UNIVERSIDADE FEDERAL DO PARANA

Especialização em Informática – [Desenvolvimento Web]

MARCELO AUGUSTO HENNING

ANÁLISE DE *FRAMEWORKS* PARA O DESENVOLVIMENTO DE SISTEMA DE CONTROLE SINDICAL

CURITIBA
2013

Parecer de Aprovação
Monografia de Especialização em Informática
Ênfase em Desenvolvimento de Sistemas para a WEB
Programa de Pós-Graduação em Informática/UFPR

Declaramos que o aluno **MARCELO AUGUSTO HENNING** entregou a versão final da sua Monografia de Especialização em Informática da Universidade Federal do Paraná, com Ênfase em Desenvolvimento de Sistemas para a WEB, intitulada ***ANÁLISE DE FRAMEWORKS PARA O DESENVOLVIMENTO DE SISTEMA DE CONTROLE SINDICAL.***

Curitiba, 20 de dezembro de 2013

Prof. Dr. Andrey Ricardo Pimentel
Professor Adjunto
Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Informática
Caixa Postal 19081
CEP 81531-990 - Curitiba-PR

Prof. Me. Setembrino Soares Ferreira Junior
Professor Assistente
Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Informática
Caixa Postal 19081
CEP 81531-990 - Curitiba-PR

MARCELO AUGUSTO HENNING

ANÁLISE DE *FRAMEWORKS* PARA O DESENVOLVIMENTO DE SISTEMA DE CONTROLE SINDICAL

Monografia apresentada como requisito parcial para obtenção do grau de Especialista em Informática, com ênfase em Desenvolvimento de Sistemas para a Web, ao Curso de Especialização em Informática, Departamento de Informática, Setor de Ciências Exatas da Universidade Federal do Paraná.
Orientador: Prof. Dr. Andrey Ricardo Pimentel

CURITIBA
2013

RESUMO

Para o desenvolvimento de sistemas de maior porte, com grande nível de integração, está se tornando cada vez mais necessário a adoção de um *framework*. Ele é responsável pela organização do código fonte de todo o sistema, além de prover diversas funcionalidades que facilitam o processo de desenvolvimento. O problema surge diante da diversidade de *frameworks* existentes no mercado, devido à quantidade de requisitos e utilidades diferentes, acaba se tornando necessária uma análise aprofundada antes de uma escolha. Esta monografia tem por objetivo analisar uma determinada quantidade de *frameworks*, levando em consideração o contexto atual da instituição onde será aplicado, um sindicato. Após esta análise, foram feitas comparações entre os *frameworks* eleitos e, a partir daí, selecionado o mais adequado para a instituição.

Palavras chaves: seleção, *framework*, php, sindicato.

ABSTRACT

For the development of larger systems, with high level of integration, it is becoming increasingly necessary to adopt a framework. It is responsible for the organization of source code of the whole system, and provide several features that facilitate the development process. The problem comes on the diversity of frameworks in the market, due to the amount of requirements and different utilities, eventually becomes necessary a thorough examination before a choice. This thesis aims to analyze a certain amount of frameworks, taking into account the current context of the institution which will be applied, a union. After this analysis, comparisons were made between the frameworks elected, and selected the most suitable for the institution.

Keywords: selection, framework, php, trade union.

SUMÁRIO

1 INTRODUÇÃO

Atualmente o sindicato se encontra relativamente informatizado. A maioria das rotinas já se encontram parametrizadas em sistemas, porém ainda existem diversas rotinas ligadas a relatórios que são baseados em planilhas e fichas. Os sistemas existentes estão desenvolvidos em linguagens diferentes, com especificações e plataformas diversas. Também contamos com sistemas proprietários para algumas rotinas específicas.

Sindicatos são, em geral, do ponto de vista do quadro de funcionários, empresas de pequeno porte e, como tal, raramente contam com um departamento de informática. Algumas exceções são empresas com visão estratégica, que primam pela qualidade do serviço prestado, atendimento adequado e sistemas sob medida às suas necessidades.

Infelizmente, ainda em muitos desses casos, a equipe do departamento de informática (às vezes com 1 ou 2 funcionários) administram toda a infraestrutura de TI da empresa. Desempenhando papéis como: desenvolvimento e a manutenção sistemas, intraestrutura completa de servidores, *hardware*, sistemas operacionais tanto em ambiente *desktop* quanto servidor. Cabe também a esses profissionais a manutenção da rede lógica, incluindo cabeamento de rede e telefonia. Sem contar toda a burocracia encontrada na pesquisa e compra de equipamentos de informática.

Diante de questões como as apresentadas, para o desenvolvimento de um novo sistema é necessário uma técnica ágil, e principalmente uma ferramenta que possibilite um aumento de performance no desenvolvimento. Essa necessidade vem de encontro aos *frameworks*, cuja finalidade é justamente padronizar o desenvolvimento, possibilitando que vários programadores trabalhem em conjunto. Além de disponibilizar diversas ferramentas para evitar ao máximo o desperdício de código-fonte.

1.1 PROBLEMA

Com o quadro reduzido de funcionários, a rotatividade de tarefas se torna alta, o que é agravado ao se somar a grande variedade de tarefas desempenhadas. Diante desses fatores, não é possível manter o foco em somente um dos problemas por um longo período de tempo. É necessário controlar diversas situações de acordo com suas prioridades. Isso impacta diretamente na quantidade de inovações do departamento, seja em desenvolvimento de *software* quanto em qualquer outro aspecto.

A escolha de um *framework* é uma escolha difícil, pois dela dependerá todo o sistema que será desenvolvido posteriormente. Possíveis problemas, incompatibilidades ou entraves terão de ser solucionados, gerando às vezes mais problemas do que se nenhum *framework* fosse utilizado.

Embora existam riscos, as vantagens de se utilizar um *framework* se somam às necessidades da empresa. Uma infraestrutura de código robusta, com possibilidade de atender às mais diversas necessidades, possibilitar um crescimento e uma manutenção simplificada, fazem mais do que necessário a adoção de um *framework* para desenvolvimento.

Assim como em toda empresa em processo de informatização, a informação é encontrada em vários estágios de organização. Desde o controle manual em fichas, passando por planilhas e chegando até sistemas propriamente ditos.

O problema é que muitas vezes o sistema em questão já existe há um bom tempo. Foi baseado em uma realidade que a empresa não vivencia mais, e que, mesmo com a informação padronizada e devidamente informatizada, ela é de certa forma impossível de ter seu aproveitamento aprimorado, visto que não é possível a sua integração com outros sistemas, e também que já está funcionando no limite de sua capacidade operacional.

Isso se aplica principalmente no caso de sistemas defasados, onde não é mais possível desenvolver aprimoramentos devido a questões tecnológicas, linguagens de programação e arquiteturas. Nesses casos, são feitas somente

manutenções emergenciais, sem ganhos reais em performance para a organização.

Outro problema, dessa vez partindo para o desenvolvimento de um novo sistema, é que esses sistemas antigos impossibilitam a implantação por módulos de um novo sistema com linguagem atual e com possibilidade de crescimento. Já que por questões de agilidade e segurança, não é possível manter único o local das informações.

Nesses casos não é possível desenvolver um sistema de forma gradativa e colocá-lo em produção, pois a integração com sistemas defasados torna-se impossível em questões de segurança e agilidade, além, é claro, da incompatibilidade de tecnologias. Seria necessário fazer importações e exportações entre os sistemas novo e antigo. Dessa forma, não estaríamos lidando com uma informação unificada, o que pode ocasionar vários problemas. Principalmente quando se trata de um sistema com possíveis acessos simultâneos, em áreas semelhantes do sistema.

Dessa forma, só é possível implantar seguramente um novo sistema após ele estar com toda a estrutura principal pronta. E após ter sido testado em paralelo com o sistema existente.

1.2 OBJETIVO GERAL

Levando em consideração todos os problemas expostos, o objetivo dessa monografia será analisar diversos tipos de *frameworks*, e verificar qual ou quais deles atendem às reais necessidades de um novo sistema, que ainda será desenvolvido.

Foram selecionados *frameworks* de código livre (*Open-source*) por questões de custos e liberdade na utilização do código fonte, nacionais e estrangeiros, baseados em linguagem de programação Java e PHP. Após a seleção, cada um dos *frameworks* será analisado de uma forma geral, ressaltando

seus diferenciais e também situações que para o caso específico do sistema, irão dificultar ou facilitar o desenvolvimento do mesmo.

1.3 OBJETIVOS ESPECIFICOS

Os principais pontos que serão levados em consideração no momento da escolha serão a rapidez na aprendizagem do *framework*, a integração com sistemas existentes, facilidade na criação de módulos adicionais, segurança, interatividade e facilidade de utilização por parte do usuário, geração de relatórios e o suporte a impressão. Demais aspectos serão analisados de acordo com cada *framework* especificamente. Ao final, será possível estimar os prazos de execução para cada etapa do projeto.

Com a escolha de um *framework*, esperamos atender à demanda da instituição, além de possibilitar maior velocidade nas consultas e rotinas, facilitar a manutenção e a criação de novos módulos, bem como agilizar o processo de desenvolvimento de código.

1.4 JUSTIFICATIVA

Com a adoção de um *framework* que venha de encontro às necessidades da empresa, será possível agilizar o desenvolvimento, através do reaproveitamento de código, utilização de rotinas prontas, formulários e relatórios padronizados. Será possível, dependendo do caso, utilizar módulos já criados, como autenticação de usuários.

Será facilitada a manutenção do sistema, visto que, utilizando o modelo *MVC* (*Model, View and Controller*), adotado em todos os *frameworks*

selecionados, é possível analisar separadamente cada uma das partes envolvidas em uma ação no sistema.

Outro ponto importante, levando em consideração o funcionamento do departamento, é a facilidade de se desenvolver o sistema entre vários desenvolvedores. Devido aos padrões adotados, é possível dividir as tarefas de uma maneira muito mais simples. Diante de todas essas questões, será possível desenvolver um novo sistema, com prazos definidos, robusto e de forma ágil.

1.5 ORGANIZAÇÃO DO DOCUMENTO

Após este capítulo introdutório, apresentam-se os a seguir relacionados.

ANÁLISE DO SISTEMA A SER DESENVOLVIDO

O sistema utilizado atualmente na instituição foi analisado, juntamente com as configurações de servidores e terminais (*hardware* e *software*). Posteriormente foram propostas melhorias ao sistema existente e mensurados os impactos relacionados ao desenvolvimento do *software*.

METODOLOGIA DE ANÁLISE DE *FRAMEWORK*

A partir do estudo das necessidades do sistema, partiu-se para o levantamento dos parâmetros necessários para fazer a análise dos *frameworks* disponíveis. Questões de ordem econômica e técnica foram detalhadas de forma que somente *frameworks* realmente úteis fossem selecionados.

FRAMEWORKS

Com o levantamento dos requistos, dentre os vários *frameworks* existentes, foram selecionados cinco *frameworks* que atendem às necessidades básicas do sistema que será desenvolvido.

COMPARAÇÃO ENTRE *FRAMEWORKS*

Após a escolha dos *frameworks*, foram feitas comparações entre eles, das principais funcionalidades necessárias para o pleno funcionamento do sistema a ser desenvolvido. Características como segurança, requisitos de instalação, e detalhes técnicos foram abordados de uma maneira um pouco mais profunda.

DESENVOLVIMENTO E IMPLANTAÇÃO

Com a escolha de um *framework*, deve ser iniciado um estudo para o desenvolvimento e a implantação do sistema. Nesse capítulo são abordadas e exemplificadas diversas situações possíveis de acontecer durante essa etapa.

2 ANÁLISE DO SISTEMA A SER DESENVOLVIDO

Atualmente o sistema de controle sindical em questão se encontra separado basicamente em 4 módulos, e em 3 linguagens diferentes, e todas elas utilizando linguagem estruturada de programação (GANE, 1983). A integração é difícil, em alguns casos devido a sistemas defasados, a performance deixa a desejar, pelo excesso de informações e acessos simultâneos (que não foram previstos na época em que o sistema foi desenvolvido)

Como objetivo para o desenvolvimento do novo sistema, visamos criar um sistema que sirva de base para todos os outros que venham a seguir. Seguindo padrões de desenvolvimento e utilizando as ferramentas adequadas, esperamos que a performance seja melhorada, tanto no sistema principal quanto em módulos posteriores. Há um interesse em utilizar um sistema em que a manutenção seja fácil e intuitiva, o que será conseguido através da modelagem em 3 camadas (MVC). Esse ponto se torna importante, visto que o departamento de informática não dispõe de um profissional dedicado somente ao desenvolvimento.

Os principais requisitos desse sistema são: manter o cadastro de sócios e empresas, manter o histórico de todas as ações, bem como o controle financeiro de todas as contribuições e débitos associados. Junto a isso, deverá disponibilizar relatórios referentes aos cadastros e também à emissão dos boletos de arrecadação do sindicato.

Atualmente contamos com 2 servidores principais, com um misto de banco de dados e sistemas. Utilizam sistema operacional Linux Debian. Nos terminais de acesso é utilizado 80% Linux, e no restante Windows. Devido a alguns sistemas utilizarem linguagens que necessitam do Windows, algumas máquinas ainda o utilizam, porém com o desenvolvimento do novo sistema, o sistema operacional será substituído gradativamente.

2.1 MÓDULO CADASTRO

Este módulo é responsável pelo cadastro dos sócios, sendo limitado somente a trabalhadores da categoria que o sindicato representa. Nesse cadastro constam informações pessoais como nome, sobrenome, RG e CPF. Endereço completo, com rua, número, CEP, bairro, cidade e estado, e informações profissionais, como função desempenhada, empresa na qual trabalha, número do PIS, entre outros.

O cadastro dos sócios se dá a partir da assinatura da proposta social pelo trabalhador. É função do departamento de cadastro verificar se o trabalhador é contratado por uma empresa filiada ao sindicato (empresas do ramo representado pelo sindicato). Todas as informações cadastradas devem ser comprovadas através de documentos, sendo esses copiados e arquivados juntamente com a proposta social assinada. Todos os benefícios do titular são estendidos aos seus dependentes legais, sendo eles esposo (a) ou filhos (as). Nesses casos, os principais dados dos dependentes devem ser cadastrados no sistema para futuras utilizações.

A emissão de carteirinhas é feita para todos os sócios e seus dependentes. A carteirinha contém todas as informações necessárias para identificar unicamente, de maneira fácil, o trabalhador. Ela não possui validade, sendo permitida a sua atualização em casos de perda ou troca do sobrenome.

O cadastro é responsável pela emissão de cupons para os mais diversos eventos disponibilizados pelo sindicato. Dependendo do evento, diversos parâmetros são definidos, sendo possível especificar data, idade, tipo de dependente e sexo. Os cupons são emitidos com código de barras, para que o levantamento de reais participantes possa ser feito de forma mais rápida. Existem ainda casos onde a conferência do cupom precisa ser feita em tempo real.

Com a entrada de um sócio, após um período de carência, ele pode utilizar diversos serviços, desde convênios, passando por assistência médica, até eventos com sorteio de prêmios. Da mesma maneira que um sócio tem o direito de desfrutar de todos esses benefícios, ao se desfiliar, seja por decisão própria, ou

por demissão da empresa, é importante que o seu cadastro seja inativado. Dessa forma ele não poderá gerar mais despesas ao sindicato.

2.2 MÓDULO LIBERAÇÃO

Após o cadastro do associado, é possível ao mesmo fazer a utilização de diversos serviços, sendo que cada tipo de serviço necessita de certo tempo de carência. No cadastro do associado é informada a função desempenhada, e por consequência o salário definido para a categoria.

O sistema conta com a liberação de diversos tipos de convênios, com possibilidade de parcelamento, e limite de utilização de acordo com um percentual do salário. Caso o valor a ser liberado supere o limite, este ainda pode ser liberado com a autorização de um diretor.

A liberação pode ocorrer nas dependências do sindicato, subsedes ou até mesmo em postos conveniados. Através de um sistema integrado é possível que o associado siga diretamente a esses locais portando documento oficial com foto e a carteirinha do sindicato.

Após a liberação de um convênio, com o sistema integrado, o valor utilizado fica automaticamente bloqueado em qualquer posto conveniado. Dessa forma problemas relacionados à utilização indevida ou extrapolação do crédito são evitados. Por fim, essas liberações são contabilizadas de acordo com o mês de referência (liberações parceladas possuem referências sequenciais) e processadas pelo módulo de cobrança.

2.3 MÓDULO COBRANÇA

Após o cadastro do sócio, e posteriormente à utilização de serviços e convênios, todos os débitos referentes a eles devem ser quitados mensalmente.

Cabe ao módulo da cobrança emitir os relatórios de utilização proveniente dos sócios, baseados na referência de utilização. Esses dados são separados por sócio e posteriormente por empresa, gerando um boleto com o somatório das utilizações e convênios, bem como das contribuições previstas na convenção coletiva de trabalho em vigor.

O módulo possibilita a configuração dos tipos de cobrança e dos dados bancários relacionados à emissão de boletos. Também é possível fazer o controle de todos os boletos emitidos, bem como sua reimpressão em casos de perda por parte da empresa.

Após os boletos serem gerados, o módulo fará a confirmação de pagamento através de arquivos de retorno disponibilizados pelo banco, devido a convênio firmado previamente. Através desse controle é possível gerenciar os pagamentos e, em caso de inadimplências, tomar as medidas judiciais cabíveis.

3 METODOLOGIA DE ANÁLISE DE *FRAMEWORK*

O processo de análise dos *frameworks* começou na definição dos pré-requisitos. Após um levantamento detalhado dos conhecimentos atuais da equipe de desenvolvimento, estrutura física da empresa no que diz respeito a servidores, e também a sistemas já existentes, foram definidas regras para que a seleção dos *frameworks* fosse feita (LAKATOS, 1986).

Os pré-requisitos básicos foram a adoção da linguagem de programação PHP, devido ao fato de os desenvolvedores da empresa já trabalharem com a linguagem. Foi definido também que a forma de acesso ao sistema seria via *browser*, minimizando os custos em instalação e manutenção de *softwares*, principalmente em ambiente externo.

Após esse levantamento, uma pesquisa foi feita, levando em consideração todos os *frameworks* que obedeciam aos pré-requisitos. Cada um deles foi analisado separadamente, de acordo com os parâmetros informados a seguir.

3.1 ANÁLISE ECONÔMICA

Devido às políticas financeiras adotadas pela entidade, serão utilizadas linguagens de programação de código livre, que possam ser utilizadas em sistemas operacionais igualmente livres. Como o sistema a ser desenvolvido será utilizado pela própria empresa, num primeiro momento não será feita distinção entre as diversas licenças livres. Será dada preferência para linguagens já dominadas pelo departamento de informática, que será responsável pelo desenvolvimento; portanto, para esse projeto será utilizado o PHP.

Como o sistema utilizado atualmente ainda supre as necessidades da empresa, não existe uma urgência para o desenvolvimento. Como consequência, a possibilidade de terceirizar a confecção do *software* foi descartada, bem como a

contratação de um funcionário auxiliar. Será estudada a possibilidade novamente a partir da metade do projeto.

3.2 REQUISITOS DE *SOFTWARE*

O *software* a ser produzido será utilizado em sistemas operacionais Windows e Linux, sem a adoção de acesso remoto ou semelhantes. Seu acesso será feito tanto na rede interna da instituição quanto externas (subsedes e conveniados). Além disso, alguns módulos do sistema serão acessados diretamente da *internet*, através do portal da instituição (*web site*).

Para reduzir os custos operacionais relacionados à manutenção do sistema cliente, foi optado pela interface *web*, onde basta o computador ter acesso a rede local ou *internet*, para que possa utilizar o sistema, evitando instalações e configurações complexas. O *framework* deverá possuir obrigatoriamente um controle rígido de acesso às rotinas do sistema, para que a segurança do sistema possa ser garantida mesmo em ambientes externos.

O sistema deverá ter sua instalação e pleno funcionamento em plataforma Linux, com os serviços Apache, PHP e PostgreSQL. Para o cliente, o único requisito é o acesso à *internet* e um navegador que siga os padrões W3C (*World Wide Web Consortium*).

Uma preocupação na migração para ambiente *web* é a impressão dos arquivos. Após várias pesquisas foi verificado que pouco evoluiu o ambiente para configuração de relatórios, principalmente em relatórios extensos. De maneira alternativa a um assistente de relatório, será obrigatória a existência de um módulo ou ferramenta que possibilite a geração de arquivos PDF. Em se tratando de relatórios extensos, ou geração em massa de dezenas de relatórios, fica inviável a emissão diretamente pelo navegador, devido ao tempo necessário para a emissão e possíveis travamentos decorrentes de utilização de memória. Para isso, e também para rotinas de manutenção, é indispensável ao *framework*

possibilitar a execução de *scripts* diretamente no terminal. Dessa maneira, o relatório seria agendado e, depois de gerado o arquivo PDF, seria disponibilizado para o usuário.

O *framework* deverá validar todos os valores informados nos formulários do sistema, bem como padronizar a exibição através de máscaras. Cabe também ao *framework* verificar todas as informações para evitar os principais tipos de ataques a sistemas, sejam pela utilização de linhas de código maliciosas, ou a execução de instruções ilegais no banco de dados.

3.3 CURVA DE APRENDIZAGEM

Mesmo com o prévio conhecimento na linguagem adotada pelo *framework*, é necessário que os desenvolvedores aprendam a utilizá-lo da melhor forma possível. É muito comum, em períodos iniciais de um projeto utilizando metodologias novas, que o tempo gasto para análise do *framework* seja muito maior do que o tempo gasto em desenvolvimento do sistema propriamente dito.

Nos casos em que a equipe de desenvolvimento possui poucos funcionários, e focados em tarefas diferentes, isso pode ser um diferencial ainda maior, visto que o tempo poderá ser mais bem aproveitado no desenvolvimento propriamente dito.

O tempo de aprendizagem também pode ser muito benéfico em situações de manutenção, tanto na correção de *bugs* quanto no estudo de novas funcionalidades, e até mesmo na instalação de *adons* e *plugins* disponibilizados pelo fabricante do *framework*.

3.4 TESTES

É esperado que os *frameworks* selecionados permitam a utilização de rotinas específicas para teste de *software*. Seja por meio de recursos próprios ou através de sistemas paralelos, deve haver um auxílio ao desenvolvedor no momento da codificação do sistema. Rotinas que permitam a rápida identificação de erros no momento da compilação, ou mesmo que possam analisar diversas variáveis possíveis, com certeza farão grande diferença em todo o processo de desenvolvimento do *software*.

Embora existam ferramentas capazes de fazer testes básicos, até mesmo incorporados ao próprio editor de código, ou IDE de programação, a utilização de ferramentas específicas torna os testes menos suscetíveis a falhas. Além de melhorar a segurança do *software*, e também de quem o desenvolve, possibilita que sejam efetuados testes entre diferentes equipes e/ou programadores.

3.5 MANUTENÇÃO

Após todo o período de desenvolvimento e a devida homologação para uso através de uma bateria de testes, inevitavelmente chegará o momento em que serão necessárias alterações no sistema. Entra novamente em cena o *framework*, dele dependerá a facilidade de se desenvolver funcionalidades novas, alterar o funcionamento de fluxos anteriormente utilizados, e até mesmo corrigir eventuais problemas que passaram despercebidos pela fase de testes.

Geralmente é nesse ponto que muitas complicações ocorrem. As manutenções conseguem ser feitas com sucesso, porém a um alto custo. Isso pode ser evitado com uma organização estrutural do código já preocupada com as futuras manutenções. Um *framework* que possibilite eventuais alterações, e que

não seja totalmente engessado, ajudará muito no momento de se expandir módulos específicos do sistema.

4 *FRAMEWORKS*

No sentido mais amplo, *framework* seria uma maneira de se organizar a informação, um modelo de dados. Já na área do desenvolvimento, o *framework* trata dessa organização, porém a um nível mais profundo. Nele são definidas regras de negócio e de acesso às informações. Geralmente construído com o propósito de atender áreas específicas, é notado a semelhança entre os *frameworks* desenvolvidos. Diante disso, podemos afirmar que o uso de um *framework* genérico pode ser utilizado para o desenvolvimento das mais diversas aplicações. Isso já vem sendo feito na grande maioria das linguagens de programação modernas, com grandes ganhos em performance e segurança.

Podemos afirmar que um *framework* orientado a objetos permite a reutilização de código, tanto desenvolvido para o sistema da instituição quanto os já presentes no *framework*, que geralmente atendem os requisitos mais básicos e comuns à maioria dos sistemas. Com a divisão em módulos, e a implementação de mecanismos de segurança internos, os *frameworks* conseguem manter-se estáveis e seguros, podendo ser utilizados em sistemas de todos os tamanhos. (FAYAD; SCHMIDT, 1997)

O desenvolvimento de sistemas de grande porte já é por si só um trabalho complexo. Com a adoção de um *framework*, o trabalho tende a se tornar ainda mais complexo, porém com a utilização de um padrão, todo o processo de desenvolvimento é desmistificado, sendo mais fácil o trabalho em equipe, o crescimento do sistema e a futura manutenção. (FAYAD; SCHMIDT, 1997)

Levando em consideração a metodologia mencionada anteriormente, foram selecionados cinco *frameworks* para uma análise detalhada, evidenciando funcionalidades que serão de vital importância no sistema que será desenvolvido, o sistema de controle sindical. Todos os *frameworks* relacionados abaixo são baseados na linguagem de programação PHP, para desenvolvimento *web*. Utilizam o modelo de arquitetura de *software* MVC, junto com a orientação a objetos.

4.1 *CAKEPHP*

Lançado em 2005 por Michal Tartarynowicz, o CakePHP foi criado baseado nos modelos propostos pelo Ruby on Rails, com o propósito de criar sistemas robustos e flexíveis (CAKE SOFTWARE FOUNDATION, 2013).

Possui a licença MIT, uma das mais livres dentre as disponíveis. Esta licença deixa clara a liberdade de utilização, sendo possível alterar e vender *softwares* baseados no *framework*.

Em sua estrutura está prevista a validação de informações nos formulários, assim como a utilização de Ajax para a exibição de informações, deixando o sistema mais amigável. Também prevê a proteção de dados inseridos em formulários, evitando problemas relacionados à invasão por códigos maliciosos, ataque a banco de dados, entre outros. (CAKE SOFTWARE FOUNDATION, 2013)

Em sistemas que utilizam muitos cadastros simples (CRUD), o CakePHP apresenta uma alternativa interessante, uma ferramenta para a geração de códigos básicos. Nele é possível gerar praticamente todas as telas de cadastro e alteração informando simplesmente quais campos serão utilizados.

4.2 *CODEIGNITER*

Desenvolvido pela EllisLab, o *CodeIgniter* tem como ponto forte a fácil adaptação aos servidores (Sistema operacional e dependências relacionadas ao PHP) e a utilização de bibliotecas menores e ajustáveis. Isso é muito bem vindo, visto que o sistema a ser desenvolvido foge de certa forma ao padrão de desenvolvimento desse *framework*, que são sistemas inicialmente criados para a *internet*. Ao mesmo tempo o *CodeIgniter* possui uma grande quantidade de ferramentas e utilitários, onde é possível executar diversas tarefas com o mínimo de codificação.

Dentre os *frameworks* estudados, é o que apresenta maior facilidade de aprendizado, devido a sua simplicidade e ótima documentação. Em sua estrutura básica já está prevista a validação dos dados de formulários, porem não prevê a utilização de Ajax de forma nativa, o que deixa a interface com o usuário menos amigável. Em se tratando de um sistema que vai substituir outro já obsoleto, detalhes que facilitam a utilização tem grande importância. (ELLISLAB, INC, 2013)

Questões de performance não foram levadas em consideração visto que a alteração de poucos parâmetros dentro de cada *framework* faz com que a mesma seja elevada. Em algumas situações o *CodeIgniter* possui vantagens em performance, mas nada significativo diante do contexto geral, de confiabilidade e variedade de funcionalidades.

4.3 *SYMFONY*

Lançado em 2005, pela empresa Sensio Labs, o *Symfony* vem ganhando seguidores devido ao seu desenvolvimento robusto e modular. Talvez um dos poucos *frameworks* desenvolvidos para sistemas *web*, porém com preocupação em atender à demanda crescente de *softwares* originalmente de plataforma *desktop* para o ambiente *web*.

Uma das preocupações dos criadores do *Symfony* foi facilitar ao máximo a tarefa de testar o sistema que está sendo desenvolvido. Através de uma camada específica, é possível verificar com riqueza de informações tudo que está acontecendo na execução do sistema, podendo inclusive simular requisições HTTP. O *Symfony* também é integrado ao PHPUnit, onde é possível fazer uma bateria de testes, de forma totalmente automatizada, facilitando ainda mais no caso de sistemas de médio e grande porte. Um ponto positivo, visto que na programação PHP simples, os testes e a visualização dos erros são muito complicados. Geralmente os desenvolvedores criam funções próprias para facilitar essa tarefa (SENSIO LABS, 2013).

Através da integração com diversos *softwares* de código livre, o *Symfony* é capaz de desempenhar diversas tarefas, como o mapeamento objeto-relacional e também a persistência dos dados, feitos através de integração com o *software* “Doctrine 2”, e também a execução de operações diretamente nos bancos de dados. Isso se torna muito importante, principalmente quando tratamos de bases de dados de grande porte, onde a execução de operações diretamente no navegador podem gerar travamentos entre outros problemas. Outra ferramenta interessante é o “Capifony”, onde é possível executar *scripts* em servidores fisicamente separados (SENSIO LABS, 2013).

O *Symfony* também possibilita a total separação entre os módulos do sistema, facilitando seu entendimento e também dando autonomia aos desenvolvedores de cada módulo.

4.4 *YII*

Ultimamente um dos *frameworks* mais procurados, o *Yii* tem como propósito ser um *framework* rápido, seguro e profissional. Seu projeto foi iniciado em janeiro de 2008, por Qiang Xue, mesmo desenvolvedor e mantenedor do *framework* Prado. O *Yii* acabou de certa forma incorporando diversas idéias de outros grandes *frameworks*, dentre eles o *Symfony*, já abordado nessa monografia.

Assim como os demais *frameworks* abordados, o *Yii* adota a arquitetura MVC, com abstração de banco de dados, validação de formulários, Ajax e módulo de autenticação, proteção contra invasão (SQL Injection, XSS, entre outros.) Também permite a criação de telas de cadastro facilmente através de um método totalmente automatizado (YII SOFTWARE LLC, 2013).

Como diferenciais desse *framework* podemos citar a possibilidade de se criar skins e temas para os sistemas, sendo possível alternar facilmente entre os mesmos. Possui também classes específicas para utilização de *web services*, como por exemplo uma pesquisa de CEP através de um *site*, ou cálculo de frete.

Embora não seja uma funcionalidade primordial para o contexto, o sistema de cache em camadas torna o *Yii* um *framework* muito ágil, sendo possível configurar diversos parâmetros para que se adaptem da melhor forma possível ao sistema.

Sua documentação é excelente. Já de início, ao se acessar o *site* do projeto, é muito simples encontrar todas as informações a respeito do *framework*. É possível notar a preocupação com a simplicidade, todos os temas são abordados de forma prática e direta. Além do *site*, é possível ter acesso a diversos livros sobre o *framework*, além de fóruns e *chats*.

4.5 *ZEND FRAMEWORK*

Lançado em 2005, pela *Zend Technologies*, o *Zend Framework* aposta na baixa dependência entre componentes. Isso permite que os desenvolvedores usem os mais diversos componentes em sua estrutura, tornando esse *framework* mais fácil de se adaptar, o que, em alguns casos, acaba sendo o seu maior ponto positivo em relação aos outros *frameworks* (ZEND TECHNOLOGIES, 2013).

Com estrutura totalmente orientada a objetos, este *framework* disponibiliza uma plataforma robusta com metodologia MVC, incluindo várias funcionalidades, como abstração de banco de dados, além da validação e tratamento de dados inseridos em formulários. Através de componentes específicos, é possível adicionar funções de segurança, como autenticação e listas de acesso, entre outros.

Possui a licença New BSD, com a qual é possível distribuir comercialmente o *software* desenvolvido desde que sejam mantidas e respeitadas as suas cláusulas. Dependendo do propósito do *software,* isso pode ser uma grande vantagem, o que não é o nosso caso, visto que o *software* será desenvolvido para o uso da própria empresa.

O *Zend Framework* possui várias funções prontas para serem utilizadas em conjunto com grandes *site*s, como Amazon, Google e Yahoo. Isso torna a tarefa de desenvolver sistemas integrados muito mais fácil. Ainda focado na facilidade, já de início o *framework* dispensa maiores configurações. Possui uma arquitetura flexível e um ambiente testado exaustivamente. (ZEND TECHNOLOGIES, 2013).

Assim como a maioria dos *frameworks*, o *Zend Framework* possui diversas funcionalidades praticamente indispensáveis a qualquer sistema, como o suporte a Ajax, validação de formulários, pesquisa, autenticação, entre outros.

5 COMPARAÇÃO ENTRE *FRAMEWORKS*

Após a análise detalhada dos *frameworks* selecionados, foi possível verificar diferenças interessantes entre cada um deles, que fazem grande diferença dependendo do tipo de sistema a ser desenvolvido. Como pode ser visto nas tabelas a seguir, foram comparados os *frameworks* de acordo com as características desejáveis ao sistema de controle sindical.

Foi levada em consideração a documentação oficial de cada *framework*, disponibilizada no *site*. No caso de *plugins* e extensões, foram selecionados somente os que possuem larga utilização, ou confiabilidade mínima comprovada (seja em utilização em sistemas atuais ou tempo de funcionamento). Vale ressaltar que diversas informações abaixo poderão variar com o passar do tempo. O que está sendo retratado abaixo reflete a situação atual, referente ao ano de 2013.

5.1 LICENÇA

QUADRO 1: COMPARAÇÃO ENTRE AS LICENÇAS DOS *FRAMEWORKS*

CAKEPHP	*CODEIGNITER*	*SYMFONY*	*YII*	*ZEND FRAMEWORK*
MIT	OSLv3	MIT	NewBSD	NewBSD

A licença MIT, criada pelo Instituto de Tecnologia de Massachusetts, é uma licença utilizada no desenvolvimento de *software* livre. Através dela é possível reutilizar o código fonte sob a licença em *softwares* livres ou proprietários. Ou seja, é possível desenvolver um *software* para qualquer utilização.

Nascida na Universidade de Berkeley, a licença BSD impõe poucas restrições, assim como se pode observar na licença MIT. A licença BSD foi revisada desde sua criação, tendo a sua última versão a chamada New BSD. O

texto da licença pode ser alterado sem qualquer tipo de restrição. Este *software* também pode ser utilizado em *softwares* proprietários.

A Open *Software* License v. 3.0, ou OSLv3, criada por Lawrence Rosen em 2005, é muito similar a LGPS (*GNU Lesser General Public License*). Ela basicamente permite a utilização do código sem custo algum, desde que seja informado onde foi baseado; caso sejam feitas modificações, essas também devem ser claramente explicitadas.

Como o sistema que será desenvolvido tem por propósito atender a própria instituição, sem intenção de venda, podemos afirmar que todas as licenças poderão ser utilizadas sem qualquer tipo de privação na utilização.

5.2 FACILIDADES DE APRENDIZAGEM

QUADRO 2: COMPARAÇÃO ENTRE A FACILIDADE DE APRENDIZAGEM ENTRE OS *FRAMEWORKS*

CAKEPHP	***CODEIGNITER***	***SYMFONY***	***YII***	***ZEND FRAMEWORK***
9	10	7	8	5

* NOTA: Foram atribuídas notas para efeito de comparação entre os *frameworks*. O *framework* com maior facilidade recebeu a nota máxima (10).

Nos casos analisados, é possível relacionar a dificuldade de aprendizado com a robustez do *framework*. Os *frameworks CodeIgniter* e CakePHP, por serem inicialmente voltados para o desenvolvimento de *site*s, possuem uma facilidade maior de aprendizagem, pois geralmente atenderão a projetos menores.

O *framework Yii* teve um rendimento médio, como é um dos *frameworks* mais recentes dentre os analisados, pôde juntar os pontos positivos de diversos *frameworks*. Um deles foi a facilidade de aprendizagem dos *frameworks* voltados para o desenvolvimento de *site*s, como o *CodeIgniter* e o CakePHP.

Por último temos o *Zend Framework* e o *Symfony*, *frameworks* com uma grande robustez, com camadas muito bem divididas e um processo de utilização bem parametrizado, o que acaba por tornar o seu aprendizado um pouco mais difícil e demorado.

É possível afirmar que, para profissionais com domínio da linguagem PHP e orientação a objetos, o aprendizado se torna um pouco demorado, o que pode ser melhorado com a qualidade da documentação e a utilização de exemplos. O que pode dificultar é principalmente a divisão em camadas, adotado por todos os *frameworks* (MVC). Caso o profissional já tenha trabalhado com programação em camadas, o trabalho de aprendizagem será somente se acostumar com o padrão adotado.

5.3 COMPLEXIDADES DE INSTALAÇÃO

QUADRO 3: COMPARAÇÃO ENTRE AS COMPLEXIDADES DE INSTALAÇÃO DOS *FRAMEWORKS*

CAKEPHP	***CODEIGNITER***	***SYMFONY***	***YII***	***ZEND FRAMEWORK***
6	5	9	8	10

* NOTA: Foram atribuídas notas para efeito de comparação entre os *frameworks*. O *framework* com maior complexidade recebeu a nota máxima (10).

De acordo com a complexidade de instalação do sistema básico dos *frameworks*, foram atribuídas notas para efeito de comparação. Frameworks com requisitos mais específicos de instalação receberam notas mais baixas. Abaixo o detalhamento dos requisitos e procedimento básico da instalação de cada um deles.

Para a instalação do CakePHP, é necessário um servidor HTTP (geralmente Apache, devido à variedade de configurações e versatilidade), PHP

4.3.2 ou superior, e, para a utilização de banco de dados, existe suporte para os mais utilizados do mercado, dentre eles MySQL e PostgreSQL. Na instalação, basicamente é feita uma cópia dos arquivos para o diretório *web*, e configurada a permissão de escrita no diretório /app/tmp (CAKE SOFTWARE FOUNDATION, 2013).

O *framework CodeIgniter* necessita, para sua instalação, um servidor HTTP, PHP 5.1.6 e bancos de dados como MySQL e PostgreSQL. Para instalação, assim como no CakePHP, basta fazer uma cópia dos arquivos fonte para o diretório *web*, e fazer a configuração de dois parâmetros, a URL base e a conexão com o banco de dados (CAKE SOFTWARE FOUNDATION, 2013).

Partindo para *frameworks* mais robustos, o *Symfony* necessita para sua instalação, do PHP 5.3.3, JSON para a comunicação interna do *framework* (substituindo o XML), *ctype* para a verificação do tipo de variáveis e também as configurações de fuso horário dentro do arquivo “PHP.ini” precisam estar configuradas (SENSIO LABS, 2013).

Os requisitos mínimos para a instalação do *Yii* em um servidor são o suporte ao PHP 5.1.0 ou superior e um serviço HTTP como o Apache. Basicamente estes são os únicos requisitos para o funcionamento básico do *Yii* (YII SOFTWARE LLC, 2013).

Em sua última versão, o *Zend Framework* necessita do PHP 5.3.3, juntamente com serviço HTTP. Para a execução de testes de código, é necessário a instalação do PHPUnit. Dependendo dos componentes utilizados pelo *Zend Framework*, é necessário que várias extensões do PHP sejam ativadas, dentre elas a Apc “Ctype” para validação do tipo de variáveis, “Dom” para utilização com XML e “Hash” para diversos componentes como autenticação e pesquisa (ZEND TECHNOLOGIES, 2013)

5.4 SEGURANÇA

QUADRO 4: COMPARAÇÃO ENTRE OS MÉTODOS DE SEGURANÇA ADOTADOS PELOS *FRAMEWORKS*

MÉTODO	***CAKEPHP***	***CODEIGNITER***	***SYMFONY***	***YII***	***ZEND FRAMEWORK***
Validação em formulários	V	V	V	V	V
SQL Injection	V	V	V	V	V
Script Injection	V	V	V	V	V
Módulo de autenticação	V	-	V	V	V

Dentre os vários níveis de segurança a serem aplicados em uma aplicação *web*, começamos pelo tratamento e validação das informações inseridas em formulários, normalmente o maior causador de falhas, pois nos casos mais específicos analisados posteriormente, já se tratam de atitudes de má fé por parte do usuário do sistema (ou um invasor).

Todos os *frameworks* possuem funções específicas para a validação de formulários, sendo possível definir regras para as validações, como tipo de valor, tamanho e formatação do valor a ser digitado, com possibilidade de incluir máscaras personalizadas dependendo do tipo do dado, como datas e número de documentos.

A prevenção de invasões do tipo “*SQL Injection*”, em que se trata de códigos maliciosos executados através de brechas do sistema diretamente na base de dados, e de “Script Injection”, onde códigos maliciosos alteram o comportamento do sistema, podem ser evitadas com o tratamento adequado de toda informação proveniente de fora do próprio sistema. Informações de *login*, senha, atualizações em cadastros ou até mesmo a impressão de um documento podem permitir, sem o devido tratamento, que uma pessoa má intencionada danifique o sistema.

Devido a isso, todos os *frameworks* selecionados prevêem a análise dos dados antes de darem sequencia ao processamento. Alguns *frameworks*

necessitam de codificação específica para tal, porém nada que interfira na segurança.

Outro ponto de segurança, dessa vez mais perceptível ao usuário comum do sistema, trata do controle de acesso, onde definimos quais ações um usuário pode executar no sistema. Isso é possível através de um módulo de autenticação, que pode ser encontrado em todos os *frameworks*, exceto o *CodeIgniter*, onde é preciso adicionar uma extensão criada pela comunidade para tal funcionalidade.

5.5 AGILIDADES NO DESENVOLVIMENTO

QUADRO 5: COMPARAÇÃO ENTRE FUNCIONALIDADES QUE PERMITEM AGILIDADE NO DESENVOLVIMENTO

MÉTODO	***CAKEPHP***	***CODEIGNITER***	***SYMFONY***	***YII***	***ZEND FRAMEWORK***
Geração de formulários (CRUD)	V	V	V	V	V
Geração de classes	V	V	V	V	V
Tratamento de erros	V	V	V	V	V

Todos os *frameworks* apresentam, de alguma maneira, a possibilidade de se criar modelos ou simplesmente formulários padrão. Também chamada de CRUD (*Create, read, update and delete*) esta funcionalidade se torna importante em sistemas onde existe pouca ligação entre classes, ou com grande ramificação nas extremidades do diagrama de classes. Além das telas comuns ao cadastro de uma entidade, é possível também gerar boa parte do código relacionado ao modelo de cada classe, como a camada responsável pela persistência dos dados e também da classe representada pelo mesmo.

Assim como as funções acima, todos os *frameworks* também apresentam funções específicas para o tratamento de erros, de forma a traduzir para o usuário o problema que foi gerado devido ao erro. Também é possível criar modelos de

páginas de erro para que, nesses casos, a mensagem exibida esteja de acordo com o *layout* adotado pelo resto do sistema.

5.6 PERFORMANCE

QUADRO 6: COMPARAÇÃO ENTRE FUNCIONALIDADES QUE PERMITEM O GANHO DE PERFORMANCE NA EXECUÇÃO DO SISTEMA

MÉTODO	***CAKEPHP***	***CODEIGNITER***	***SYMFONY***	***YII***	***ZEND FRAMEWORK***
Suporte à APC	V	V	V	V	V
Controle de Cache	V	V	V	V	V
Controle de compontenes	-	-	-	-	-

O APC, "*Alternative PHP Cache*" é uma extensão do PHP para a utilização de *cache* nos sistemas. Atualmente é uma das maneiras mais eficientes de se elevar a performance, visto que é uma função integrante do próprio PHP. Para o total aproveitamento, é necessário que o *framework* tenha suporte a essa tecnologia. Através de classes e métodos específicos é possível ao programador controlar com maior eficiência todas as variáveis e arquivos envolvidos na rotina que está sendo desenvolvida.

O controle de *cache* também pode ser feito diretamente pelo *framework*. Com a persistência de informações previamente utilizada, tanto em banco de dados como em arquivos de texto, é possível aumentar a velocidade no funcionamento do sistema, pois as informações já se encontram previamente processadas e salvas.

Além do *cache* de páginas, a performance pode ser aumentada com o uso racionalizado de classes e métodos. Em se tratando de *frameworks*, é comum nos depararmos com dezenas de classes interligadas, o que torna o sistema inevitavelmente mais lento do que um sistema comum. Com esse controle de

componentes, é possível utilizar somente as classes necessárias e no momento necessário. Evitando a comum prática de se carregar todas as classes durante qualquer rotina no sistema.

5.7 IMPRESSÃO E RELATÓRIOS

Talvez um dos principais pontos fracos dos *frameworks* abordados, e também uma particularidade de sistemas voltados para a *web*, o suporte à impressão e relatórios, é complicado e deficiente.

QUADRO 7: COMPARAÇÃO ENTRE FUNCIONALIDADES QUE AUXILIAM A IMPRESSÃO E EMISSÃO DE RELATÓRIOS

MÉTODO	***CAKEPHP***	***CODEIGNITER***	***SYMFONY***	***YII***	***ZEND FRAMEWORK***
Componente para geração de arquivo PDF	-	-	-	-	V
Ambiente gráfico para desenvolvimento de relatórios	-	-	-	-	-
Execução em lote de *scripts* (Linha de comando)	V	V	V	V	V
Assistentes de relatório	-	-	-	-	-
Componente auxiliar para impressão	-	-	-	-	-

Foram analisados cinco fatores que, no sistema que será desenvolvido, tem importância expressiva. Podemos perceber que pouquíssimos *frameworks* possuem algum suporte diretamente no *framework*. Não foram analisados componentes de terceiros por se tratar de uma questão estrutural e de vital importância.

Somente o *Zend Framework* possui uma classe específica para lidar com arquivos pdf; no restante deles, a manipulação dos arquivos é feita diretamente pelos comandos básicos ou então por classes feitas pela comunidade.

Todos os *frameworks* suportam a execução em lote de *scripts*, o que possibilita a geração de relatórios extensos, evitando o travamento de navegadores e até mesmo do servidor. Através dessa técnica, é possível agendar os relatórios e simplesmente fazer o *download* do arquivo em pdf para futura impressão.

Nenhum dos *frameworks* disponibilizou ferramentas para o design gráfico de relatórios ou auxiliar para impressão. Dessa maneira, a paginação e a disposição dos itens no relatório devem ser feitas manualmente.

5.8 SUPORTE E DOCUMENTAÇÃO

QUADRO 8: COMPARAÇÃO ENTRE AS FORMAS DE SUPORTE E QUALIDADE DA DOCUMENTAÇÃO DOS *FRAMEWORKS*

MÉTODO	***CAKEPHP***	***CODEIGNITER***	***SYMFONY***	***YII***	***ZEND FRAMEWORK***
Fórum	V	V	V	V	V
Lista de email	V	V	V	V	V
Documentação	V	V	V	V	V
Documentação em português	V	-	-	-	-

Além de prover uma infraestrutura adequada ao sistema, é desejado que o *framework* disponibilize formas de contato com o desenvolvedor, de forma a sanar eventuais dúvidas relacionadas ao funcionamento do mesmo. Todos os *frameworks* analisados anteriormente possuem, em seu *site* oficial, as ferramentas fórum, lista de *e-mail* e uma documentação de fácil acesso.

Com essas ferramentas é possível avançar ainda mais no desenvolvimento, seja na utilização de melhores praticas, reaproveitamento de código ou na solução de problemas relacionados ao *framework*.

O único *framework* que disponibilizou de forma oficial uma documentação em português foi o *CakePHP*. De certa forma isso não chega a ser um problema, visto que existem diversos *site*s dedicados a cada um dos *frameworks* acima, todos com um conteúdo bem diversificado e amplo. Além disso, na área de desenvolvimento de sistemas é praticamente obrigatório o conhecimento da língua inglesa.

5.9 DISCUSSÃO

Após a análise dos *frameworks*, levando em consideração todos os quesitos anteriormente mencionados, concluímos que o *framework* mais indicado para a utilização no ambiente proposto sejam os *frameworks* de maior porte, com maior quantidade de recursos relacionados a sistemas empresariais. Embora os frameworks menores possuam diversas ferramentas que facilitam o desenvolvimento, e uma documentação mais acessível, *frameworks* focados basicamente no desenvolvimento de páginas *web* foram deixados de lado, são eles *CakePHP* e *CodeIgniter*.

As funcionalidades presentes nos frameworks restantes (*Symfony, Yii* e *Zend Framework*) são semelhantes. Todos eles possuem diversos recursos importantes para o ambiente corporativo, como a execução de *scripts* em *shell*, embora também deixem a desejar em alguns aspectos como auxiliadores no desenvolvimento de relatórios.

Mesmo diante dessas questões, é possível afirmar que o *framework* mais indicado para o desenvolvimento de um sistema para controle sindical seja o *Zend Framework*. Como foi mostrado anteriormente, ele possui uma ligeira vantagem em relação aos outros *frameworks* na impressão e emissão de relatórios, assim

como possui uma instalação relativamente fácil. Vale lembrar, que a confiabilidade do *Zend Framework* pode ser comprovada diante dos grandes parceiros tecnológicos, dentre eles *IBM, Google* e *Microsoft*.

6 DESENVOLVIMENTO E IMPLANTAÇÃO

Após a difícil escolha de um *framework*, passamos para a fase de desenvolvimento. Deve ser previsto o quadro de profissionais envolvidos, bem como a divisão das tarefas. Logo de início, a maior dificuldade será o aprendizado do novo *framework*. Analistas com experiência em programação orientada a objetos terão um pouco mais de facilidade. Já os que trabalham com programação dividida em camadas, como o MVC, terão o prazo de aprendizagem certamente diminuído.

Com uma equipe de desenvolvimento reduzida, e uma alta diversidade de tarefas desempenhadas, serão atribuídas tarefas diárias visando facilitar a divisão do trabalho entre os analistas. Tendo em mente o tempo das tarefas, e junto a isso um detalhamento de todas as tarefas a serem terminadas, será possível prever o prazo para término do sistema (QUADROS, 2002). Para o sistema proposto, estima-se um prazo aproximado de um ano e meio para o término, tempo necessário para o desenvolvimento por dois analistas, além de testes e implantação.

Para a instalação do sistema não haverá grandes dificuldades, pois a seleção do *framework* já levou em conta o atual quadro de servidores. Sendo necessários somente pequenos ajustes em extensões e permissões em pastas.

A introdução do sistema ao usuário será feita de maneira gradativa. Na fase final de testes de cada módulo será apresentado ao usuário final para homologação. Após esse teste, o próximo módulo será desenvolvido e testado da mesma maneira.

Como dificuldades para o desenvolvimento teremos o baixo suporte para a impressão, principalmente de relatórios. Onde não foram encontradas soluções viáveis e robustas para se facilitar a criação e impressão dos mesmos. Pequenas alterações no *framework* também são previstas, porém dependendo do *framework* selecionado será possível efetuá-las sem maiores dificuldades.

7 CONCLUSÃO

Com a escolha em se utilizar um *framework*, passamos a seguir um padrão. Padrão esse que já foi exaustivamente testado, e geralmente seguem regras de boas práticas em programação. Todos os *frameworks* utilizam linguagem PHP, que permite um bom desempenho e segurança razoável, se adotadas as medidas de segurança necessárias.

Por questões de legibilidade de código e principalmente para facilidade de manutenção, é utilizada a orientação a objetos em todos os *frameworks* que, junto com a divisão em camadas (modelo, visão e controle) torna a manutenção do sistema mais simples e clara, além de possibilitar o trabalho com diversos profissionais ao mesmo tempo.

A escolha de um *framework* deve ser muito bem baseada, pois mesmo se tratando de *frameworks* semelhantes, utilizando a mesma linguagem e metodologias, existem diversos parâmetros diferentes. A análise do sistema que será desenvolvido deve ser detalhada, pois é a partir desse detalhamento que será possível definir qual *framework* atenderá melhor as necessidades da empresa.

Por fim, é válida a utilização de um *framework* pois, através dele, nos forçamos a seguir uma série de regras, que no começo serão difíceis e trabalhosas de se aprender, mas em longo prazo vão fazer com que o desenvolvimento seja cada vez mais ágil e seguro.

REFERÊNCIAS

Cake Software Foundation. **CakePHP**: *the rapid development PHP framework*. http://cakephp.org/ . 2013. (Acessado em outubro de 2013).

Cake Software Foundation. ***Cookbook Documentation*. Release 2.x** http://book.cakephp.org/2.0/_*downloads*/en/CakePHPCookbook.pdf. 2013. (Acessado em outubro de 2013).

EllisLab, Inc. *CodeIgniter* / EllisLab. ***A Fully Baked PHP Framework***. http://ellislab.com/codeigniter/ . 2013. (Acessado em outubro de 2013).

ExpressionEngine Development Team. ***CodeIgniter User Guide Version 2.1.4*** http://ellislab.com/codeigniter/user-guide/. 2013. (Acessado em outubro de 2013).

FAYAD, M. E., SCHMIDT, D. C. *Object-oriented Application frameworks*. **Communications of the ACM,** 1997

GANE, C.; SARSON, T. **Análise estruturada de sistemas**. Rio de Janeiro: Livros Técnicos e Científicos Editora S. A., 1983.

LAKATOS, E. M.; MARCONI, M. A. **Metodologia científica**. São Paulo: Atlas, 1986.

QUADROS, M. **Gerência de projetos de *software***: técnicas e ferramentas. Florianópolis: Visual Books Editora, 2002.

Sensio Labs. *Symfony,* ***High Performance PHP Framework for Web Development*.** http://symfony.com/ . 2013. (Acessado em outubro de 2013).

Sensio Labs. ***The Book for Symfony 2.3*** http://symfony.com/pdf/*Symfony*_book_2.3.pdf?v=4 . 2013. (Acessado em outubro de 2013).

The Apache Software Foundation. ***The Apache HTTP Server Project*** http://httpd.apache.org/docs/ . 2013. Acesso em: out. 2006.

The PHP Group. ***PHP Manual*** http://www.php.net/manual/en/ . 2013. (Acessado em outubro de 2013).

Yii Software LLC. ***Yii PHP Framework***: *Best for Web 2.0 Development* http://www.yiiframework.com/. 2013. (Acessado em outubro de 2013).

Yii Software LLC. *Yii PHP Framework*. ***The Definitive Guide to Yii*** http://www.yiiframework.com/doc/guide/. 2013. (Acessado em outubro de 2013).

Zend Technologies. ***Zend Framework 2***. *The most popular framework for modern, high-performing PHP applications*. http://www.zend.com/. 2013. (Acessado em outubro de 2013).

Zend Technologies. ***Programmer's Reference Guide of Zend Framework 2*** http://*framework*.zend.com/manual/2.2/en/index.html . 2013. (Acessado em outubro de 2013).