Service Browser*.

enviá-las para a aplicação cliente e vice-versa. Para esclarecer, considere um SGBD que armazene *strings* codificadas em Latin-1 e uma consulta que retorne os nomes dos cursos ministrados em uma faculdade. Para que os nomes dos cursos que contenham acentos sejam visualizados corretamente, é necessário configurar o AMFPHP para que ele proceda a conversão de *strings* no formato Latin-1 para o formato UTF-8. É possível configurar o tipo de conversão por meio do método `setCharsetHandler` da classe Gateway. Isto é feito no arquivo gateway.php no diretório do AMFPHP. A listagem 3.2 exemplifica como fazer essa configuração para o exemplo anterior.

```
$gateway->setCharsetHandler(
        "utf8_decode",  //Converte strings Latin-1 em UTF-8
        "Latin-1", // String PHP
        "Latin-1"  // String SQL
);
```

Listagem 3.2: Parâmetro de codificação de caracteres.

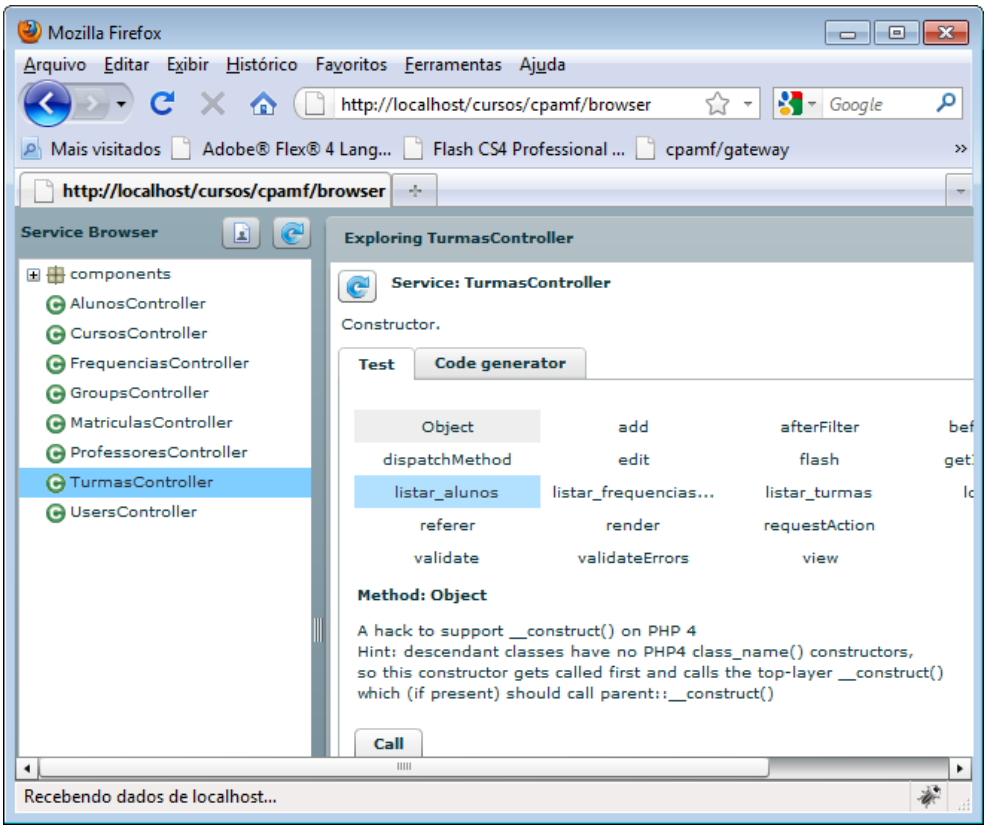

Figura 3.4: Janela para explorar Controles e métodos

Embora não esteja documentado, use o arquivo `<APP>/plugins/cpamf/vendors/amfphp/cake_gateway.php` para configurar a codificação de caracteres no *plugin* Cpamf. Este arquivo PHP é uma versão otimizada para uso com o *framework* CakePHP.

Neste trabalho, o SGBD foi configurado para armazenar *strings* codificadas em UTF-8, portanto, evitando o *overhead* da conversão de *strings*. A configuração utilizada para essa instalação está ilustrada na listagem 3.3

```
$gateway->setCharsetHandler(
        "none",        // Desativa conversão
        "ISO-8859-1",  // String PHP
        "ISO-8859-1"   // String SQL
);
```

Listagem 3.3: Parâmetro de codificação de caracteres.

3.1.2 Mapeamento de Objetos

Um dos pontos fundamentais do funcionamento do AMFPHP é a conversão de objetos PHP em objetos ActionScript. A fim de entender melhor, considere um objeto PHP chamado Aluno e sua contraparte em AS3 chamada AlunoVO. Considere, também, um método no CakePHP chamado *listar_alunos* que devolve a lista de alunos cadastrados de uma instituição de ensino. Antes de transmitir a resposta a solicição *listar_alunos*, o AMFPHP converte cada objeto Aluno em PHP na respectiva contraparte ActionScript, AlunoVO. Dessa forma, a aplicação Flex cliente poderá restaurar esses objetos na memória e apresentá-los adequadamente.

Uma dificuldade que surge quando tentamos integrar o CakePHP com o AMFPHP é o fato de que as consultas realizadas pelos Modelos do CakePHP sempre tem como resultado um ou mais registro codificados em *arrays* associativos. Embora isso não seja um problema grave, é mais conveniente termos a mão um *array* de objetos. A razão dessa necessidade será explicada na próxima seção, quando falaremos da configuração do Flex. Por hora, mostraremos apenas uma solução para este problema.

A solução inicial foi encontrada no *site Bakery CakePHP*[4]. Esta consistia em utilizar uma função de *callback*[5] para converter o *array* associativo em objetos antes do Modelo retornar a resposta para o Controle. Entretanto, esta abordagem produzia um efeito colateral nas Visualizações em HTML. Estas paravam de funcionar corretamente pois usam arrays associativos para guardar os dados que serão exibidos em tela.

Para contornar esse problema foi criado um Componente para converter arrays associativos em objetos. Este componente é utilizado sempre que uma requisição origina-se do Flex.

De acordo com (FOUNDATION, 2010a), um componente é uma classe que auxilia a lógica de um Controle. É importante lembrar que um componente não é disponibilizado automaticamente em um objeto Controle, é necessário adicioná-lo por meio do atributo *$components* disponível para qualquer classe que herde da classe Controller. A Listagem

[4] `http://bakery.cakephp.org/articles/vernerd/2009/06/28/flex-remoting-with-cakephp-cpamf-plugin-1`

[5] *Callbacks* são funções invocadas quando um evento ocorre.

3.4 ilustra como disponibilizar um componente para um Controle.

```
var $components = array('Util', 'Acl', 'Auth', 'Session');
```

Listagem 3.4: Componentes disponibilizados para uso em um Controle.

3.1.3 Configuração dos Serviços Remotos no Flex

Como foi mencionado anteriormente, utilizaremos o serviço RPC para comunicação entre a aplicação em Flex e o *framework* do lado do Servidor. O RPC é usado para duas finalidades. A primeira finalidade tem haver com receber dados do servidor para serem manipulados pela aplicação (cliente) e a segunda para realizar chamadas de métodos no servidor. Já foi visto que usamos a biblioteca AMFPHP, encapsulada no *plugin* Cpamf, para serialização em AMF3. Agora veremos como configurar o Flex para estabelecer a outra ponta da comunicação.

Para que o Flex seja capaz de invocar métodos dos Controles do CakePHP é necessário configurar um *EndPoint*. O *EndPoint* é o endereço de rede do servidor que receberá as requisições RPC. Esta configuração pode ser feita em um arquivo XML geralmente nomeado como *services-config.xml*. Para exemplificar, a listagem 3.5 mostra um trecho da configuração utilizada neste trabalho. Note que o *EndPoint* referenciado no arquivo *services-config.xml* aponta para a ação *gateway* no *plugin* Cpamf. O parâmetro *destination* será explicado na próxima seção.

O outro ponto importante de arquivo de configuração é o parâmetro *class* do *channel-definition*. Na listagem 3.5, este aparece configurado como AMFChannel, visto que o protocolo AMF3 será o canal de comunicação. O arquivo *services-config.xml* será utilizado na fase de compilação quando este for passado como parâmetro para o compilador. Desta forma, a aplicação cliente conhecerá o endereço do servidor para troca de mensagens. Note que, configurando o *EndPoint* desta maneira, uma aplicação em Flex apenas será capaz de comunicar-se em AMF3 e através do endereço de rede especificado. Caso o endereço do servidor mude, será necessário alterar o *EndPoint* e recompilar a aplicação.

```
<?xml version="1.0" encoding="UTF-8"?>
<services-config>
    <services>
        <destination id="amfphp">
                <channel ref="cake-amfphp"/>
        </destination>
    </services>
    <channel-definition id="cake-amfphp"
        class="mx.messaging.channels.AMFChannel">
        <endpoint uri="http://localhost/scl/cpamf/gateway"
            class="flex.messaging.endpoints.AMFEndpoint"/>
    </channel-definition>
</services-config>
```

Listagem 3.5: Trecho do arquivo *services-config.xml*

3.1.4 Classes de serviços em AS3

Na seção anterior foi visto como se configura o canal de comunicação para que cliente e o servidor possam trocar pacotes AMF através da rede. Nesta seção, conheceremos como API[6] RPC do Flex acessa objetos no servidor, remotamente.

O API de RPC do Flex disponibiliza o objeto RemoteObject para acesso remoto a objetos em um servidor. Para ilustrar, utilizaremos as classes Servico e UsuarioServico que compõem a aplicação de demonstração deste trabalho.

O funcionamento da classe Servico é bem simples. O diagrama da Figura 3.5 ilustra a relação de herança entre a classe Servico e UsuarioServico. O Método construtor da classe serviço é usado para inicializar dois parâmetros do RemoteObject. O primeiro parâmetro faz com que apareça um cursor de ocupado na tela quando houver um método remoto sendo executado. O segundo parâmetro configura a propriedade *destination* do RemoteObject para *amfphp*. Desse modo, toda chamada remota será encaminhada para o *Gateway* correto.

A classe RemoteObject disponibiliza os métodos obter e definir valores para o atributo *source*. O valor que atribuímos ao *source* é o nome da classe controle da qual chamaremos métodos remotos. Por exemplo, para que a classe UsuarioServico seja capaz de chamar o

[6]Application Programming Interface

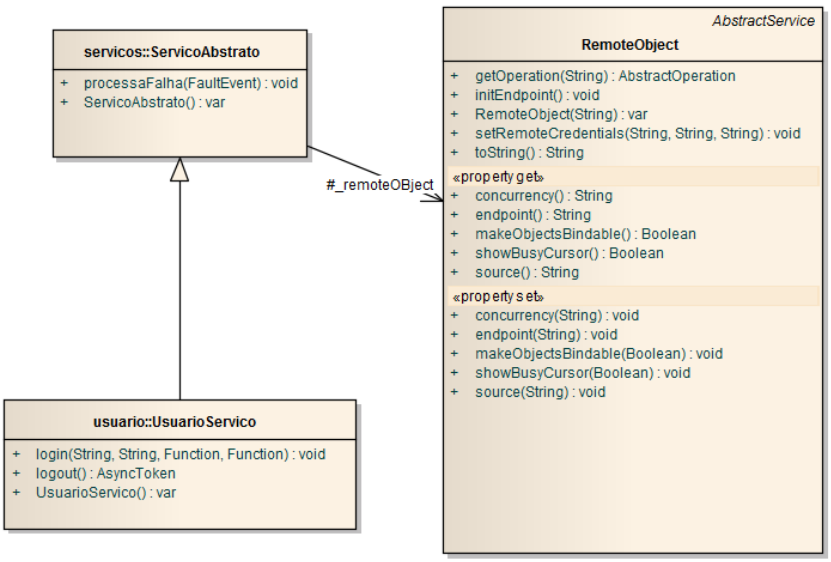

Figura 3.5: Hierarquia Servico/UsuarioServico.

método remoto *login* é necessário atribuir ao *source* o valor *UsersController*.

Em nossa implementação, fizemos com cada subclasse da classe Servico atribuísse o valor de *source* apropriado. O nome dos serviços é formado pelo o nome do controle mais o sufixo Servico. Deste modo, a classe TurmaServico corresponde ao TurmasController, e assim por diante.

Capítulo 4

Estudo de caso: Sistema de Controle Acadêmico

Neste capítulo apresentaremos um sistema de controle acadêmico para demonstrar a integração entre os *frameworks* Flex e CakePHP. Seguindo a metodologia FDD, apresentaremos o modelo de objetos proposto juntamente com o modelo de banco de dados. Em seguida, mostraremos a lista de funcionalidades (*Feature List*, no jargão do FDD) que o sistema será capaz de desempenhar. Após esta, apresentaremos alguns procedimentos e algoritmos importantes e as telas do sistema que são o produto final desde trabalho.

4.1 Modelagem dos Objetos do domínio do problema

A diagrama de classes da Figura 4.1, na página 39, ilustra o como funciona o gerenciamento de matrículas realizado por um usuário com perfil de funcionário. Note que as classes verdes são especializações da classe *cake::libs::model::Model*. A classe *Model* fornece para as suas subclasses métodos CRUD (*Create, Retrieve, Update, Delete*) e outros métodos utilitários.

As classes *momento-intervalos* (de cor rosa), são subclasses de

cake::libs::controller::Controller. Esta classe fornece aos seus descendentes a capacidade de manipular e responder as requisições de usuários. O arquétipo *papel* funcionário não é uma classe. Na verdade, funcionário é um grupo do sistema. Este diagrama foi mantido informar que somente usuários do grupo funcionário podem participar do gerenciamento das turmas.

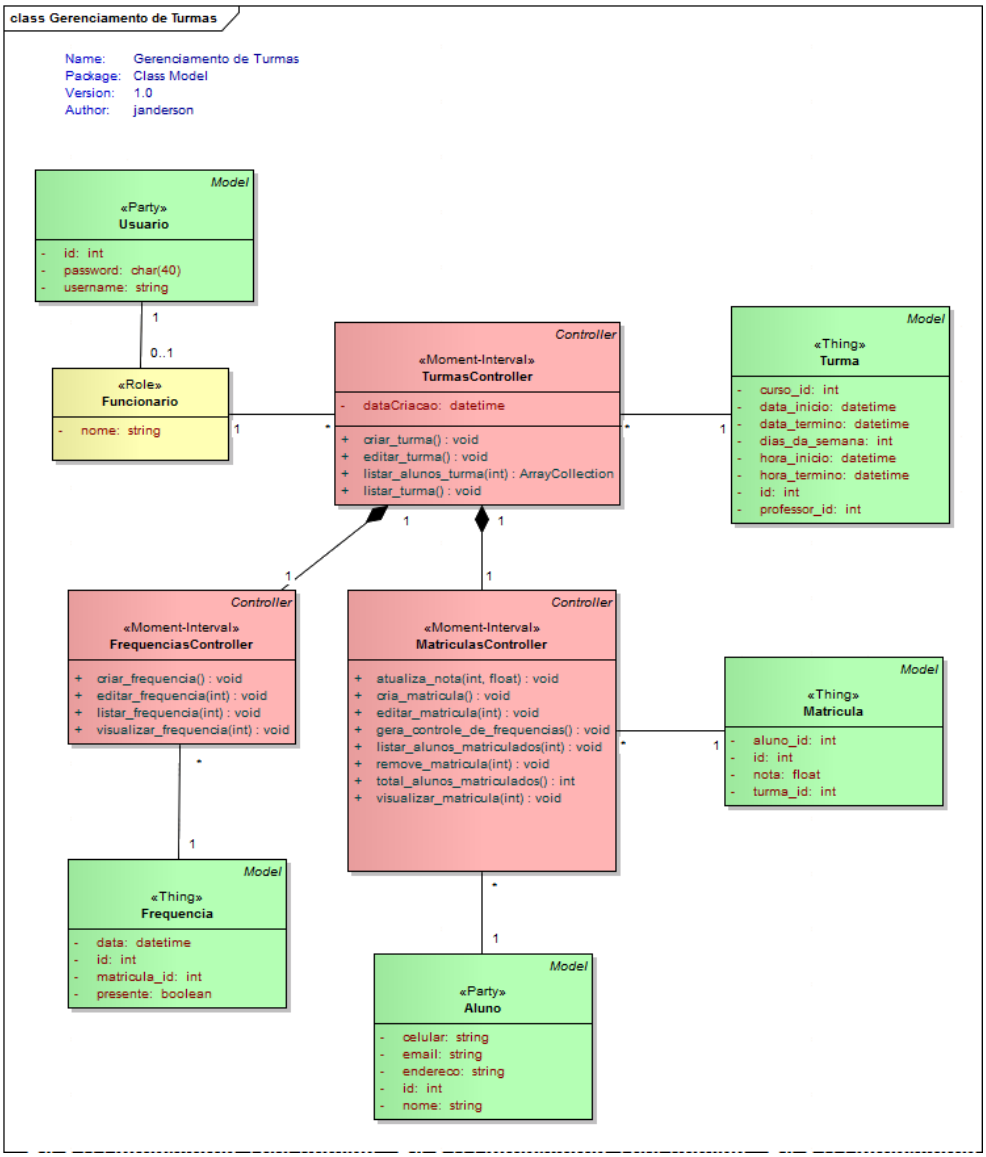

Figura 4.1: Modelo Gerenciamento de Turmas.

4.2 Casos de Uso

Os casos de uso foram organizados em 3 grupos, gerenciamento de usuários e grupos, gerenciamento de turmas pelo professor e pelo funcionário da instituição. Os três casos de uso estão ilustrados pelos diagramas de caso de uso .

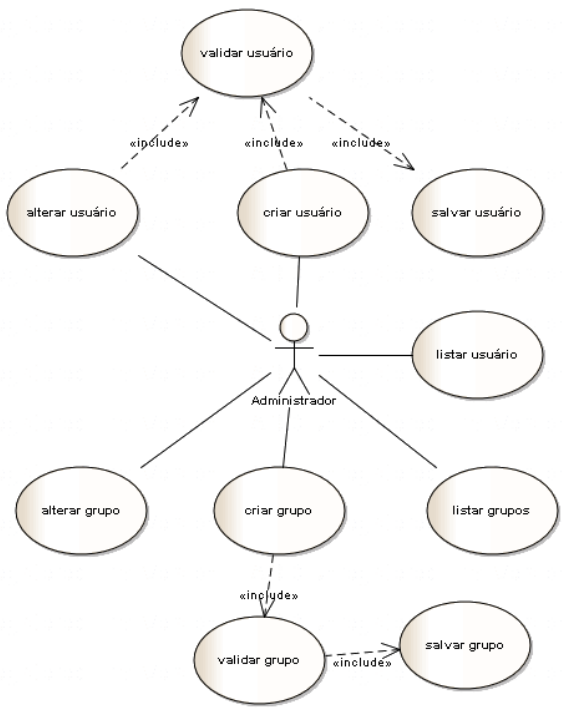

Figura 4.2: Caso de Uso Gerenciamento de Usuários.

4.3 Lista de Funcionalidades

As funcionalidades do SCL estão agrupadas em três grupos de funcionalidades: Gerenciar Turmas, Gerenciar Turmas Ministradas e Gerenciar Usuários e Grupos.

O detalhamento da funcionalidade gerenciar turmas está ilustrado na Figura 4.5.

O detalhamento da funcionalidade gerenciar turmas ministradas está ilustrado na Figura 4.5.

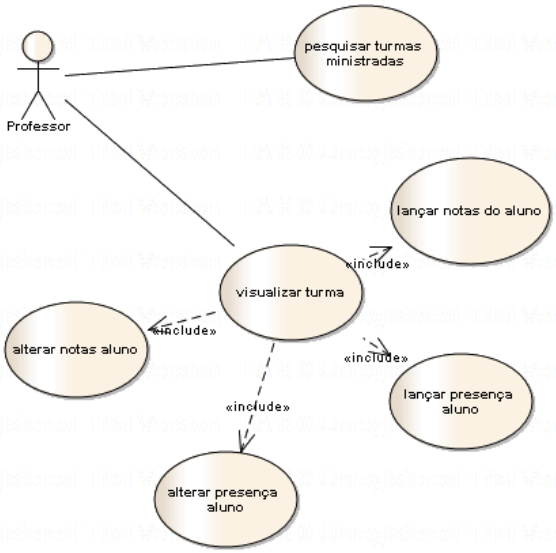

Figura 4.3: Caso de Uso Gerenciamento de Turmas Ministradas.

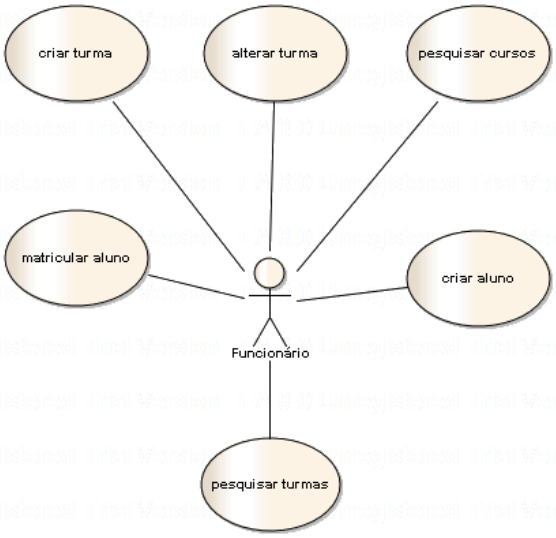

Figura 4.4: Caso de Uso Gerenciamento de Turmas e Matrículas.

O detalhamento da funcionalidade gerenciar turmas está ilustrado na Figura 4.5.

Funcionário	**Gerenciar Turmas**		
		Criar uma turma	
			validar turma
			alterar professor da turma
			alterar curso da turma
			salvar turma
		Matricular aluno	
			validar aluno
			salvar aluno
			alterar dados do aluno
		Pesquisar turmas	
			listar turmas
	Gerenciar Cursos		
		Pesquisar Cursos	
			Listar turmas de determinado curso
			Listar cursos ministrados de professor informado
	Gerenciar Alunos		
		Criar aluno	
			validar aluno
			salvar aluno
			alterar aluno
			listar cursos que aluno está matriculado
	Relatórios		
		Criar relatório de notas	

Figura 4.5: Detalhamento do Gerenciamento de Turmas.

Professor	**Gerenciar Turmas Ministradas**		
		Pesquisar turmas ministradas	
			listar turmas ministradas pelo professor atualmente logado no sistema
		Visualizar turma	
			lançar notas do aluno
			alterar notas do aluno
			lançar presença do aluno
			alterar presença do aluno

Figura 4.6: Detalhamento do Gerenciamento de Turmas Ministradas.

Admin	**Gerenciar Usuários**		
		Criar usuário	
			validar usuário
			salvar usuário
			alterar usuário
			listar usuários
		Criar Grupo	
			validar grupo
			salvar grupo
			alterar grupo
			listar grupo

Figura 4.7: Detalhamento do Gerenciamento de Usuários e Grupos.

4.4 Modelo Entidade-Relacionamento

A Figura 4.8 na página 43 ilustra o diagrama de Entidade-Relacionamento do Sistema de Gestão de Cursos.

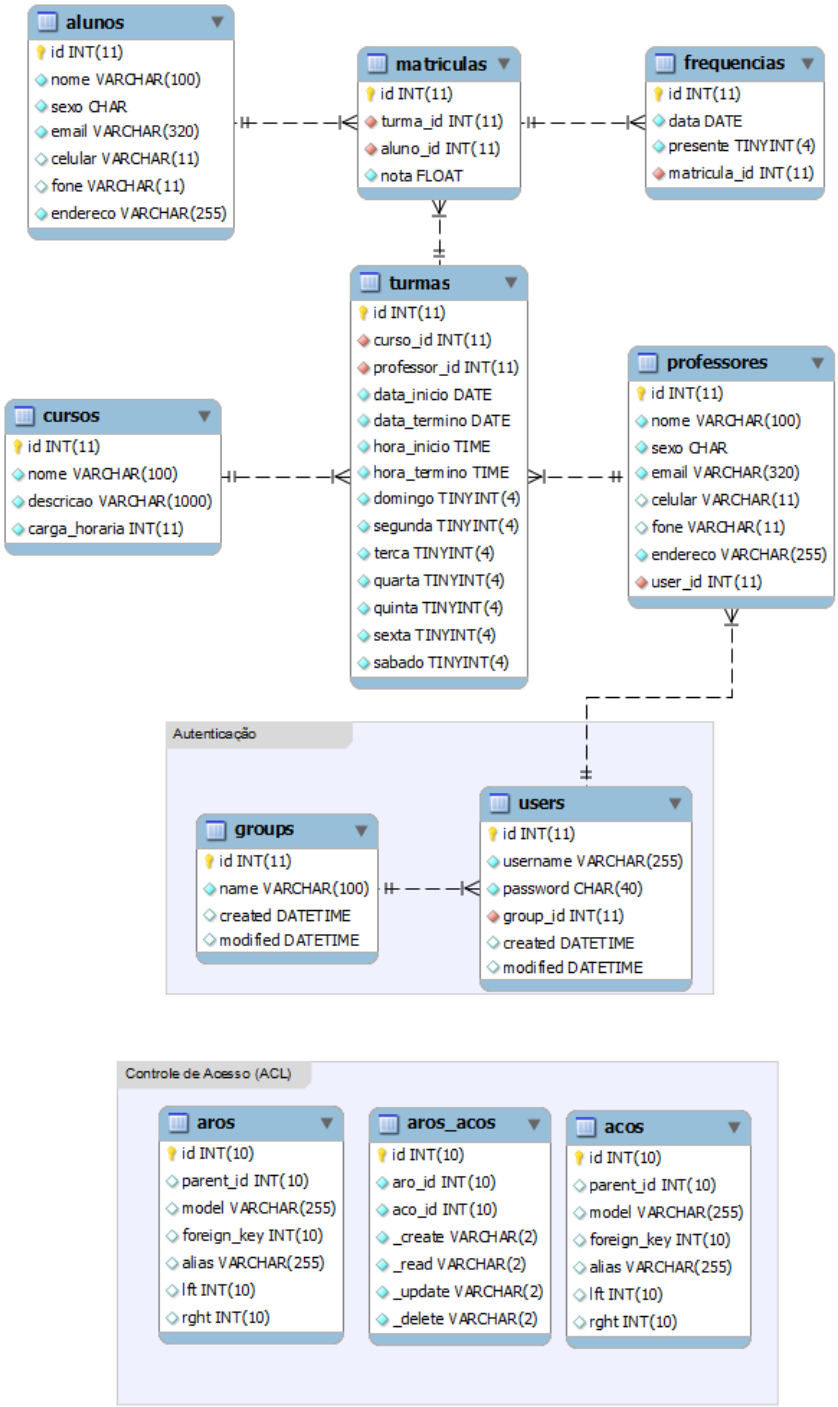

Figura 4.8: Modelo Entidade-Relacionamento

4.5 Arquitetura da Aplicação

O Sistema de Gestão de Cursos Livres pode ser entendido através de sua arquitetura geral, a qual mostra os passos principais do funcionamento da aplicação. A Figura 4.9

representa a arquitetura geral.

A Figura 4.9 ilustra as principais camadas do Sistema de Gestão de Cursos Livres: Visualização, Controle e Modelo. Estas correspondem ao padrão MVC(*Model-View-Controller*). Note que há uma camada extra, o Gateway.php/AMFPHP. Esta camada estabelece a comunicação entre as camadas de visualização e de controle por meio do protocolo AMF. A Figura 4.9, também, ilustra o fluxo de informações entre as camadas.

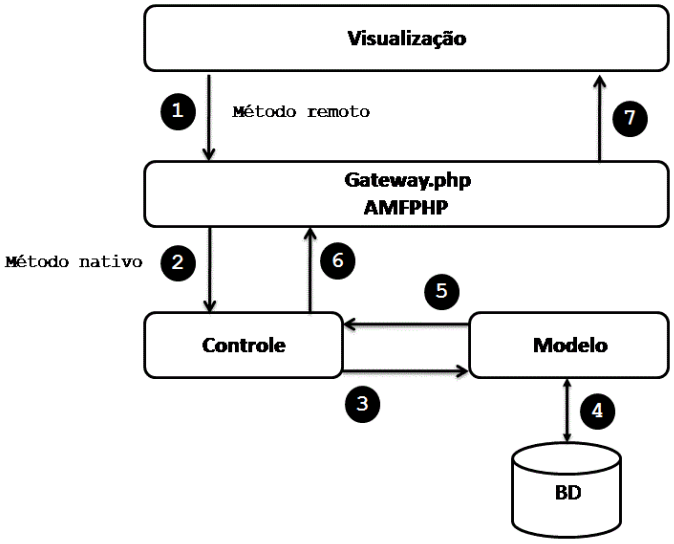

Figura 4.9: Arquitetura geral do SCl.

O fluxo de comunicação é iniciado na camada de visualização. Esta realiza uma chamada de método remoto (1) à camada de `Gateway.php/AMFPHP`. Esta camada interpreta as informações codificadas segundo o protocolo AMF e realiza a chamada ao método nativo, na camada do *Controlador* (2). No passo (3), a camada *Modelo* é invocada por meio de um método e transfere os dados do Banco de dados (4 e 5) para o *Controlador*. No passo (6) a resposta à chamada do método é transferida para o `Gateway.php/AMFPHP`. O ciclo termina, quando a camada de visualização recebe a resposta ao método remoto (7).

4.6 Gerenciamento de Usuários e Grupos

A funcionalidade gerenciamento de usuários e grupos compreendem as operações básicas para a manutenção do cadastro de usuários e grupos de usuários tais como: validar, salvar, listar, alterar e remover dados. Vale ressaltar que tais funcionalidades foram geradas através do gerador de códigos do CakePHP. Logo, não deremos muita ênfase a elas.

A Figura 4.10 mostra a forma de como é feito o preenchimento dos dados de um usuário. Portanto, detalhamos através desta as duas primeiras operações básicas, validar e salvar. Perceba que a validação ocorre nos campos obrigatórios sinalizados através do sinal de asterisco (*). O botão *submit* é o responsável pela realização da persistência dos dados em banco.

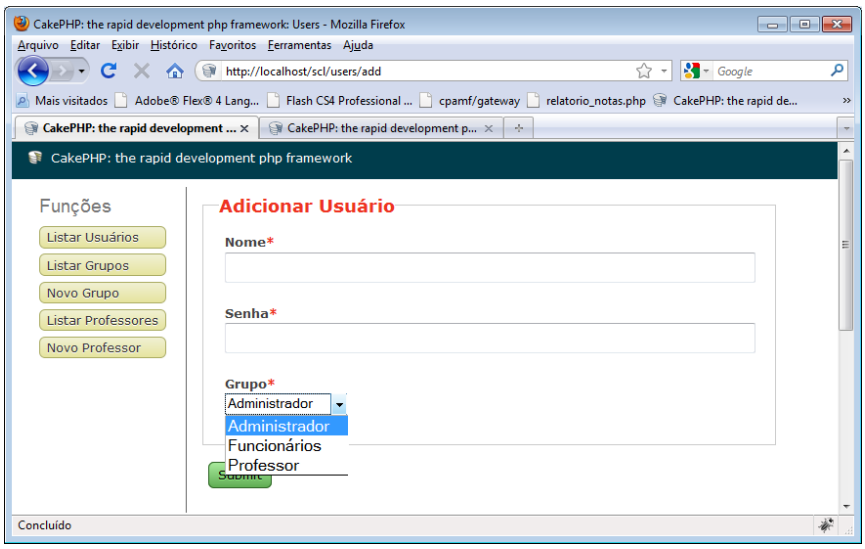

Figura 4.10: Tela Adicionar Usuário.

Em seguida, a funcionalidade listar dados pode ser acionada nesta mesma tela. Observe o menu ao lado chamado de funções, o qual mostra as funções relacionadas ao cadastro de usuário, especicamente listar dados, tais como: Listar Usuários, Listar Grupos, Listar Professores, e também funcionalidades de criação como Novo Grupo e Novo Professor.

Ressaltando que essas funções são geradas automaticamente pelo gerador de código CakePHP.

A Figura 4.11 ilustra a funcionalidade de alterar dados. Perceba que os dados ainda

são validados através dos campos obrigatórios. E, além de alterar, ou seja, modificar dados podemos também remover dados, o qual foi considerado pelo gerador uma alteração na persistência dos dados em banco.

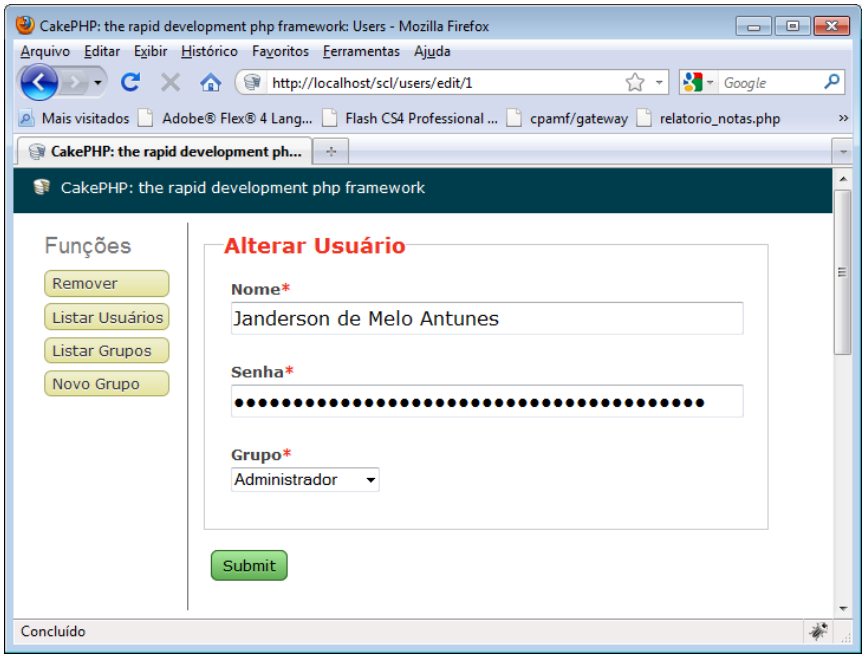

Figura 4.11: Tela Alterar Usuário.

A Figura 4.12 mostra os principais campos da tabela usuário. Note que a interface permite visualizar os detalhes do usuário, alterar e remover as informações do usuário.

Até o momento vimos funcionalidades relacionadas ao gerenciamento de usuários, agora veremos as que estão relacionadas ao gerenciamento de grupos de usuários.

A Figura 4.13 ilustra o cadastro de um novo grupo no sistema. Note que há apenas um campo obrigatório, tornando-a bem simples de ser usada. Através do botão *submit* se é gerado um novo grupo.

A Figura 4.14 ilustra a funcionalidade alterar dados de um grupo de usuários. Note que esta tela também permite excluir um grupo de usuáiros do sistema.

A Figura 4.15 ilustra a funcionalidade listar grupos de usuários cadastrados. Por meio desta tela é possível criar, alterar ou remover um grupo.

Funções

Novo Usuário

Listar Grupos

Novo Grupo

Usuários

Id	Nome	Grupo	Criado em	Funções		
1	bianor.rubens	Funcionários	2010-11-21 15:43:01	Detalhes	Alterar	Excluir
2	americo.carmo	Funcionários	2010-11-21 15:43:01	Detalhes	Alterar	Excluir
3	vitalina.batista	Funcionários	2010-11-21 15:43:01	Detalhes	Alterar	Excluir
4	analice.ribeiro	Funcionários	2010-11-21 15:43:01	Detalhes	Alterar	Excluir
5	ozelito.vecchi	Funcionários	2010-11-21 15:43:01	Detalhes	Alterar	Excluir
6	elvis.pereira	Funcionários	2010-11-21 15:43:01	Detalhes	Alterar	Excluir
7	eladir.regina	Funcionários	2010-11-21 15:43:01	Detalhes	Alterar	Excluir
8	jamily.cristina	Funcionários	2010-11-21 15:43:01	Detalhes	Alterar	Excluir
9	rogerio.valeria	Funcionários	2010-11-21 15:43:01	Detalhes	Alterar	Excluir
10	marcones.maneca	Funcionários	2010-11-21 15:43:01	Detalhes	Alterar	Excluir

Página 1 de 9, exibindo 10 de 88 registro(s), iniciando no registro 1, terminando em 10

<< anterior | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | próximo >>

Figura 4.12: Detalhe da Tela Listar Usuário.

Figura 4.13: Tela Adicionar Grupo.

4.7 Gerenciamento de Cursos

As funcionalidades relacionadas ao gerenciamento de cursos são: criar, alterar, remover e listar cursos.

A Figura 4.16 ilustra o cadastro de um novo curso no sistema. Há três campos

Figura 4.14: Tela Alterar Grupo.

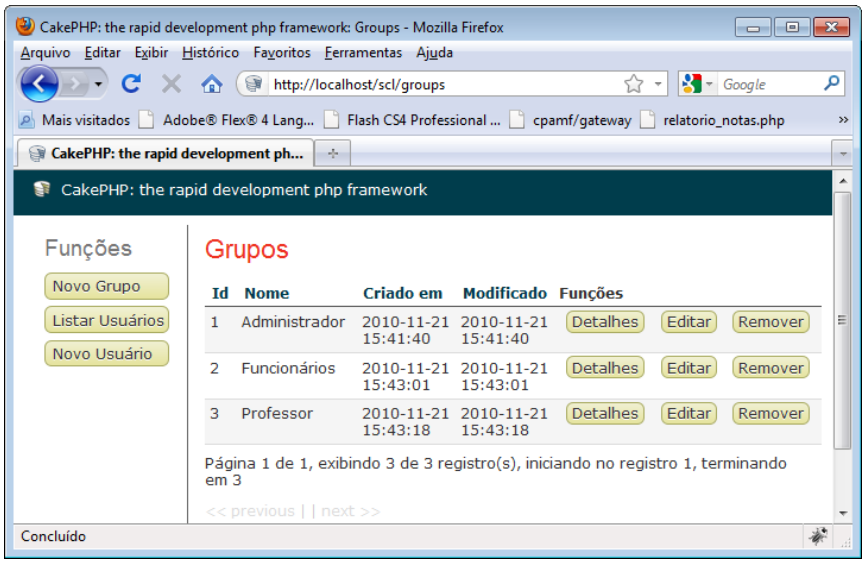

Figura 4.15: Tela Listar Grupo.

obrigatórios Nome, Descrição e Carga horária. O campo Carga horária tem um critério de validação extra, pois além de ser obrigatório deve ser um campo numérico.

A Figura 4.17 ilustra a funcionalidade alterar dados de um curso. Note que esta tela também permite excluir curso do sistema.

A Figura 4.18 ilustra a funcionalidade listar os cursos cadastrados no sistema. Por meio desta tela é possível criar, alterar ou remover um curso. Note também que é possível listar todas as turmas.

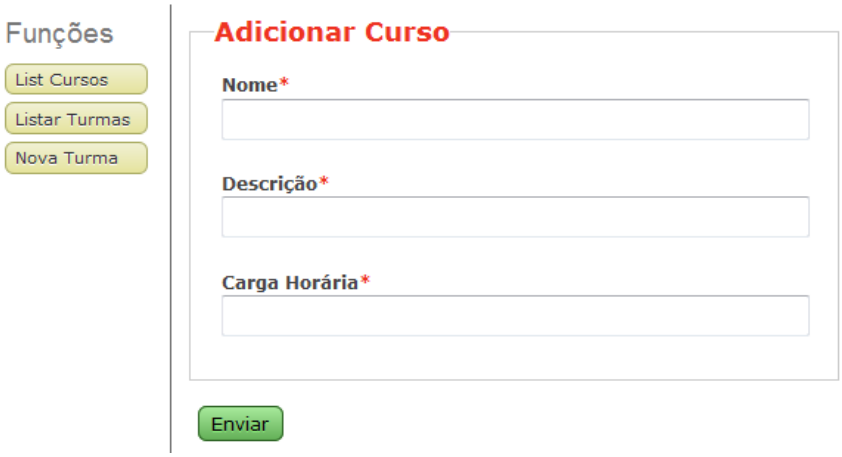

Figura 4.16: Tela Adicionar Curso.

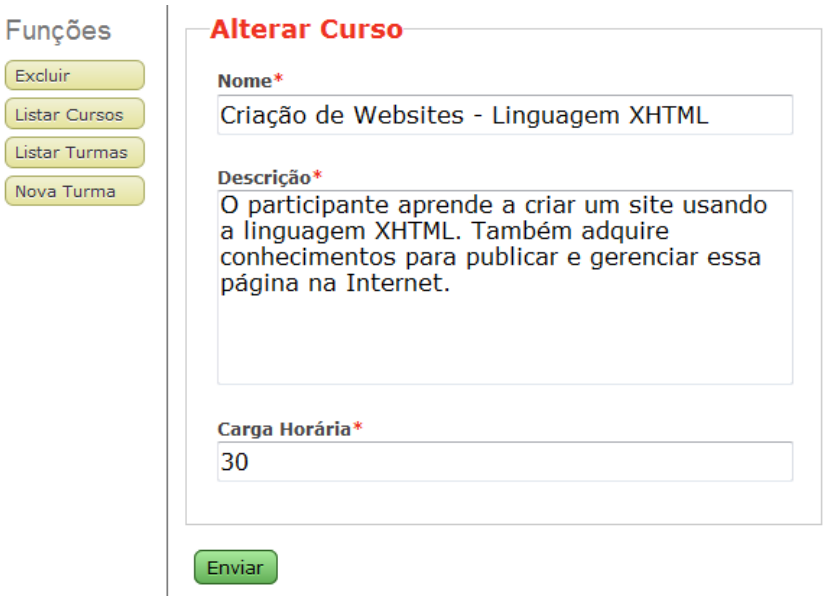

Figura 4.17: Tela Alterar curso.

4.8 Gerenciamento de Turmas

A funcionalidade Gerenciamento de Turmas além de possuir todas as funções típicas de aplicações CRUD, possui as regras de validação mais complicadas. A Figura 4.19 ilustra a tela de cadastro de turmas. Esta tela é a que contém o maior número e as regras de validação mais diversificadas. O campo Curso exige um curso existente.No campo curso é possível selecionar um entre os cursos oferecidos por meio de uma caixa de seleção. O

Funções

New Curso

Listar Turmas

Nova Turma

Cursos

Id	Nome	Descrição	Carga Horária	Funções
1	Criação de Websites - Linguagem XHTML	O participante aprende a criar um site usando a linguagem XHTML....	30	Detalhes Alterar Excluir
2	Caracterização no Sistema Tomaz Gomes em Prótese Total	A proposta do curso é oferecer ao participante o aprimoramento na...	16	Detalhes Alterar Excluir
3	Gerência e Supervisão de Vendas	O participante aprende a supervisionar/gerenciar a área de vendas...	24	Detalhes Alterar Excluir
4	Orçamento Empresarial	O participante aprende a elaborar o orçamento empresarial...	16	Detalhes Alterar Excluir
5	Prevenção e Atendimento à PCR em Crianças Hospitalizadas	O participante aprende como atuar na prevenção de PCR pediátrica,...	20	Detalhes Alterar Excluir
6	Excel 2010 - Avançado	O participante aprende a criar planilhas, partindo da análise de...	39	Detalhes Alterar Excluir
7	Cargos e salários	Este curso possibilitará ao egresso descrever, analisar, avaliar e...	45	Detalhes Alterar Excluir
8	Visual merchandising para o varejo de moda	O participante aprende a utilizar as principais técnicas de visual...	21	Detalhes Alterar Excluir
9	Espaços Confinados para Supervisor	O curso visa capacitar supervisores de entrada em espaços...	40	Detalhes Alterar Excluir
10	Norma SA 8000 - Gestão das práticas sociais do trabalho	O participante aprende a desenvolver um Sistema de Gestão de...	24	Detalhes Alterar Excluir

Página 1 de 30, exibindo 10 de 299 registro(s), iniciando no registro 1, terminando em 10

<< anterior | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | próximo >>

Figura 4.18: Tela Listar curso.

campo professor também é uma caixa de seleção. A duração é composta por dois campos: a data de início do curso e a data de término do curso.

Quantas as regras de validação de cada campo temos que o campo Professor é obrigatório e a validação contempla se o professor existe no sistema. Os campos Data de início e Data de término possuem três validações. Validam se o campo foi preenchido, se a data está formatada corretamente e se a data de término é maior ou igual que a data de inicio do curso. Os campos Hora de início e término tem regras análogas. Além disso, a hora término deve ser obrigatoriamente maior que a hora início. Quanto aos campos relativos aos dias da semana, pelo menos um deve ser preenchido.

Há ainda uma regra extra. Esta é aplicada quando todos os campos individuais foram validados. Quando uma turma é salva o procedimento de validação não permite que uma turma duplicada seja armazenada e exibe uma mensagem de erro alertando o usuário com um possível diagnóstico do problema. O caso básico de duplicação ocorre quando tenta-se gravar uma turma cujos horário de início, horário de término, duração e professor coincidam com os dados correspondentes de uma turma do banco de dados. Essa regra pode ser resumida pela seguinte premissa: Um professor não pode ministrar duas turmas, no mesmo horário e dia.

A regra de validação das turmas não leva em conta apenas o período inteiro de tempo durante a validação, mas é capaz de tratar sobreposições parciais, também. Um exemplo de sobreposição parcial de turmas é uma situação em que se tenha uma turma que sendo ministrada às segundas, terças e quartas no horário de 08:00 às 10:00 e outra turma que ocorra nas quartas, quintas e sextas, no horário de 09:00 ao meio-dia, sendo que o mesmo professor ministre as duas aulas.

A sobreposição ocorre por três motivos simultâneos: As duas turmas acontecem na quarta-feira, o horário das 09:00 às 10:00 horas é usado pelas duas turmas e professor ministra as duas aulas. Para que não ocorresse sobreposição, o usuário teria que alterar umas das variáveis. Ou trocaria o professor de uma das turmas ou mudaria o horário para que não coincidissem ou trocasse quarta-feira por sábado.

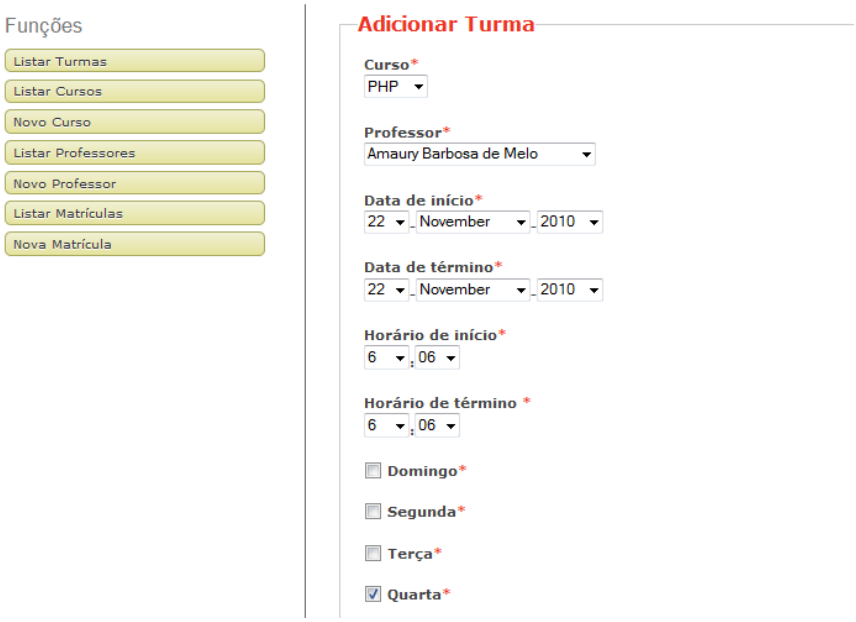

Figura 4.19: Tela Adicionar Turma.

A tela alterar turma é exibida na Figura 4.20. Todas as regras de validação supra-citadas aplicam-se a esta funcionalidade também.

A funcionalidade listar turmas, Figura é análoga as outras já citadas de modo que não repetiremos o que já foi comentado em outras telas semelhantes.

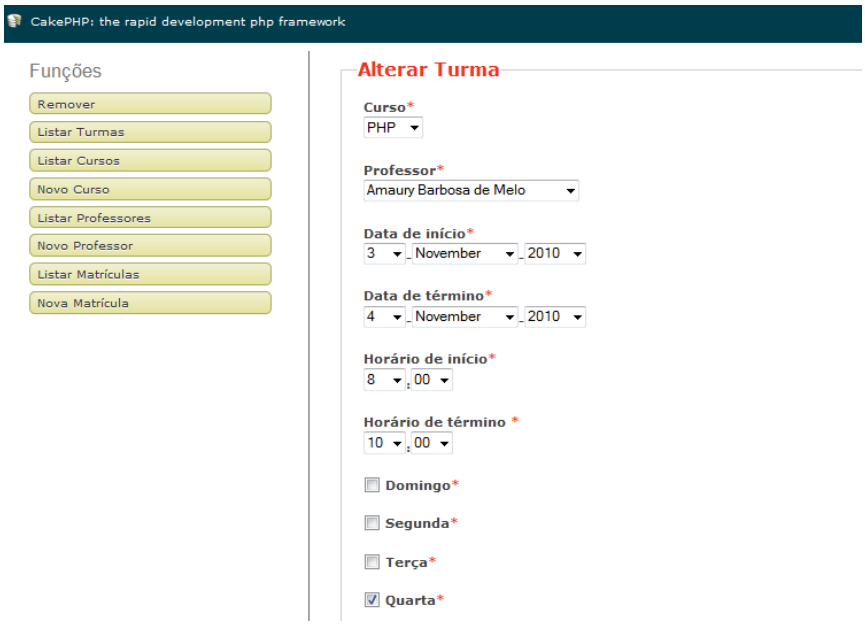

Figura 4.20: Tela Alterar Turma.

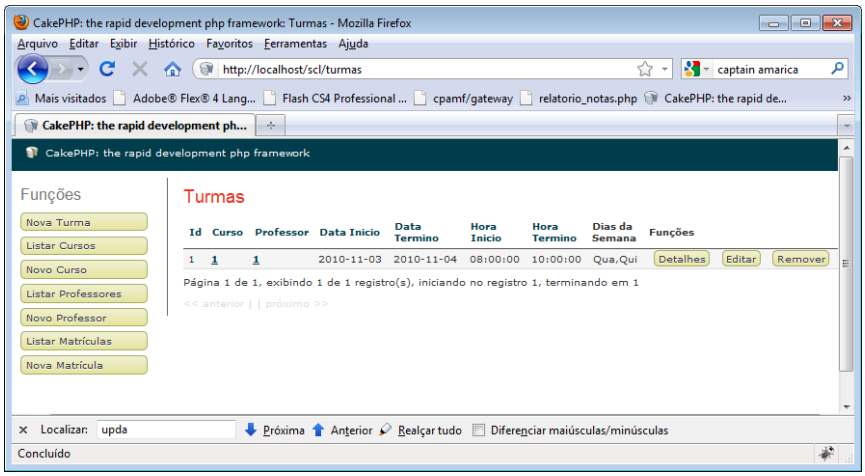

Figura 4.21: Tela Listar Turmas.

4.9 Gerenciamento de Alunos

As funcionalidades relacionadas ao gerenciamento de alunos são: criar, alterar, remover e listar cursos. A Figura 4.22 ilustra o cadastro de um novo aluno no sistema. Na Figura 4.23 temos implementada a funcionalidade alterar aluno. Por fim, na Figura 4.24, vemos ilustrada a funcionalidade listar alunos.

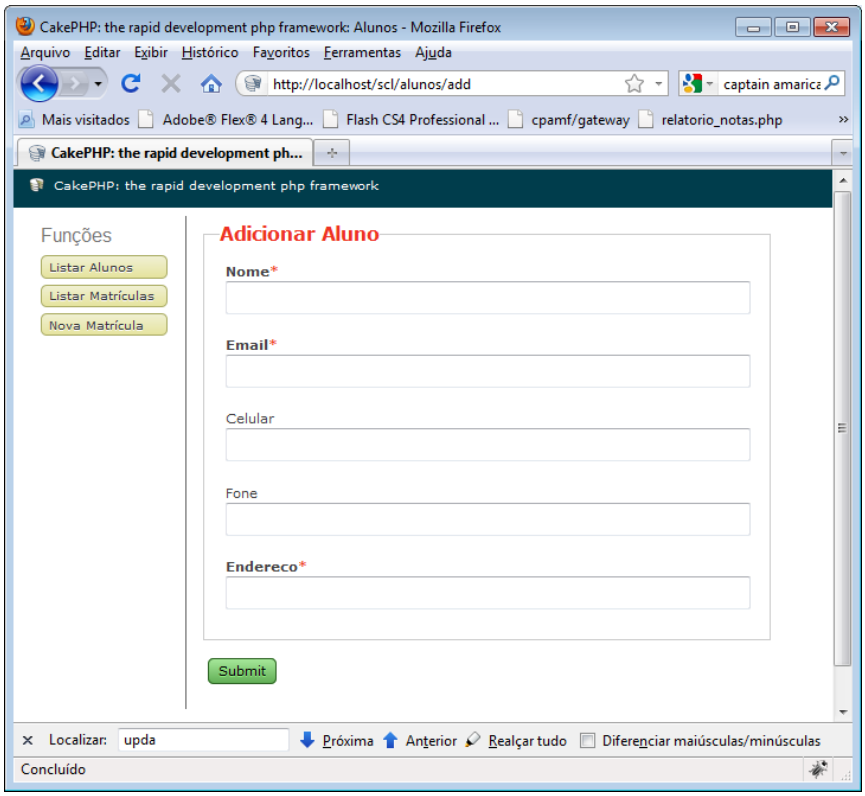

Figura 4.22: Tela Adicionar Alunos.

4.10 Gerenciamento de Turmas Ministradas

O gerenciamento de turmas contempla as seguintes funcionalidades: lançamento de nota, alterar notas, atribuir presença e atribuir presença do aluno. Com relação as turmas, temos a funcionalidade listar turmas ministradas do professor.

O gerenciamento de turmas ministradas é uma funcionalidade vinculada ao perfil professor. Esta funcionalidade foi implementada em Flex (Interface Gráfica) e em CakePHP (funções de suporte nos controles). A tela do professor é mostrada na Figura 4.25. Note as opções disponibilizadas no menu: pesquisar turmas, listar turmas e exibir aulas de hoje. Estas funcionalidades levam em consideração o professor logado. Desta forma, o professor só terá acesso as turmas ministradas por ele.

A funcionalidade pesquisa de turmas é bem simples e contempla apenas o critérios de busca por nome do curso. A tela de pesquisa é exibida na Figura 4.26. Ao preencher o

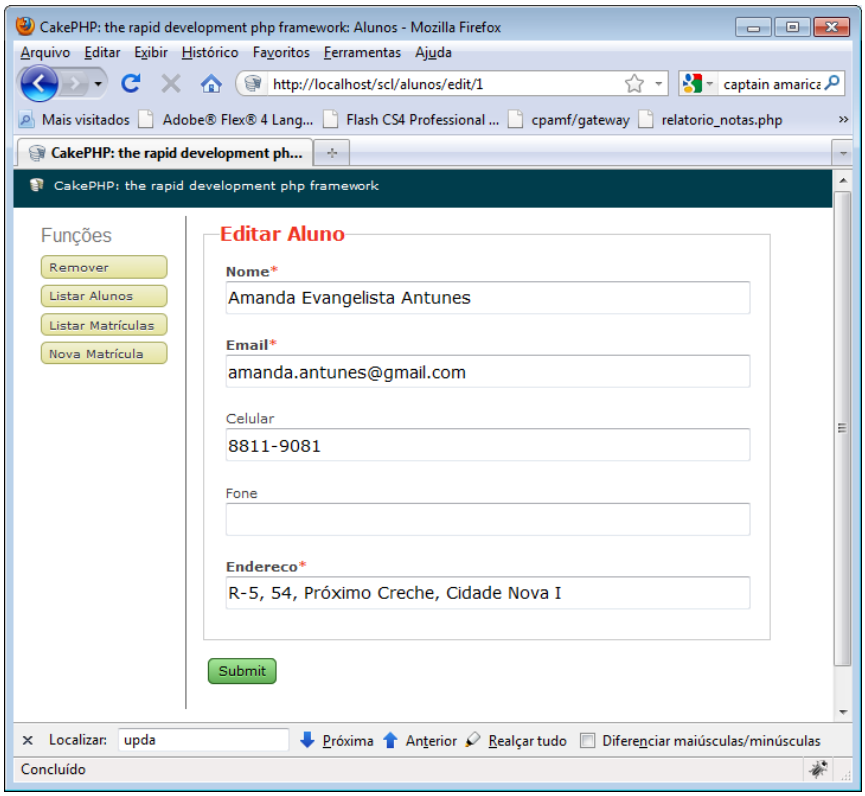

Figura 4.23: Tela Editar Alunos.

Alunos

Id	Nome	Sexo	Email	Celular	Fone	Endereco	Funções		
1	CHIEN T. RIBEIRO BEVILAQUA	F	nonummy@magna.org	9243221808	9297578055	Rua Donec 341-1104	Detalhes	Editar	Remover
2	NILDA D. SOARES DE ARAUJO	F	adipiscing.lacus.Ut@sedestNunc.ca	9211887379	9284320961	Avenida Elit Ap #574-1728	Detalhes	Editar	Remover
3	DERALDINO K. JOSE DO NASCIMENTO	M	urna@augueSed.edu	9235107190	9289379250	Av. Blandit Caixa Postal 459, 1698	Detalhes	Editar	Remover
4	DECIO W. XAVIER FERREIRA	M	pretium.et.rutrum@Integeridmagna.edu	9236900948	9290959124	Rua Fringilla. 3543	Detalhes	Editar	Remover
5	ROCILENE T. HONORIO DE MATTOS JUNIOR	F	semper.auctor@atnisi.edu	9233404117	9299345693	Travessa. Mattis 3202	Detalhes	Editar	Remover
6	SEIMIR D. CRISTIANO FELIX	M	vel.vulputate.eu@diamSeddiam.com	9232205774	9288410569	Avenida Purus Ap #120-9595	Detalhes	Editar	Remover
7	RUTH G. JOSE ANDRADE SOARES	F	gravida@nequeetnunc.org	9219605587	9297411975	Rd. Eu Caixa Postal 154, 6847	Detalhes	Editar	Remover
8	JOSEFA T. MOURA DA CUNHA	F	non@Aeneaneuismodmauris.edu	9212448806	9294310093	Rua Tellus 228-5189	Detalhes	Editar	Remover
9	SINÉSIA I. JANDER PESQUEIRA DA SILVA	M	Nunc@dolorFusce.ca	9227819823	9287517794	Travessa. Pede 922-2280	Detalhes	Editar	Remover
10	BERNARDO P. OLIMPIO GOMES	M	interdum@nisimagnased.edu	9230405435	9286800408	Avenida A 5046	Detalhes	Editar	Remover

Página 1 de 112, exibindo 10 de 1111 registro(s), iniciando no registro 1, terminando em 10

<< anterior | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | próximo >>

Figura 4.24: Tela Listar Alunos.

campo *nome* e pressionar o botão *Buscar* a pesquisa é submetida para o CakePHP e os resultados exibidos em *datagrid* (Veja Figura).

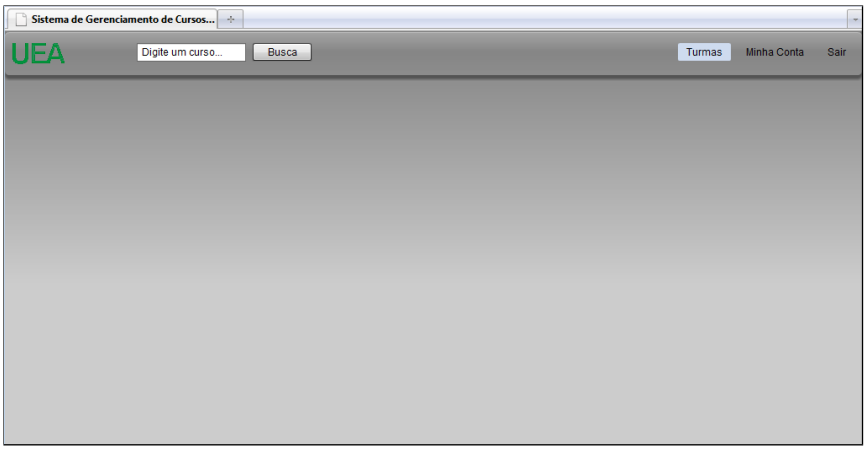

Figura 4.25: Tela Área Professor.

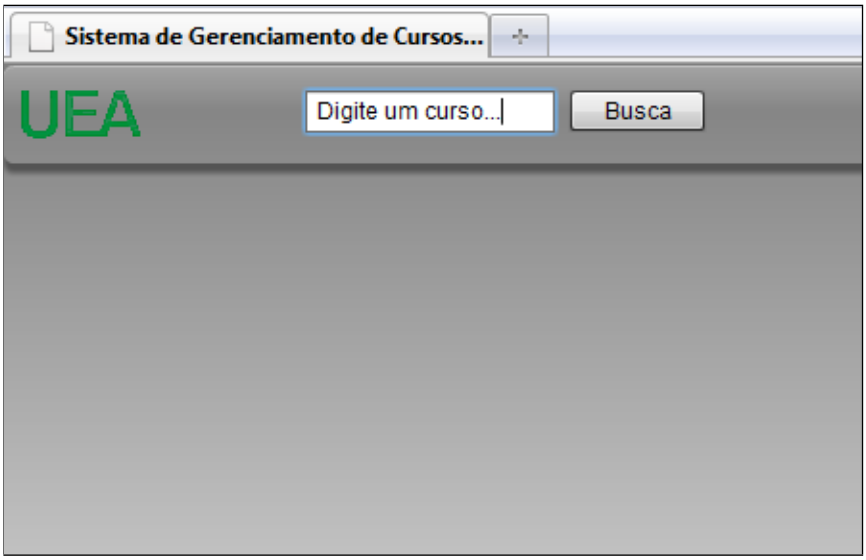

Figura 4.26: Tela Pesquisa Turma.

No *datagrid* de turmas podemos selecionar uma turma clicando na linha correspondente. Ao selecionar uma turma, os botões *Notas* e *Frequências*.

Na Figura 4.28 está ilustrada a tela gerenciar turmas. Note que os campos que descrevem a turma são apresentados ao usuário no modo somente leitura. Os únicos campos editáveis são Nota e Frequências. Para editá-los, basta clicar na célula correspondente.

O campo Frequência abre uma outra janela ao ser editado. A janela de edição de frequências consiste em um *datagrid* contendo os alunos e os dias de aula. Cada campo do *datagrid* correspondente a um dia aula exibe um *checkbox* para que o usuário atribua uma

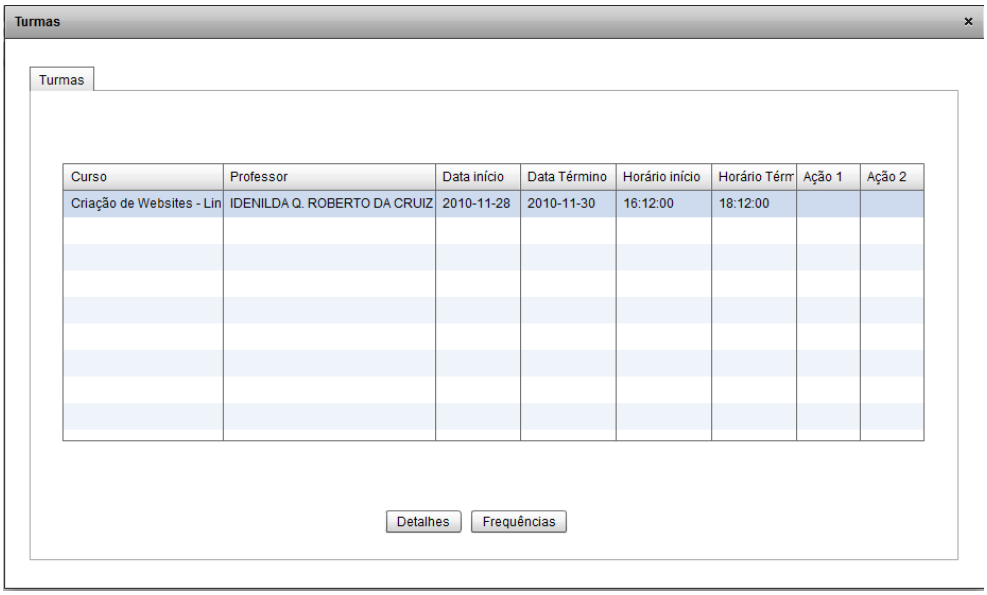

Figura 4.27: Tela de Resultados de Pesquisa de Turma.

presença (*checkbox* marcado) ou falta (*checkbox* desmarcado).

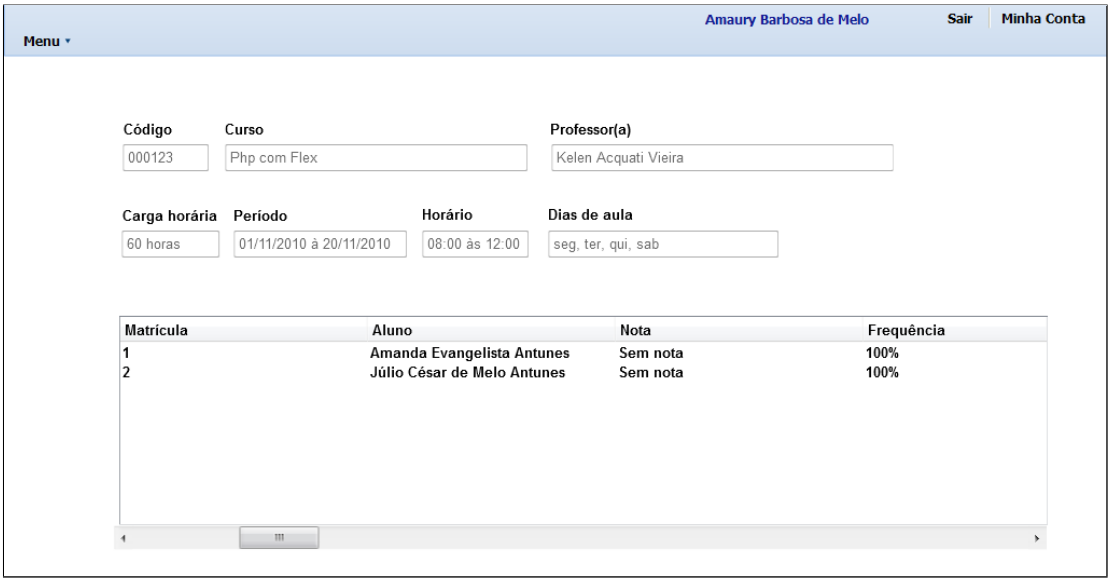

Figura 4.28: Tela Gerenciar Turma.

4.11 Relatórios

Esta funcionalidade gera um relatório, em excel, das notas dos alunos de uma determinada turma. A Figura exibe o relatório gerado pelo sistema.

Turma	1 - PHP		
Professor	Amaury Barbosa de Melo		
Período	2010-11-03	2010-11-04	
Horário	08:00 as 10:00	Qua, Qui	

Matrícula	Aluno	Frequência	Nota	Situação
00001	Amanda Evangelista Antunes	100%	Sem Nota	
00002	Júlio César de Melo Antunes	100%	Sem Nota	

Figura 4.29: Relatório de Notas da Turma.

4.12 Configuração do CakePHP para o SCL

Entre as vantagens dos *frameworks Web* destaca-se que eles trazem em si muitas funcionalidades comuns prontas. Desta forma o desenvolvedor não precisa reescrever os mesmos conjuntos de funcionalidades para toda a aplicação que vier construir. Nesta seção, exploraremos como nos beneficiámos destas vantages na implementação deste estudo de caso. Comentaremos também sobre as configurações que realizamos para concretizar o SCL.

4.12.1 Geração do CRUD básico do SCL

Uma vantagem de utilização do CakePHP em relação a outros *frameworks* é a geração automática do CRUD do sistema. Isso significa que tendo terminado a fase de análise e de

projeto de banco de dados, podemos testar a modelagem através do CRUD MVC gerado através do CakePHP. Em seguindo pode-se personalizar o CRUD, criando novas classes para serem adicionadas no projeto inicial, mudando o *layout* das páginas HTML, etc.

Para realizar esta tarefa, o CakePHP dispõe do utilitário *bake*. O *bake* é um utilitário com interface linha de comando. Portanto é necessário abrir o prompt do Windows ou o shell do linux para gerar os CRUDs da aplicação. Antes de gerá-los é necessário uma configuração de banco de dados válida, no CakePHP (Veja a seção 2.3.1 na página 14). Tendo configurado o CakePHP conexão com o banco de dados, basta invocar o *bake* para abrir uma sessão na linha de comando. A iteração é bem simples. Basicamente, consiste em um menu de opções. Cada opção corresponde a uma tabela no banco de dados. A partir da escolha de uma opção (tabela), o *bake* dar opções para a criação das três classes da tríade MVC: Modelo, Visualização e Controle. Há algumas opções extras como gerar classes para testes unitários com o *SimpleTest* mas não faremos uso destas opções no presente trabalho. Uma sessão *bake* está ilustrada na Figura 4.30.

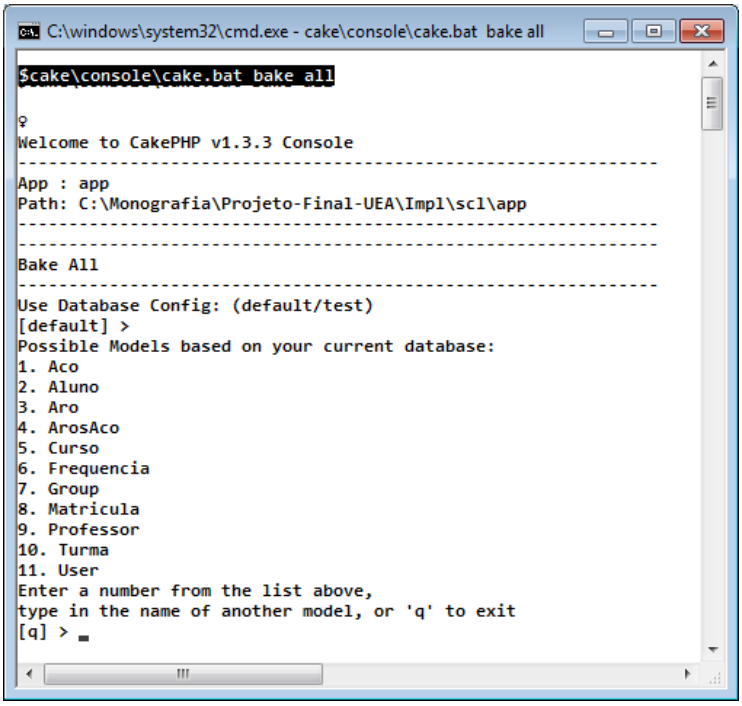

Figura 4.30: Menu de opções ao executar o *bake* com a opção *all*.

4.12.2 Melhorando a Interface com o Usuário

O gerador de código não é perfeito e eventualmente precisamos fazer alguns ajustes nas telas geradas. Um exemplo comum é quando visualizamos um objeto na tela que contenha campos representando chaves-estrangeiras. O gerador de código antes de gerar tal campo, busca por atributos descritivos, quando não os encontra, o padrão é exibir a chave-estrangeira. Os campos *name* (nome), *description* (descrição) , *password* (senha) são exemplos de típicos que o CakePHP considera como descritivo. Por exemplo, A tela alterar turma (Figura 4.20, possui dois campos de chaves-estrangeira, o campo curso e professor. Para que o nome do curso e do professor fossem exibidos, em vez das chaves-primárias, foi necessário incluir o atributo *$displayField* nos modelos Professor e Curso. Na Listagem 4.1 vemos que o dito campo informa a camada de Visualização qual o campo do objeto deve ser utilizado nesses casos.

```
<?php
class Curso extends AppModel {
        var $name = 'Curso';

        var $displayField = 'nome';
```

Listagem 4.1: Campo preferencial para exibição na camada de visualização

Ainda considerando a tela alterar turma, observe os campos relativos aos dias da semana. Estes campos são armazenados no banco *Mysql* como números inteiros. O CakePHP faz um bom trabalho de associar os campos de um objeto ao *widget* HTML mais adequado para exibí-los em tela. Entretanto, isso nem sempre é satisfatório. No presente caso, o CakePHP associou ao campo um *text input* (caixa de texto) o que tornava a entrada dos campos dias da semana nada intuitivos (ver Figura 4.31). Além disso, o formato dos campos relativos a data e horários não fazem sentido em nossa língua. A Figura 4.32 mostra a versão melhorada desta tela.

Alterar Turma

Curso*
PHP

Professor*
Amaury Barbosa de Melo

Data Inicio*
November - 3 - 2010

Data Termino*
November - 4 - 2010

Hora Inicio*
8 : 00 am

Hora Termino*
10 : 00 am

Domingo*
0

Segunda*
0

Figura 4.31: Tela Alterar Turma sem Melhorias

Add Turma

Curso*
Criação de Websites - Linguagem XHTML

Professor*
IDENILDA Q. ROBERTO DA CRUIZ

Data de início*
30 - November - 2010

Data de término*
30 - November - 2010

Horário de início*
5 : 44

Horário de término *
5 : 44

Domingo*

Segunda*

Figura 4.32: Tela Alterar Turma com Melhorias

Capítulo 5

Conclusão e Trabalhos Futuros

A necessidade de se ter um produto, ou seja, um sistema de software bem elaborado, eficaz e eficiente de forma rápida torna os frameworks CakePHP e Flex cada vez mais procurados por desenvolvedores de aplicações Web. E com isso surge, também, outra necessidade a de se ter um documento capaz de mostrar e resolver as peculiaridades de cada framework. Observe que ao final deste trabalho pudemos obter telas ou interfaces agradáveis e amigáveis de forma simples e rápida. Interface capaz de cadastrar, de alterar e de remover dados sem nenhum esforço com implementação de linhas de códigos, mas apenas com configurações e com uma pitada de organização estrutural fora eficientes para se ter um sistema de qualidade. Sistema o qual é capaz de validar, proteger e manipular de forma segura os dados persistentes em um banco de dados. E conforme o sistema iria tomando forma em paralelo surgia uma documentação de suma importância para programadores com pouca experiência no ramo de uso de frameworks. Tal documentação revela os pontos críticos usados para a realização da internacionalização e localização, como I18N e L10N, ou seja, a tradução automática da língua nativa dos frameworks (inglês) para o português e outras informações a respeito da persistência de dados, como CRUD . O principal ponto desde trabalho é a importância da organização estrutural de sistema e mostrar uma metodologia bastante eficiente na organização estrutural de sistema, a FDD. Outro ponto importante é o uso de padrões de projetos, o qual facilita e agiliza a concepção do sistema. Neste, enfatiza bem a importância e a magia que se tem ao usar o padrão MVC juntamente com a

tecnologia RIA, cujos resultados são interfaces mais amigáveis (muito próximas das interfaces usadas em Desktop) e previamente preparadas para captação e persistência de dados feita automaticamente. Deixamos como trabalhos futuros a autenticação criptografada para entrada de usuários, o aperfeiçoamento do Sistema de Cursos Livres. Por exemplo, os cadastros dos professores, alunos e usuários poderiam obter mais informações a respeito deles e a inclusão do módulo para empresas interessadas na capacitação de seus funcionários através dos cursos livres. Outro módulo interessante seria a integração do Sistema de Cursos Livres ao Sistema Financeiro da EST e por último implantá-lo via internet, para que os alunos e empresas possam acessá-lo remotamente.

Referências Bibliográficas

COAD, P.; LEFEBVRE, E.; DELUCA, E. *Java Modeling in Color with UML: Enterprise Components and Process*. Upper Saddle River, NJ: Prentice Hall, 1999. ISBN 978-0-13-011510-2.

DOYLE, M. *Beginning PHP 5.3*. Birmingham, UK, UK: Wrox Press Ltd., 2009. ISBN 0470413964, 9780470413968.

FOUNDATION, I. C. S. *Componentes: Extensões de Controles*. cakephp.org, 2010. Disponível em: <http://book.cakephp.org/pt/view/894/Controller-Extensions-Components>.

FOUNDATION, I. C. S. *Entendendo o Model View Controller*. Cake Software Foundation, Inc., 2010. Disponível em: <http://book.cakephp.org/pt/view/10/Understanding-Model-View-Controller>.

GAMMA, E. et al. *Design patterns: elements of reusable object-oriented software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0-201-63361-2.

GOLDING, D. *Beginning CakePHP: From Novice to Professional*. 1. ed. Berkely, CA, USA: Apress, 2008. ISBN 1430209771, 9781430209775.

HARVEY, D. P. J. D. *AJAX, RIA e Desnvolvimento Web para Programadores*. 1ª ed.. ed. [S.l.]: Pearson Education, 2008. ISBN 0131587382, 9780131587380.

MOORE, D.; BENSON, E.; BUDD, R. *Professional Rich Internet Applications: AJAX and Beyond*. Birmingham, UK, UK: Wrox Press Ltd., 2007. ISBN 0470082801.

PALMER, S. *Modelling in Colour: Model Archetypes:Using the Domain Neutral Component (DNC)*. knol.google.com, 2009. Disponível em: <http://knol.google.com/k/stephen-palmer/modelling-in-colour-model-archetypes/3e0t9wv30hso7/8>.

PALMER, S. *Uma Visão Geral do FDD*. www.step-10.com, 2010. Disponível em: <http://www.step-10.com/SoftwareProcess/FeatureDrivenDevelopment/introduction.html>.

SOMMERVILLE, I. *Engenharia de Software*. oitava edição. [S.l.]: São Paulo: Pearson., 2010.

WIKIPEDIA, A. e. l. *Expressão Regular*. Wikipedia, 2010. Disponível em: <http://pt.wikipedia.org/wiki/Expressão_regular>.