Aplicação do Framework CakePHP para o desenvolvimento de WebSite

Lucas Luan Pontarolo[1], Elena Mariele Bini[2]

Instituto Superior de Educação – Faculdade Guairacá - Tecnologia Em Análise E Desenvolvimento De Sistemas - Guarapuava – PR – Brasil

lucaspontarolo@gmail.com, elenamarielebini@gmail.com

Abstract. *This article describes the development of a personal information control system, with the primary requirement dynamic expansion of registration elements. This work is aimed at professionals who find difficult to manage a variety of information that change over the time, it had a focus on reducing costs and simplifying the management and storage of information. The system was encoded with the PHP language on the CakePHP framework, along with the Apache web server. For the storage of data was used DBMS PostgreSQL 9.3.*

Resumo. *Este artigo descreve o desenvolvimento de um sistema de controle de informações pessoais, tendo como principal requisito a expansão dinâmica de elementos de cadastro. Destinado a profissionais que encontram dificuldades em gerir uma variedade de informações que se modificam no decorrer do tempo, teve foco na redução de custos e na simplificação do gerenciamento e armazenamento de informações. O sistema foi codificado com a linguagem PHP, sobre o framework CakePHP, juntamente com o servidor web Apache. Para o armazenamento de dados foi utilizado o SGBD PostgreSQL 9.3.*

Introdução

Este trabalho objetiva apresentar o processo de desenvolvimento de uma aplicação *web* para controle de informações pessoais, com expansão dinâmica de elementos de cadastro e possibilidade de mudança da abrangência ou foco das informações. A aplicação destina-se a profissionais que necessitam reunir informações sobre pessoas (personalidades) e que encontram dificuldades no uso de sistemas tradicionais de cadastro pessoal, uma vez que a abrangência dos elementos de informação destes sistemas geralmente é limitada, por vezes, fixos.

A aplicação foi desenvolvida utilizando a linguagem de programação PHP, em sua versão 5.4, juntamente com o framework de aplicação CakePHP, em sua versão 1.3.17 com persistência de dados no SGBD PostgreSQL 9.3.

O software desenvolvido tem por objetivo auxiliar os profissionais através de uma arquitetura expansiva e contribuir diretamente com estudos de estratégias que auxiliarão na dinamização de aplicações. Visa também expandir indefinidamente os elementos de informação em um sistema de cadastro, melhorando a interação do usuário com o sistema, tornando os softwares mais eficientes e reduzindo custos com manutenções.

Fundamentação Teórica

Softwares são usados como combustíveis pelos negócios modernos, pois realizam inúmeras tarefas em uma empresa, aperfeiçoando, assim, procedimentos que levariam horas ou até dias de trabalho manual. Rainer JR. e Cegielski (2011) afirmam que um software de aplicação proprietário é desenvolvido única e exclusivamente para lidar com necessidades específicas ou exclusivas de uma empresa. Este tipo de aplicação deve ser encomendado diretamente de um fornecedor de software, que realizará uma série de procedimentos a fim de coletar o máximo possível de informações, para posterior desenvolvimento do software. Diferentemente de um software de aplicação comercial, comumente chamado de software "de prateleira", que é desenvolvido e vendido da mesma maneira para diversas empresas, podendo somente ser alterado com a compra de pacotes de atualização em separado.

Rainer JR. e Cegielski (2011) afirmam ainda que investimentos em um software de aplicação proprietário são muito importantes por trazer confiabilidade, robustez, funcionalidade e segurança aos dados e processos da mesma, já que são desenvolvidos de forma a se adequar à maneira de trabalho da empresa para atender ao máximo suas necessidades.

Os softwares, então, auxiliam em muitas tarefas, proporcionando os melhores resultados para as diversas áreas onde forem aplicados. Sistemas *web*, segundo Sales (2009), estão sendo utilizados em grande escala pela sua facilidade de acesso e sua independência de plataforma. Podem ser abrigados em servidores e são facilmente acessados de maneira remota e, por consequência, aperfeiçoam assistências, evitando deslocamentos desnecessários.

O desenvolvimento *web*, segundo Eis e Ferreira (2012), inclui todo o desenvolvimento de conteúdo dinâmico que interage diretamente com o servidor (*Server-side*) ou navegador *web* (*Client-side*). Ainda autores descrevem um sistema *Client-side* subdividindo-o em três camadas principais: informação, formatação e comportamento.

A camada de informação, segundo Eis e Ferreira (2012), é a mais importante, pois é responsável por estruturar toda a informação constituída pelo HTML. Segundo Silva (2008), o HTML é uma abreviação de *Hypertext Markup Language*, que traduzindo para o português significa Linguagem de Marcação de Hipertexto, responsável única e exclusivamente pela marcação semântica de textos, construídos de elementos e atributos. O site oficial da W3C (2013) afirma que essa linguagem deve ser somente utilizada para marcação semântica dos elementos, ou seja, usar os elementos da linguagem em conformidade com o seu significado.

A camada de formatação é responsável por controlar o visual da informação exibida pelo documento gerado através da CSS (*Cascading Style Sheets*). Eis e Ferreira (2012) afirmam que a CSS deve ser única e exclusivamente responsável pela formatação e estilização da informação, para que ela seja consumida em qualquer meio de acesso, de forma simples e da melhor maneira possível.

A camada de comportamento é composta pelo JavaScript. Silva (2010) assegura que o JavaScript é responsável por manipular, personalizar e automatizar as

funcionalidades em um sistema, sendo assim responsável por trazer interatividade à página, melhorando ainda mais a experiência do usuário.

Ainda segundo Silva (2010), JavaScript é uma linguagem de programação *Client-side,* que utiliza o próprio browser do usuário como interpretador. Diferente do HTML e CSS que apenas estruturam e estilizam os elementos, no JavaScript é possível empregar lógica de programação, possibilitando assim a construção de algoritmos para automação de tarefas. Niederauer (2007) comenta que o JavaScript é uma linguagem de programação com baixa tipificação, ou seja, suas variáveis não precisam ser definidas no escopo do script. Niederauer (2007) também explica que o JavaScript é uma linguagem Orientada a Objetos, o que traz ainda mais importância, pois permite o uso de uma grande quantidade de Objetos nativos, possibilitando a construção de grandes programas com uma variedade de funções e estruturas de dados complexas.

Eis e Ferreira (2012), bem como Silva (2008), afirmam que documentos HTML, CSS e JavasScript, bem escritos, estão em conformidade com as *web standards* especificadas pela W3C (*World Wide Web Consortium*). A W3C (2013) define *web standards* como um conjunto de normas e diretrizes destinadas a orientar fabricantes e desenvolvedores sobre a padronização no uso das tecnologias *web*, com a finalidade de tornar a *web* acessível a todos, independente do seu meio de acesso ou de suas necessidades especiais.

O site oficial W3C (2013) descreve-se como sendo um consórcio internacional, no qual organizações filiadas, uma equipe em tempo integral e o público trabalham juntos para desenvolver padrões para as tecnologias *web*. Queiroz (2013) defende que códigos escritos de acordo as *web standards* especificadas pela W3C permitem rapidez de interpretação para todos os agentes do usuário, sejam em desktops ou dispositivos móveis. Assim, conexões lentas, discadas ou divididas por grande número de usuários em rede, bem como ampliadores de tela que navegam associados a leitores, através de códigos bem escritos, tornam o tempo de download das páginas mais adequado.

Visando utilizar-se da melhor forma das tecnologias citadas e também proporcionar ao usuário uma melhor impressão sobre todos os componentes do sistema, optou-se pelo uso das CSS disponibilizadas pelo Joomla, de licença *GNU/GPL2* e com distribuição gratuita. O pacote CSS do Joomla é desenvolvido por um grupo de profissionais que, além de construir seus códigos em conformidade às normas estabelecidas pela W3C, pensam na experiência do usuário. Assim, padronizam todas as interfaces da aplicação, garantindo uma utilização mais agradável.

Como o sistema apresentado se propõe a manipular e persistir informações dinamicamente, se faz necessária a utilização de uma linguagem de programação, responsável por gerenciar toda a lógica e o acesso aos dados do sistema. A linguagem de programação utilizada no desenvolvimento do software foi o PHP (*Hypertext Preprocessor*) em sua versão 5.4.

Nascimento e Melo (2008) afirmam que o PHP é a linguagem de scripts mais utilizada para os ambientes de sistemas para a internet, possuindo uma vasta gama de comunidades que a apoiam e a mantêm. O site oficial do PHP (2013) descreve a linguagem como sendo muito flexível, por ter a possibilidade de ser utilizada em conjunto com o HTML, possibilitando a geração de conteúdo dinâmico com muita

facilidade. Pode ser utilizada na maioria dos sistemas operacionais, caracterizando-se com uma linguagem multiplataforma.

De acordo com Welling e Thomson (2005), o PHP é uma linguagem de programação de código aberto, executada no lado servidor (*Server-side*). Xavier (2008) define a linguagem de programação *Server-side* como uma linguagem com a capacidade de geração de conteúdo em tempo de execução, além das aplicações não poderem ser copiadas por outras pessoas, pois todos os processos, rotinas e funções serão executados no servidor e o usuário receberá apenas os resultados.

O PHP em sua versão 5, segundo Xavier (2008), teve o tratamento de objetos totalmente reescrito, assim garantindo um melhor desempenho diante a implementação da Programação Orientada a Objetos (POO). Xavier (2008) descreve a POO como uma maneira de programar, de modo a se aproximar ao máximo da realidade.

Um objeto dentro da POO, Segundo Lima (2012), é tudo aquilo que pode ser compreendido pelo conhecimento, ser manipulado, manufaturável, perceptivo por qualquer sentido, pensado ou representado. Santos (2003), afirma que a orientação a objetos se faz importante para qualquer aplicação onde seja necessário representar conjuntos de dados dependentes ou interligados entre si, bem como também adequa a qualquer aplicação que seja necessário aplicar um conjunto de rotinas a um conjunto de dados. Para se utilizar da orientação a objetos, se faz necessário o conhecimento sobre classes e objetos. De acordo com Dall'Oglio (2009), uma classe é uma estrutura utilizada para descrever objetos por meio de métodos e atributos e objetos são estruturas originadas com base em informações de uma classe.

Para se utilizar do PHP, que é uma linguagem interpretada, há necessidade da instalação de um interpretador antes de executar qualquer script. Escolheu-se o servidor *web* Apache.

O Apache, segundo Xavier (2008), é um servidor *web* de código aberto, extremamente configurável, robusto e de alto desempenho. Melo e Nascimento (2008) relatam que o Apache é excelente por possuir uma equipe de desenvolvedores voluntários, que visam criar um servidor com uma vasta gama de configurações e ferramentas que auxiliam toda manutenção. Entre estas, destacam-se: registro de logs, alto nível de segurança, fina granularidade de configuração, implementação de hosts virtuais e suporte a SSL (*Secure Sockets Layer*). Xavier (2008) também afirma que o servidor Apache é estável e seguro, por ser desenvolvido há muito tempo, além de ser mais usado que todos os outros servidores *web* do mundo juntos e ser suportado por servidores *web* do mundo todo.

No mercado de desenvolvimento para a *web*, as aplicações precisam ser desenvolvidas em um curto espaço de tempo. Belem (2013) afirma que, para alcançar velocidade e produtividade no desenvolvimento, podem-se utilizar frameworks que facilitam a reutilização e a geração de código.

Um framework, segundo Soares (2009), é um grande conjunto de classes, métodos e rotinas desenvolvidas para suportar a construção de aplicações robustas e complexas. A definição mais aceita para um framework orientado a objetos, segundo Soares (2009) *apud* Wirfs-brock (1990), é servir como base para a implementação de

um sistema de aplicação, composto por várias classes abstratas e concretas, promovendo um modelo de interação e colaboração entre as instâncias.

O desenvolvimento de aplicações utilizando um framework, aproveitando-se ao máximo de sua estrutura e utilização de seus componentes, segundo Soares (2009), produz aplicações mais robustas e com uma excelente qualidade. Além disso, o tempo de desenvolvimento torna-se muito menor, diminuindo consideravelmente os custos do projeto. Ainda Soares (2009) *apud* Wirfs-brock (1990), descreve um framework como um esqueleto para implementação de uma aplicação ou de um subsistema de aplicação. É composto de classes abstratas e concretas e provê um modelo de interação ou colaboração entre as instâncias de classes definidas, seguindo padrões de projetos específicos para cada tipo de aplicação.

Dessa forma, optou-se pela utilização do framework de aplicação CakePHP em sua versão 1.3.17 para desenvolvimento do software apresentado neste artigo. Segundo seu site oficial, o CakePHP é uma distribuição gratuita, de código aberto, desenvolvido em PHP, baseado nos conceitos do framework *Ruby on Rails*. O objetivo principal do framework é permitir um desenvolvimento robusto de aplicações para qualquer nível de programadores e deixando de lado a monotonia tradicional da implementação.

O site oficial do CakePHP (2013) ressalta que o mesmo possui uma equipe de desenvolvedores ativa e uma grande comunidade agregando e potencializando o crescimento do framework. Belem (2013) assegura que a maior vantagem no uso dos padrões estabelecidos pelos frameworks, é permitir a redução do custo do desenvolvimento, bem como reduzir muito a escrita de código, facilitando a sua manutenção.

O CakePHP é distribuído sob a licença MIT (*Massachusetts Institute of Technology*). MIT é uma licença de programas de computadores usada em softwares de código livre, criada pelo *Massachusetts Institute of Technology*. Leite (2009) afirma que a licença MIT permite a utilização, cópia, modificação, distribuição e a venda do software, com a condição que a nota de copyright seja conservada em todas as cópias.

Para ser entendido e utilizar de forma correta o framework, recomenda-se entender suas convenções básicas de organização, desde nomes de arquivos até nomes de tabelas de bancos de dados. O site oficial do CakePHP (2013) menciona que, apesar do tempo que se pode levar para aprender as convenções, o desenvolvedor ganhará muito tempo a longo prazo, pois seguindo as convenções, ganhará velocidade em seu desenvolvimento, já que todos os módulos estarão bem organizados e estruturados. As convenções também contribuem para o desenvolvimento de sistemas mais uniformes, permitindo que outros desenvolvedores entrem no projeto e comecem a trabalhar mais rapidamente.

Um dos padrões de projeto que o CakePHP emprega é o MVC (*Model, View, Controller*). Ele propõe uma divisão de camadas, tornando a lógica de negócio separada da interface de apresentação para o usuário, separando-a também do fluxo de informação. Isto facilita o desenvolvimento e a manutenção do código do aplicativo, pois poupa esforços e tempo de desenvolvimento quando manutenções adaptativas se fizerem necessárias.

A abordagem MVC segundo Soares (2009), consiste em separar modelagem, apresentação e fluxo dos dados em camadas distintas. Este padrão de projeto é considerado uma boa prática, uma vez que modulariza o processo de desenvolvimento, promove a reutilização de código e permite que várias interfaces sejam aplicadas:

- O *model* representa os dados;
- A *view* representa a visualização dos dados;
- O *controller* manipula e roteia as requisições dos usuários.

A Figura 1 abaixo representa o Design Pattern MVC empregado pelo framework:

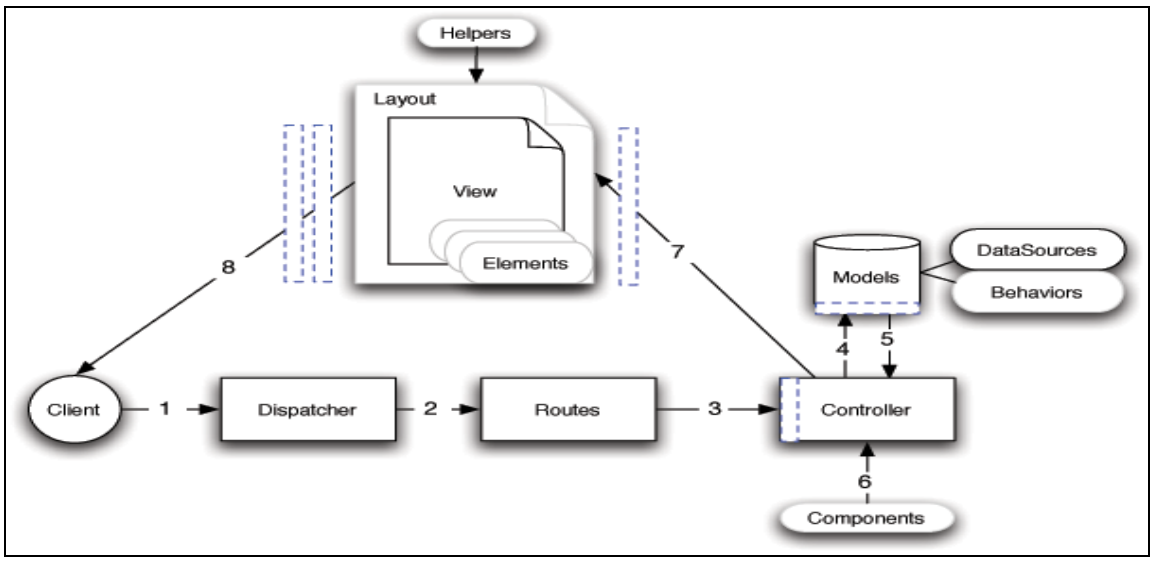


Figura 1: Uma requisição básica no MVC (Fonte: Site oficial CakePHP).

A explicação deste modelo de requisição, segundo o site oficial do CakePHP (2013), é a seguinte: um ciclo de requisição do CakePHP começa quando o usuário solicita uma página ou recurso da aplicação. Esta solicitação é primeiramente processada por um *Dispatcher* (expedidor), que irá selecionar o objeto *Controller* (controlador) correto para lidar com a solicitação requisitada.

Assim que a solicitação do usuário chega ao *Controller*, este irá se comunicar como a camada *Model* (modelo) para processar qualquer operação de busca ou armazenamento de dados que for necessário. Após o término desta comunicação, o *Controller* envia para o objeto *View* (visão) correto, a tarefa de gerar uma saída resultante dos dados fornecidos pelo *Model*. Finalmente, quando a saída é gerada, ela é imediatamente enviada para o usuário.

Para Milani (2008), qualquer sistema de gerenciamento de dados precisa manter as informações com as quais trabalha, para isso, se fez necessário a utilização de um banco de dados.

Elmasri e Navathe (2011) afirmam que os bancos de dados desempenham um papel fundamental em quase todas as áreas que os softwares são empregados, por armazenarem dados de maneira a permitir consultas futuras, geração de relatórios ou possíveis alterações nas informações. Milani (2008) afirma que, para utilização de um banco de dados, se faz necessário um Sistema Gerenciador de Banco de Dados (SGBD).

Elmasri e Navathe (2011, p.3) descrevem um SGBD como “uma coleção de programas que permite aos usuários criar e manter os bancos de dados, permitindo assim a definição, construção, manipulação e compartilhamento dos bancos de dados dentre diversas aplicações”. Assim, para a persistência dos dados do software apresentado neste artigo, fez se uso do SGBD PostgreSQL.

O PostgreSQL, segundo Milani (2008), é um excelente banco de dados por não limitar o tamanho dos seus bancos, tendo como uma de suas especialidades a estabilidade. Foi projetado para executar no método 24/7, sendo muito utilizado em servidores na internet, sejam quaisquer soluções em informática. Milani (2008) afirma que, em comparação com outros bancos de dados, o PostgreSQL se destaca pela eficiência com que realiza suas operações e principalmente pela sua licença, que permite se integrar em aplicações comerciais sem restrições e sem manter o código da sua aplicação aberto ou, ao menos, pré-requisitos.

Para a construção das interfaces de interação do sistema, fez se importante o estudo das técnicas em IHC para construção de uma interface de interação mais confortável e de fácil compreensão. A IHC (Interação Humano-Computador), segundo Barbosa e Silva (2009, p.10), “é uma disciplina interessada no projeto, implementação e avaliação de sistemas computacionais interativos para uso humano, juntamente com os fenômenos relacionados a este uso”.

Porém, não basta aplicar as técnicas de IHC sem pensar na qualidade. Segundo Netto (2010), a qualidade em IHC, especificamente a interação e a interface, devem ser adequadas para que os usuários possam aproveitar ao máximo o apoio computacional oferecido pelos sistemas. Os critérios de qualidade de usos essenciais, descritos e apresentados por Barbosa e Silva (2009), são: usabilidade, experiência do usuário, acessibilidade e comunicabilidade.

Para expressar visualmente todos os processos do sistema através de diagramas, foi utilizada a linguagem UML (*Unified Modeling Language*). Lima (2012) descreve a UML como uma linguagem gráfica padrão para a elaboração da estrutura de projetos complexos de software. A UML pode ser empregada para visualizar, especificar, construir e documentar os artefatos de sistemas de software, sem deixar de expressar-se claramente para os seus usuários e outros desenvolvedores, pois utiliza uma notação padrão. Bezerra (2007) complementa ao apresentar a UML como uma linguagem de modelagem visual, um conjunto de notações e semântica correspondente para representar visualmente uma ou mais perspectivas de um determinado sistema.

Bezerra (2007) também aponta a utilização da UML como uma tarefa complexa. O processo de programação que utiliza diagramas envolve a criação de diversos documentos. Estes documentos são apresentados de formas textuais e gráficas, denominados artefatos de software, ou simplesmente artefatos. Tudo isso irá compor a visão do sistema. Existem diversos diagramas da UML, porém, no sistema aqui proposto, serão utilizado três deles: diagrama de caso de uso, diagrama de classe e diagrama de sequência.

A engenharia de software, segundo Sommerville (2011), é um elemento indispensável para o desenvolvimento profissional de software. Ela pretende incluir técnicas que apoiam a especificação, documentação, testes, projeto e evolução de

softwares, que normalmente não são relevantes para o desenvolvimento de software pessoal. A engenharia está relacionada com a obtenção de resultados de qualidade, requeridos dentro do cronograma e do orçamento do projeto.

Para que o processo de desenvolvimento se inicie, Sommerville (2011) afirma que se faz necessário definir um modelo de ciclo de vida. Assim, o modelo adotado para este projeto foi o de prototipação. Segundo Laudon e Laudon (2007), este modelo consiste em desenvolver um sistema experimental (protótipo) rapidamente, para submetê-lo à avaliação do usuário. Sommerville (2011) afirma que protótipos do sistema permitem que os usuários vejam quão bem o sistema dá suporte a seu trabalho. Eles podem apresentar novas ideias para o desenvolvimento e encontrar pontos fortes e fracos do software, propondo novos requisitos ao sistema, ou até mesmo apontar falhas na lógica da aplicação.

Laudon e Laudon (2007) relatam que o modelo de prototipação envolve a participação intensa do usuário durante o processo de desenvolvimento, isto aumentará a probabilidade de que todos os requisitos sejam atendidos, inclusive as expectativas do usuário. Laudon e Laudon (2007) apresentam um modelo de quatro etapas para o processo de prototipagem. A Figura 2 abaixo representa o modelo apresentado.

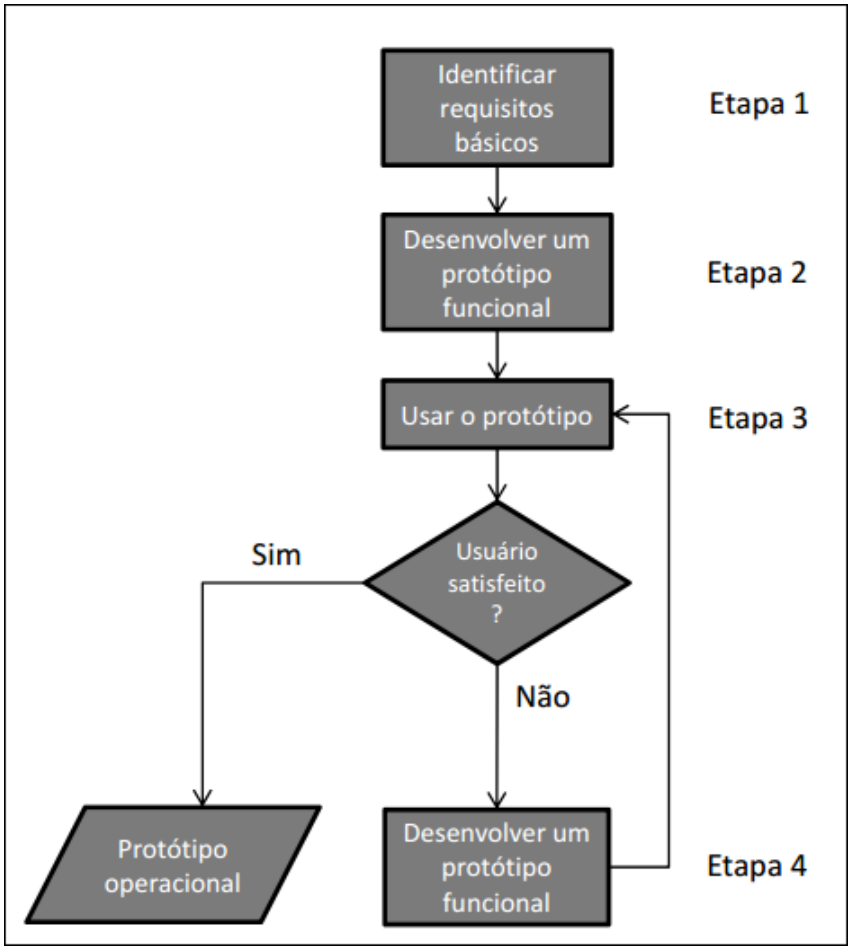


Figura 2: Processo de prototipagem (Fonte: Laudon e Laudon (2007), p. 356).

Um importante conceito aplicado no desenvolvimento do sistema apresentado neste artigo é a usabilidade. Segundo Cybis, Betiol e Faust (2010), a usabilidade é a

qualidade que caracteriza o uso dos programas e aplicações. O uso das normas de usabilidade se faz importante por proporcionar interações satisfatórias para todos os tipos de usuário. Koscianski e Soares (2006) afirmam que, mesmo em softwares que realizam funções complicadas, com grande volume de informações, que exigem maior atenção do usuário, o projeto de tais interfaces merece atenção especial para que a interface em si, lógica do software, concorde com a lógica do usuário.

Diante destas afirmações, o desenvolvimento do sistema também teve como fator importante a eficiência das ações do usuário sobre a interface. Embasado neste contexto, a aplicação da interface teve como principal foco a simplicidade, a fácil compreensão e cumprimento dos objetivos para proporcionar uma experiência prazerosa do usuário.

A Garantia de Qualidade de Software tem sido introduzida ao longo de todo o processo de desenvolvimento. Koscianski e Soares (2006), assim como Paula Filho (2009), afirmam que critérios devem ser estabelecidos para que se possa julgar a qualidade de um produto. Paula Filho (2009) afirma que o critério de qualidade mais importante para o desenvolvimento de software é seu grau de conformidade com todos os requisitos especificados. Ainda segundo Paula Filho (2009), um importante conceito de qualidade são as atitudes preventivas realizadas em relação aos problemas de qualidade que poderiam surgir durante o desenvolvimento do produto.

Desenvolvimento

A etapa inicial para o desenvolvimento do sistema foi a coleta de informações para posterior definição dos requisitos. Essa coleta inicial se deu por meio de uma conversa informal. Notou-se que o principal empecilho no uso de sistemas tradicionais de cadastro pessoal era a abrangência limitada dos elementos de cadastro, por vezes fixos. Verificou-se a possibilidade de, ao longo do tempo, haver necessidade de mudança da abrangência ou foco das informações.

Diante desta necessidade, foi estudada e proposta uma aplicação *web* para controle de informações pessoais, visando facilitar a manipulação das informações cadastradas e a expansão de elementos de cadastro. Pensou-se ainda em dispor de um controle de usuários e empresas para uma segmentação lógica (de dados) do sistema.

Com a coleta de informação, foram definidos os principais requisitos do sistema - apresentados no apêndice A deste artigo. Em seguida, iniciou-se a fase de produção de protótipos do sistema e avaliação dos mesmos - apresentados no apêndice B, C, D, E, F e G - para facilitar a discussão com o cliente sobre a forma como os aplicativos deveriam agir: disposição dos elementos, funcionalidades, modo de navegação entre telas e a padronização das operações básicas (CRUD). Para enriquecer a documentação técnica e funcional do sistema, foram elaborados diagramas UML, utilizando a ferramenta Astah Community, entre os quais: diagramas de classe, casos de uso e sequência – apresentados nos apêndices I, J, K, L, M, N e O.

Após as etapas de coleta de requisitos e aprovação de protótipos do sistema, iniciou-se o processo de definição e modelagem do banco de dados. O PostgreSQL foi adotado como SGBD. A Modelagem Lógica foi construída com o uso da ferramenta Erwin, apresentada no apêndice H. A familiaridade com o uso da ferramenta e a sua

compatibilidade com o SGBD PostgreSQL foram alguns motivos que levaram ao uso desta aplicação.

Para convencionar nomenclaturas de colunas do banco de dados do sistema, foi adotado o padrão utilizado na Universidade Estadual do Centro Oeste – Unicentro.

O primeiro caractere da nomenclatura identificará o formato básico do dado, independente do tipo de dado do banco. A Tabela 1 apresenta essas informações.

Tabela 1: Caracteres de tipificação (Fonte: Autoria própria).

Caractere	Tipos do Campo
I	Inteiro
D	Decimal
C	Caractere
D	Data
L	Lógico

O segundo e terceiro caracteres indicam a finalidade de uso do campo. A Tabela 2 apresenta o conjunto utilizado no sistema.

Tabela 2: Caracteres de nomenclatura (Fonte: Autoria própria).

Caracteres	Tipos do Campo
Cd	Código
Rt	Rótulo
Dt	Data
Hr	Hora
Nm	Nome
Ds	Descrição
Pc	Percentual
Qt	Quantidade
Tx	Texto
Nr	Número
Vl	Valor

Portanto, no sistema, “cnm_pessoa” é um campo constituído por caracteres, que representará um nome; “inr_ordem” é um campo inteiro, que representará um número ordinal, dessa maneira facilitando a manutenção do sistema. Com esta padronização em mente, ao olhar um atributo de tabela, o desenvolvedor identifica do que se trata.

Na proposta de desenvolvimento, foi identificado como requisito a funcionalidade de operar distintas empresas em uma mesma instalação do sistema. Assim, projetou-se o banco de dados, a partir de uma estrutura hierárquica, de modo que todo e qualquer registro tenha, inequivocamente, relação com uma empresa. Deste modo, reduz-se o custo por cliente e facilita a aplicação de *releases*.

Com o banco de dados estruturado, iniciou-se a fase de codificação da aplicação. Para a implementação, foi utilizado o framework de aplicação CakePHP, sobre um servidor Apache com auxílio da ferramenta Xampp.

Pelo fato do sistema ser desenvolvido baseado em um *framework* que tem como padrão de projeto o MVC, ele já vem com uma estrutura de diretórios pré-definida, apresentado na Figura 3 abaixo.

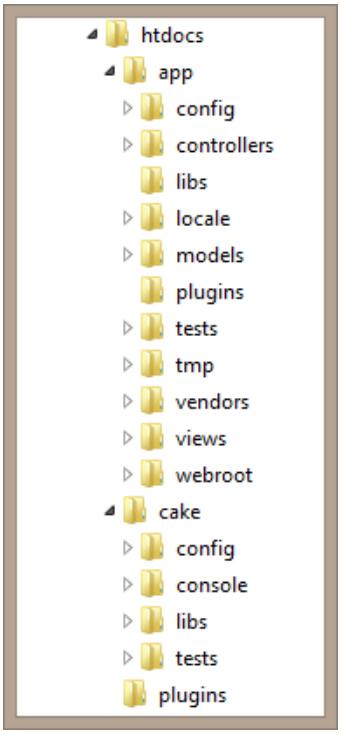


Figura 3: Estrutura de diretórios CakePHP (Fonte: Autoria própria).

No diretório *app* do Cake é realizada a maior parte do desenvolvimento da aplicação. Dentro de *app*, se encontram os principais diretórios: "controllers" - onde ficam as controladoras e componentes da aplicação; "models" - onde ficam os modelos, *behaviors* e *datasources*; "views" - onde os arquivos de apresentação são colocados (elementos, páginas de erros, *helpers* e *layouts*); "webroot" - diretório raiz da aplicação e serve, entre outras coisas, para colocar os arquivos CSS, imagens e JavaScripts. As demais pastas do sistema são destinadas ao *core* do framework, plug-ins e classes de terceiros.

Com o framework configurado, foi iniciado o processo de desenvolvimento do sistema. Para codificação de toda a aplicação, utilizou-se do editor de texto Notpad++. Primeiramente, partiu-se da codificação dos cadastros periféricos do sistema até os cadastros mais ao núcleo da aplicação, que são dependentes daqueles. Os primeiros módulos desenvolvidos e de suma importância, foram o de "usuários" e "empresas", que serão responsáveis pela autenticação e controle de acesso do usuário.

Por padrão, o CakePHP oferece alguns componentes embutidos. Esses componentes trazem métodos capazes de auxiliar e automatizar tarefas comumente utilizadas. A maneira com que o CakePHP trata autenticação é com o componente *Auth*, que se responsabiliza pela autenticação do usuário, sendo mesclado com o componente ACL *(Access Control List*), utilizado para gerenciar o acesso do usuário aos módulos do sistema. Conforme os requisitos do software, três níveis de acesso foram configurados: Administrador Geral - possui acesso a todos os módulos do sistema; Administrador Comum - possui acesso a todos os módulos do sistema, exceto a criação de novas empresas e usuários; Funcionário - possui acesso apenas a cadastros simples e as telas de busca para geração de relatórios.

Em virtude da divisão do desenvolvimento em módulos, o clico de vida de prototipação ajustou-se de forma relevante no processo de desenvolvimento. Este modelo permitiu que antes que cada módulo entrasse em desenvolvimento, fossem criados protótipos para avaliação e discussão com o usuário, possibilitando realizar alterações no software antes de sua versão final estar completa, assim melhor atendendo às expectativas do usuário, evitando possíveis alterações após o software já concluído.

Para satisfazer o principal requisito do sistema, foi desenvolvida uma funcionalidade capaz de expandir os elementos de cadastro, denominada "fatos", baseada no conceito de metadados. A funcionalidade permite adicionar ao sistema novas perguntas relacionadas a pessoas ou eventos. Estas perguntas podem ser de três tipos básicos: texto, lógico ou data. Devido a esta implementação, optou-se em reduzir, nos módulos pessoas e eventos, a quantidade de atributos – já que qualquer necessidade de informação pode ser configurada como um fato.

Um fato é um conjunto de perguntas, que são agrupadas em categorias (que podem se repetir ou não ao longo do tempo), de acordo com o tema ou condição de interesse, aplicado a pessoas ou eventos. Assim, se o usuário desejar manter o cadastro de políticos, a categoria poderia ser denominada "Políticos" e conter fatos como: "Filiação política" e "Mandato", por exemplo. Por sua vez, o fato "Filiação política" poderia conter as perguntas: "Partido" (texto), "Data da filiação" (data), "Unidade da federação" (texto); o "Mandato" poderia conter as perguntas: "Cargo" (texto), "Início" (data), "Término" (data).

Um fato pode, ainda, ser aplicado a um relacionamento entre pessoas. Na interface do sistema, a configuração de fatos aplicados a pessoas/eventos e relacionamentos é distinta, permanecendo a denominação "Fatos" para o primeiro e "Relacionamentos" para o segundo caso. Com esta aplicação dos fatos, é possível guardar informações a respeito de filiação e casamento, por exemplo.

Para a geração de consultas e relatórios dinâmicos do sistema, foram criadas duas *views* no banco de dados: a "buscagerais" e a "calendariosociais". Estas visões são responsáveis por retornar dinamicamente resultados baseados em perguntas do tipo "Texto" e "Data", especificadas em fatos e atribuídas a qualquer evento ou pessoa no sistema.

O ambiente visual do sistema apresentado neste artigo utilizou-se das folhas de estilo disponibilizadas pelo Joomla, porém a disposição dos elementos da interface foi construída de maneira a seguir alguns critérios de ergonomia e usabilidade defendidos pela IHC. Plug-ins para visualização de imagens, formatação de campos, bem como vários recursos da biblioteca JqueryUI foram utilizados para a construção de interações, animações e efeitos avançados.

Ainda dentro do contexto da IHC, todos os módulos contam com um retorno imediato das mensagens de validação, em uma região de fácil visualização. O uso das requisições Ajax em telas onde muitas interações são realizadas paralelamente evitam a atualização do browser, pois apenas a região onde a operação foi realizada será atualizada e apresentada o retorno que a operação obteve.

Por fim, testes funcionais foram realizados com objetivo de validar as entradas de dados do usuário, evitando erros comuns de entrada de informação como: preencher

campos com um número insuficiente de caracteres, preencher campos como nome com valores muito grandes, deixar campos de entrada vazios ou preenchê-los com espaços brancos ou zeros; não respeitar tipos de dados ou duplicar informações.

Resultados

Durante o processo de desenvolvimento do software, apresentado neste artigo, foi de suma importância observar o cenário atual de desenvolvimento em ambiente *web*. Sabendo que os usuários vêm exigindo das interfaces maior agilidade de uso e praticidade, considerou-se acertado o uso das folhas de estilo do Joomla, uma vez que foi possível aplicar conceitos básicos de usabilidade, sem interferir nas funcionalidades propostas. Com o uso de diversas ferramentas e linguagens, foi possível trazer para o ambiente certo dinamismo nas interações, proporcionando ao usuário uma melhor experiência com o software.

A utilização da UML na modelagem do sistema permitiu que o mesmo fornecesse uma visão ampla do projeto durante seu processo de desenvolvimento, colaborando em diversos pontos para a identificação de possíveis falhas na estrutura da aplicação.

Com o ciclo de vida de prototipação, foi possível manter forte vínculo com o usuário, expondo equívocos entre as ideias do cliente e do desenvolvedor. O protótipo de cada módulo serviu como base para especificação do sistema, aumentando a qualidade final de produção do software. Por fim, o sistema desenvolvido atendeu todos os requisitos propostos e as expectativas do usuário.

No que diz respeito a parte funcional, o uso do framework de aplicação possibilitou desenvolver um sistema *web* robusto com agilidade, colaborou para um baixo nível de manutenção e permitiu, ainda, a agregação de serviços durante a fase de codificação. Além disso, possibilitou minimizar a quantidade de linhas de código necessárias e reutilização dos elementos de segurança, agregando segurança ao software.

A liberdade empregada de configurar o software e adaptá-lo dinamicamente para satisfazer as necessidades do cliente beneficia os usuários, que se sentem mais à vontade em utilizá-lo, pois não há necessidade de requisitar alterações nos cadastros, deixando a seu critério como as informações serão especificadas.

Todos os recursos utilizados para desenvolvimento deste sistema foram explorados ao máximo, de maneira a empregar maior qualidade ao software, satisfação do usuário final e adquirir experiência para o desenvolvimento de futuros sistemas com maior eficiência e profissionalismo.

Conclusão

Os desenvolvedores de software precisam conscientizar-se da importância de uma concepção prévia bem elaborada do projeto. Ao contrário, estarão sujeitos a não atender às necessidades do cliente, refletindo em sua insatisfação, com consequente perda de tempo e dinheiro com manutenção e correção de falhas desnecessárias, provenientes da falta de um processo bem definido.

Com o estudo apresentado neste artigo, ressaltou-se a importância da adoção de um modelo que permita a expansão dos elementos de cadastro de um sistema com base

em configurações do usuário. Esta prática pode ser adotada em muitos casos por desenvolvedores que atuam no desenvolvimento de sistemas de informação. Certamente, teriam melhorias na qualidade dos produtos de software e, consequentemente, maior satisfação dos clientes e usúarios.

Optar por um framework se faz muito importante, pois provem de uma estrutura bem definida e padrões de projeto que por fim tornam o software muito mais organizado e robusto, facilitando todo o processo de desenvolvimento.

Referencial Bibliográfico

Barbosa, S. D. J. e Silva, Bruno Santana. (2009) "Interação Humano-Computador". Rio de Janeiro: Editora Elsevier.

Belem, T. (2013) "Frameworks no PHP: O que, quando, porque e qual?". Em: <http://blog.thiagobelem.net/frameworks-no-php-o-que-quando-porque-e-qual/>. Acessado em Outubro.

Bezerra, E. (2007) "Princípios de análise e projetos de sistemas com UML". 2. ed. Rio de Janeiro: Editora Elsevier.

CakePHP. (2013) Em: <http://book.cakephp.org/1.3/pt/>. Acessado em Junho.

Cybis, W., Betiol, A. H. e Faust, R. (2010) "Ergonomia e Usabilidade: Conhecimentos, Métodos e Aplicações". São Paulo: Editora Novatec.

Dall'Oglio, P. (2009) "PHP: Programando com orientação a objetos". 2. ed. São Paulo, Editora Novatec.

Eis, D e Ferreira, E. (2012) "HTML5 e CSS3 – com farinha e pimenta". São Paulo: Tabless.

Elmasri, R e Navathe, S B. (2011) "Sistemas de banco de dados". 6. Ed. São Paulo: Editora Person Addison Wesley.

Koscianski, A e Soares, M. S. (2006) "Qualidade de Software. Aprenda as metodologias e técnicas mais modernas para desenvolvimento de software". 2. ed. São Paulo: Editora Novatec.

Laudon, K. C. e Laudon, J. P. (2007) "Sistemas de informação gerenciais". 7. ed. São Paulo: Pearson Prentice Hall.

Leite, L. A. (2013) "Análise de licenças de software: Relatório Técnico". Em: <http://ltsi.pcs.usp.br/xgov/pub/anexos_xgov/@0031%20LEITE%20Analise%20das%20licencas%20de%20software.pdf>. Acessado em Agosto.

Lima, A. S. (2012) "UML 2.3: Do requisito à Solução". São Paulo: Editora Érica.

Melo, A. A. e Nascimento, M. G. F. (2008) "PHP Profissional: Aprenda a desenvolver sistemas profissionais orientados a objetos com padrões de projeto". São Paulo: Editora Novatec.

Milani, A. (2008) "PostgreSQL: Guia do Programador". São Paulo: Editora Novatec.

Netto, A. A. O. (2010) "IHC e a Engenharia Pedagógica. Interação Humano Computador". São Paulo: Editora Visual Books.

Niederauder, J. (2007) "Web Interativa com Ajax e PHP". São Paulo: Editora Novatec.

Paula Filho, W. P. (2009) “Engenharia de software: Fundamentos, métodos e padrões”. 3. ed. Rio de Janeiro: Editora LTC.

PHP. (2013) Em: <http://www.php.net/manual/pt_BR/>. Acessado em Junho.

Queiroz, M. A. (2013) “A Importância dos Padrões Web para a Acessibilidade de Sites”. Em: <http://www.acessibilidadelegal.com/23-padroes-web.php>. Acessado em Outubro.

Rainer JR., R. K. e Cegielski, G. C. (2011) “Introdução a Sistemas de Informação: Apoiando e transformando negócios na era da mobilidade”. Rio de Janeiro: Editora Campus.

Sales, J. (2009) “Por que desenvolver aplicações para Web?”. Em: <http://jomarsales.blogspot.com.br/2009/01/por-que-desenvolver-aplicacoes-para-web.html>. Acessado em Janeiro.

Santos, R. (2003) “Introdução à programação orientada a objetos usando Java”. Rio de Janeiro: Editora Elseiver.

Silva, M. S. (2008) “Construindo Sites Com CSS e (X)HTML: Sites Controlados por Folhas de Estilo em Cascata”. São Paulo: Editora Novatec.

Silva, M. S. (2010) “JavaScript: Guia do Programador”. São Paulo: Editora Novatec.

Soares, W. (2009) “Crie um Framework para Sistemas Web com PHP 5 e AJAX”. São Paulo: Editora Érica.

Sommerville, I. (2011) “Engenharia de Software”. 9. ed. São Paulo: Editora Ciência Moderna.

Thomson, L. e Welling, L. (2005) “PHP e MYSQL. Desenvolvimento Web”. 3. ed. Rio de Janeiro: Editora Campus.

W3C. (2013) Em: <http://www.w3c.br/Home/WebHome>. Acessado em Setembro.

Xavier, F. S. V. (2008) “PHP – Do Básico Á Orientação A Objetos”. Rio de Janeiro: Editora Ciência Moderna.

Xavier, F. S. V. (2008) “PHP – Do Básico Á Orientação A Objetos”. Rio de Janeiro: Editora Ciência Moderna.

APÊNDICE A – Requisitos do Sistema

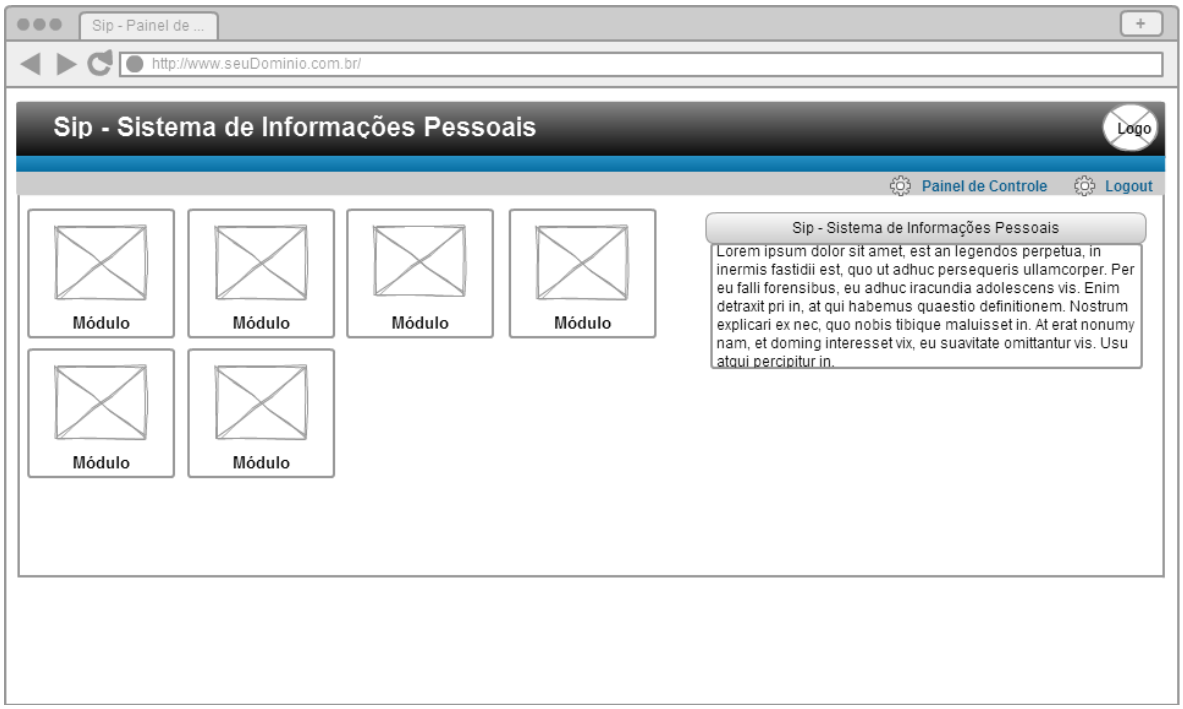
Funcionais:

1. Deve oferecer um sistema de cadastro de usuários com três níveis de acesso.
2. Deve oferecer serviço de gerencia de Empresas, Usuários, Prefixos, Fatos, Categorias para Fatos, Relacionamentos, Eventos e Pessoas.
3. Deve oferecer serviço de cadastro dinâmico dos elementos de informação
4. Deve oferecer serviço de consultas dinâmicas para perguntas do tipo Texto e Data, vinculadas a Pessoas ou Eventos.
5. O sistema deve controlar o acesso a informações diferentes empresas.
6. O sistema deve realizar a validação dos dados informados pelo usuário diante qualquer cadastro na camada *Model*, apresentar um retorno informando qual o campo a validação aplicada.
7. O sistema deve controlar o acesso aos seus recursos, por nível de acesso dos usuários.

Não Funcionais:

1. Usuários deverão operar o sistema após um determinado tempo de treinamento.
2. O sistema deve utilizar o banco de dados PostgreSQL para persistência de dados.
3. A base de dados deve ser protegida para acesso apenas de usuários autorizados.
4. O sistema deverá rodar em qualquer plataforma.
5. Deve ser desenvolvido com utilização do framework de aplicação CakePHP.
6. Ser compatível com os browsers Google Chrome, Mozilla Firefox, Internet Explorer e Opera.
7. Possuir um painel de controle aonde são apresentados os módulos que o usuário possui acesso.
8. Deve ter os procedimentos mais utilizados aperfeiçoados com requisições Ajax.

APÊNDICE B – Protótipo – Painel de Controle



APÊNDICE C – Avaliação de Protótipo – Painel de Controle

Questionário de Avaliação de Protótipo

Software: Sip – Sistema de Informações Pessoais
Protótipo: Painel de Controle

(Preencha com um X a célula escolhida)

Questões	Respostas			
	Totalmente Satisfatório	Satisfatório	Regular	Insatisfatório
Acesso e navegação no sistema	X			
Interface do sistema	X			
Facilidades do uso do sistema	X			
Clareza nos comandos		X		
Disposição dos elementos		X		
Relevância e potencial de utilização das informações apresentadas	X			
As funcionalidades necessárias estão presentes para a prestação dos serviços	X			

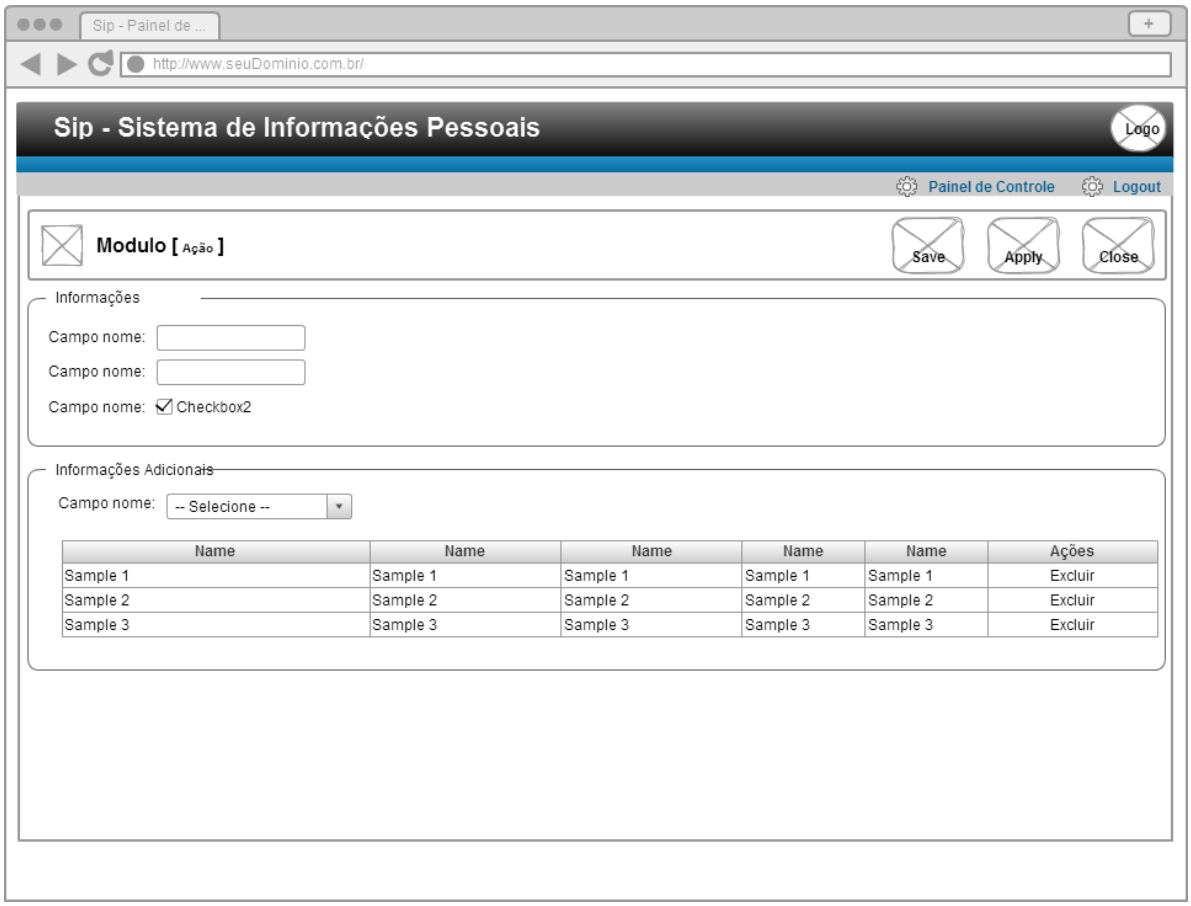
Observações:

Guarapuava, 11 de Setembro de 2013

Lucas Luan Pontarolo
CONTRATADO

Anderson Silvério
CONTRATANTE

APÊNDICE D – Protótipo – Padrão Módulos



APÊNDICE E – Avaliação de Protótipo – Padrão Módulos

Questionário de Avaliação de Protótipo

Software: Sip – Sistema de Informações Pessoais
Protótipo: Padrão Módulos

(Preencha com um X a célula escolhida)

Questões	Respostas			
	Totalmente Satisfatório	Satisfatório	Regular	Insatisfatório
Acesso e navegação no sistema	X			
Interface do sistema	X			
Facilidades do uso do sistema	X			
Clareza nos comandos	X			
Disposição dos elementos		X		
Relevância e potencial de utilização das informações apresentadas		X		
As funcionalidades necessárias estão presentes para a prestação dos serviços	X			

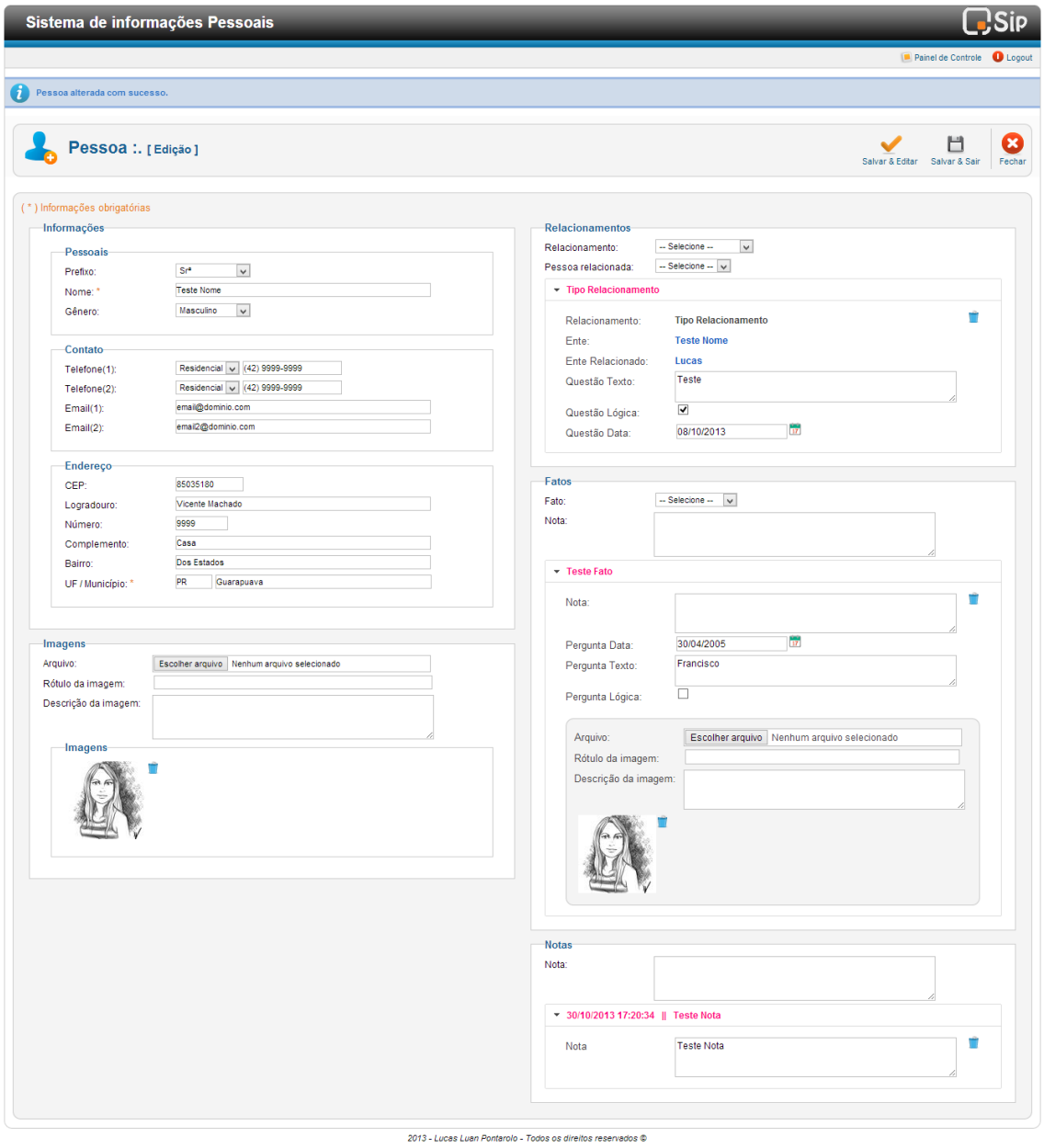
Observações:

Guarapuava, 17 de Setembro de 2013

Lucas Luan Pontarolo
CONTRATADO

Anderson Silvério
CONTRATANTE

APÊNDICE F – Protótipo – Edição de Pessoas



APÊNDICE G – Avaliação de Protótipo – Edição de Pessoas

Questionário de Avaliação de Protótipo

Software: Sip – Sistema de Informações Pessoais
Protótipo: Edição de Pessoas

(Preencha com um X a célula escolhida)

Questões	Respostas			
	Totalmente Satisfatório	Satisfatório	Regular	Insatisfatório
Acesso e navegação no sistema	X			
Interface do sistema	X			
Facilidades do uso do sistema		X		
Clareza nos comandos		X		
Disposição dos elementos	X			
Relevância e potencial de utilização das informações apresentadas	X			
As funcionalidades necessárias estão presentes para a prestação dos serviços	X			

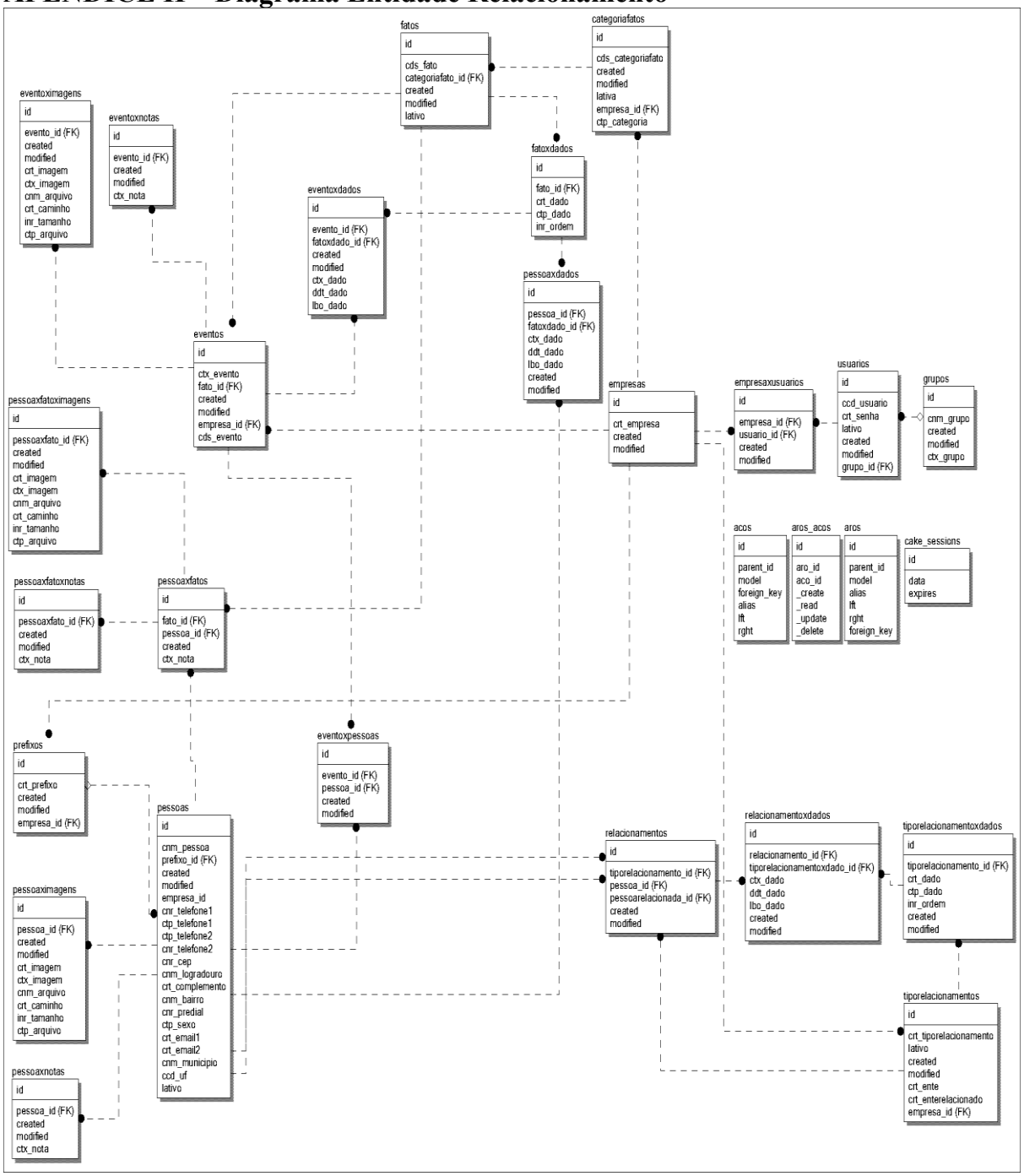
Observações:

Guarapuava, 26 de Setembro de 2013

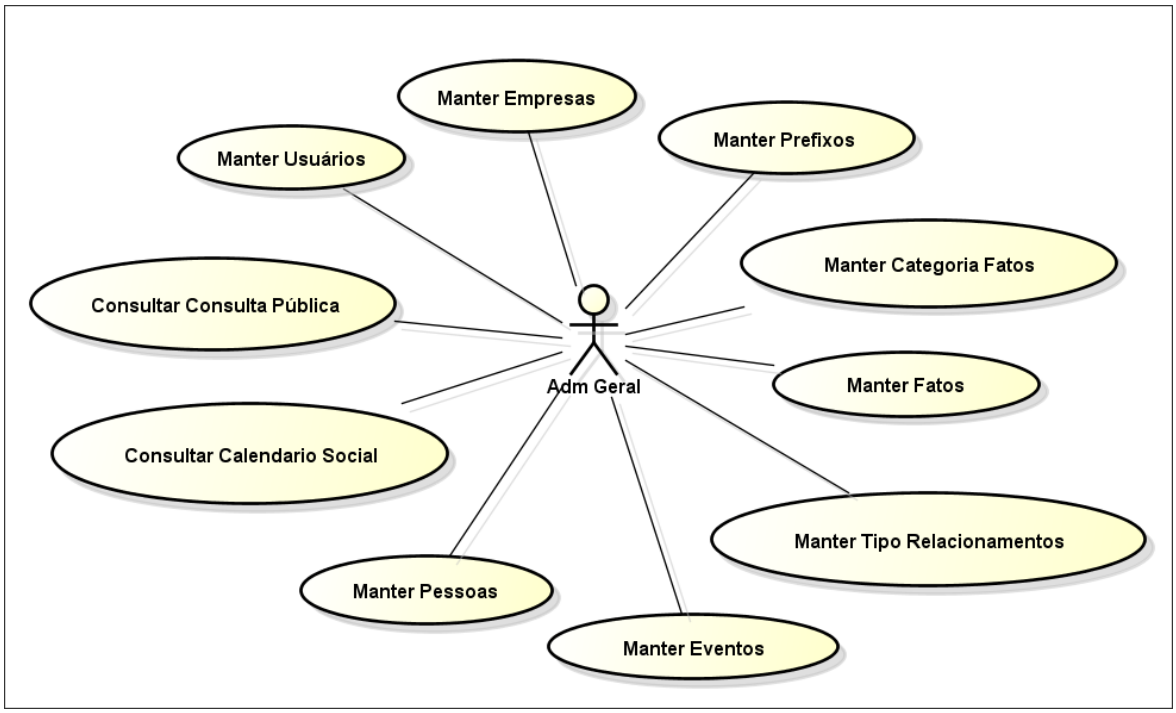
Lucas Luan Pontarolo
CONTRATADO

Anderson Silvério
CONTRATANTE

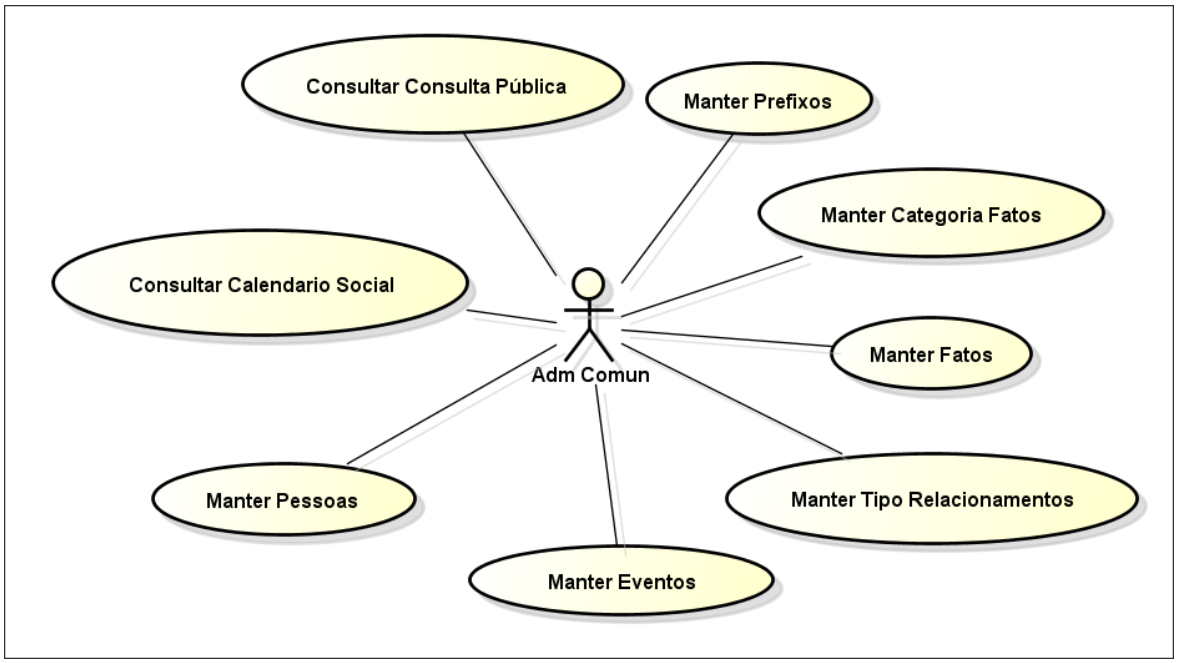
APÊNDICE H – Diagrama Entidade Relacionamento



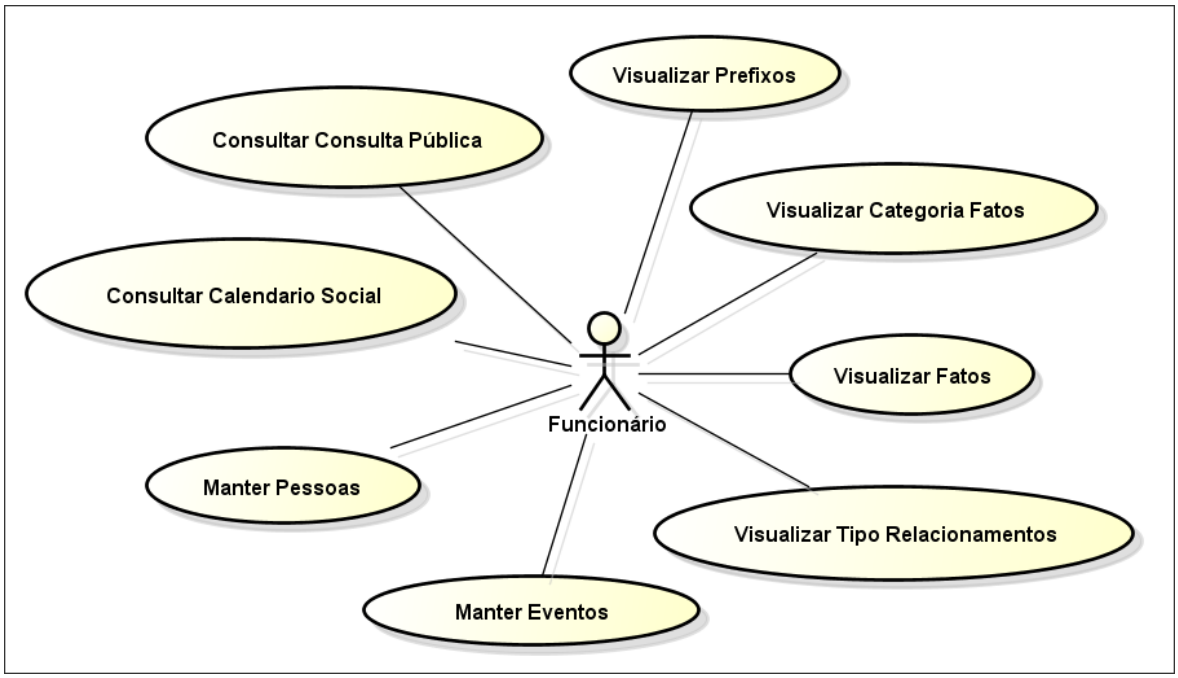
APÊNDICE I – Diagrama de Caso de Uso - Administrador Geral



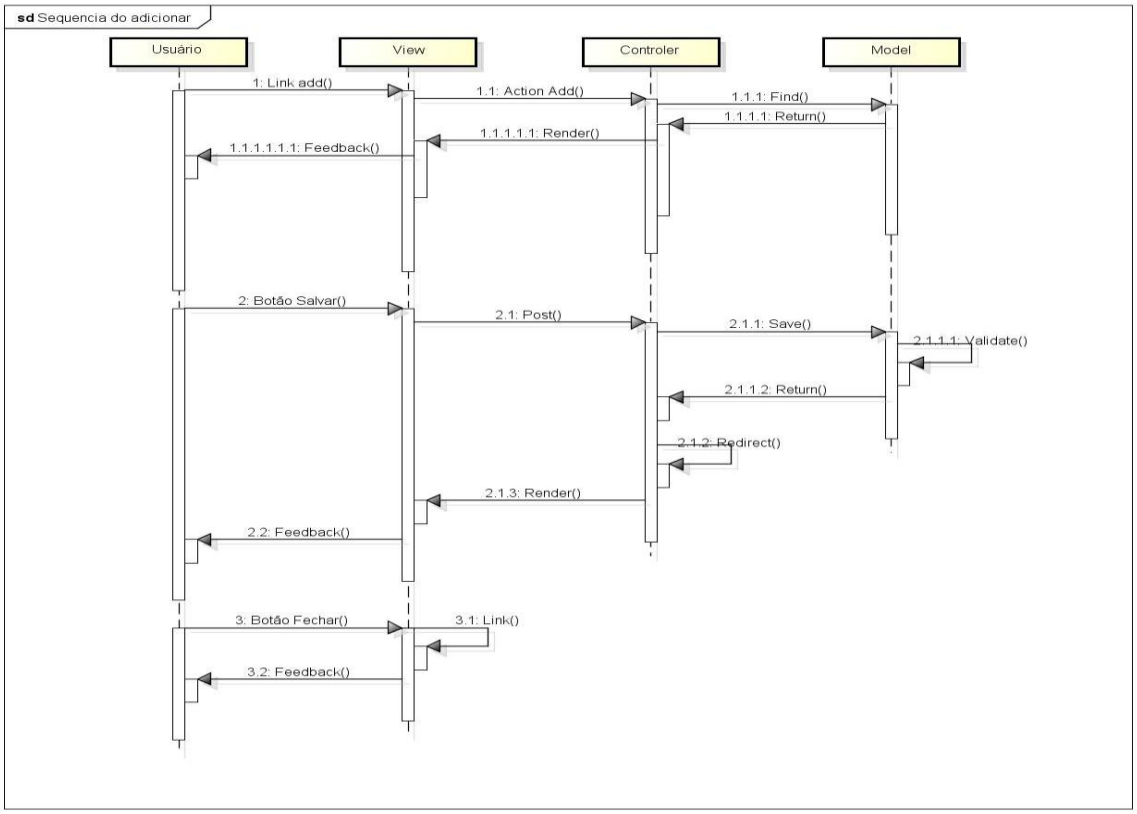
APÊNDICE J – Diagrama de Caso de Uso - Administrador Comum



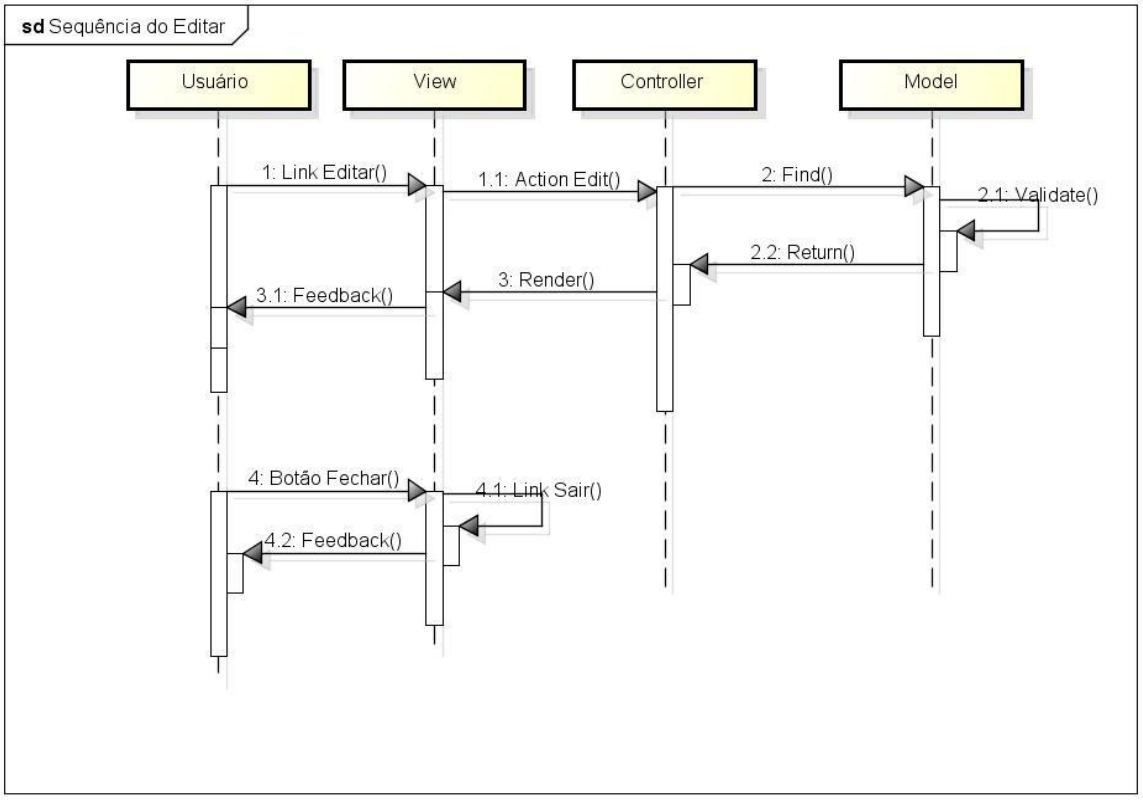
APÊNDICE K – Diagrama de Caso de Uso - Funcionário



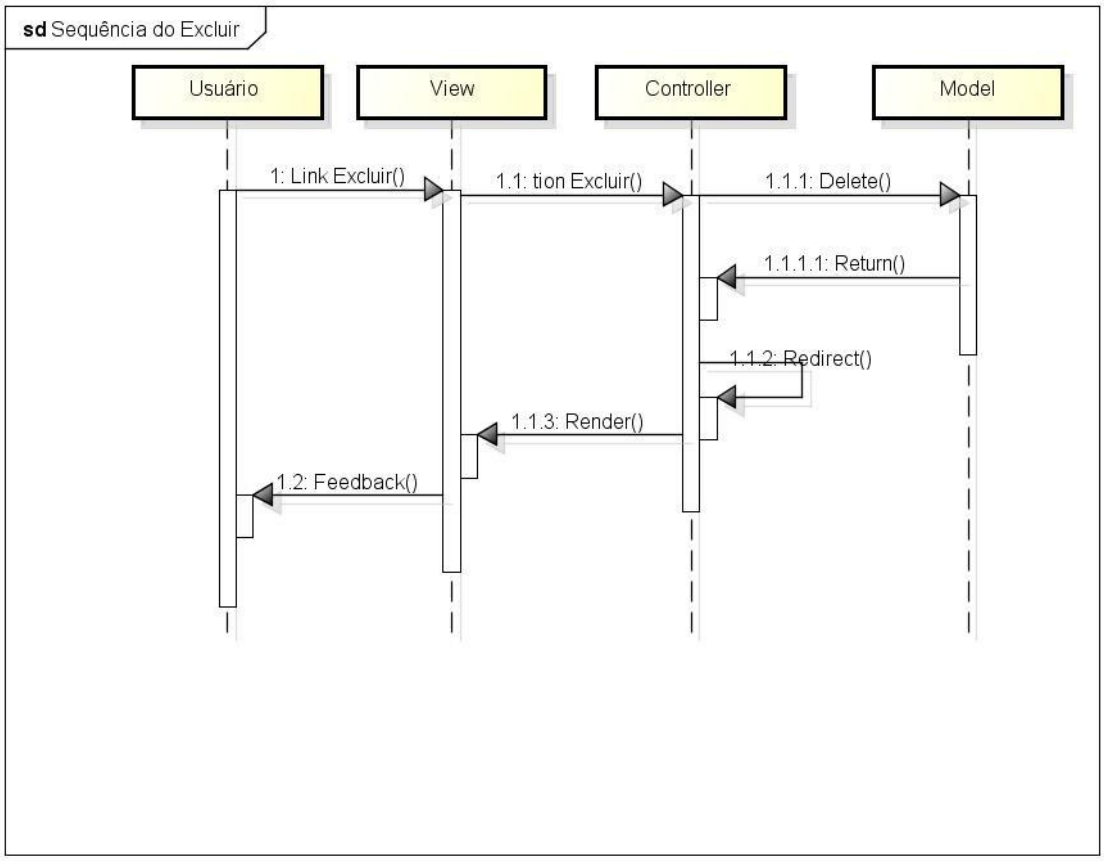
APÊNDICE L – Diagrama de Sequência - Adicionar



APÊNDICE M – Diagrama de Sequência - Editar



APÊNDICE N – Diagrama de Sequência - Excluir



APÊNDICE O – Diagrama de Classes