Principais aspectos do processo de desenvolvimento do AZEF: Um *framework* orientado a serviços para minimização de esforços de desenvolvimento

Thiago Pinheiro do Nascimento[1], Ricardo Gomes de Queiroz[2], Amélia Acácia de Miranda Batista[3]

[1]Bacharel em Ciência da Computação pelo Centro de Ensino Unificado de Teresina – CEUT. e-mail: thiagoanalistapi@gmail.com
[2]Mestre em Ciência da Computação pela Universidade Federal de Pernambuco – UFPE. e-mail: rgqueiroz@gmail.com
[3]Mestre em Engenharia Elétrica pela Universidade Federal do Maranhão – UFMA. e-mail: ameliabatista@hotmail.com

Resumo: *Frameworks* tradicionais de desenvolvimento ainda que demonstrem grande potencial tecnológico, apresentam problemas devido sua alta complexidade de implementação e utilização. Contudo, com o surgimento da arquitetura orientada a serviços (SOA), esses problemas podem ser minimizados, pois SOA baseia-se em princípios como: manutenabilidade, reusabilidade e simplicidade. O presente estudo visa abordar os principais aspectos do processo de desenvolvimento do AZEF, um *framework* baseado em SOA desenvolvido com o objetivo de tornar o processo de desenvolvimento de uma aplicação web menos complexo, pois sua utilização usualmente independe de codificação de sistema.

Palavras–chave: arquitetura orientada a serviços, engenharia de *software* baseada em componentes, *framework*, reuso de *software*

1. INTRODUÇÃO

A busca por modelos que propõem o desenvolvimento rápido de aplicações tem sido alvo de constantes estudos acerca da Engenharia de Software. Esse fato é resultado da necessidade de uma produtividade cada vez maior e da crescente demanda por aplicações para os mais variados problemas. Assim, as organizações inseridas no mercado de *software* estão preocupando-se, cada vez mais, com a redução de tempo no que tange o desenvolvimento dos seus produtos, isto associado ao aumento da qualidade dos mesmos. Nesse contexto, muitas foram as metodologias de desenvolvimento propostas, dentre estas estão inseridos os *frameworks*.

Segundo (TALIGENT, 1997), *frameworks* ou arcabouços de *software* são conjuntos de classes de *software* semiprontas e relacionadas que visam atender a solução de um determinado domínio de problema, de forma que as classes possam ser usadas, estendidas ou customizadas por desenvolvedores para soluções específicas.

Embora não haja dúvidas quanto aos benefícios que os *frameworks* tradicionais podem acrescentar aos projetos de *software*, sua usual complexidade de utilização e implementação surge como um problema que tem, cada vez mais, motivado as organizações a contratarem profissionais especialistas em *frameworks* (SOMMERVILLE, 2011).

De acordo com (CARNEIRO, 2003), essa alta complexidade inerente aos *frameworks* tradicionais, deve-se ao fato de sua manipulação depender em grande parte de extensas codificações de sistema, exigindo do desenvolvedor um alto grau de conhecimento sobre as implementações internas do *framework* em questão.

O presente estudo aborda os principais aspectos do processo de funcionamento e desenvolvimento do AZEF (*Almost Zero Effort Framework*), um *framework* orientado a serviços, capaz de provê uma redução significativa no que diz respeito aos esforços de desenvolvimento de uma miríade enorme de aplicações web, pois seu funcionamento independe de codificação de sistema. Ao invés disso, sua manipulação dar-se a partir do cadastro de funcionalidades, tornando o processo de desenvolvimento de *software* menos complexo se comparado à utilização de um *framework* tradicional.

Este artigo está organizado ao longo de seções, onde são abordados conceitos relacionados aos *frameworks* e à arquitetura orientada a serviços, características do AZEF como um modelo orientado a serviços, aspectos do processo de desenvolvimento do AZEF (análise de requisitos de *software*,

ISBN 978-85-62830-10-5

linguagem de programação e banco de dados utilizados, diagramação das entidades relacionais do banco de dados, padronização de projeto e validação de *software*) e conclusões.

2. *FRAMEWORK*

De acordo com (SOMMERVILLE, 2011), a concepção de *frameworks* deu-se ao final da década de 1990, graças à transição gradual do desenvolvimento de *software* original para o desenvolvimento baseado em reuso, uma estratégia de construção de aplicações que objetiva o ganho de produtividade, através da reutilização de unidades previamente criadas e que posteriormente deu lugar a engenharia de *software* baseada em componentes (CBSE – *Component-Based Software Engineering*).

A CBSE é uma atividade com um peso crescente na indústria de *software*. Esse crescimento é justificado economicamente, pois é encarado como uma abordagem que permite reduzir custos no processo de desenvolvimento de *software*, na medida em que, diminui o tempo empregado nesse processo, aumentando a qualidade do mesmo (GOULÃO, 2002).

Segundo (SOMMERVILLE, 2011), *frameworks* são conjuntos de classes abstratas e concretas que podem ser adaptadas, a fim de que suas funcionalidades possam ser ampliadas para criação de sistemas de aplicações. Sua principal característica é o reuso de *software*, a fim de reduzir custos no processo de desenvolvimento de sistemas, facilitar o acoplamento de novas funcionalidades, ou a mudança de funcionalidades já existentes.

3. ARQUITETURA ORIENTADA A SERVIÇOS

A arquitetura de *software* é um dos principais responsáveis por proporcionar ganhos efetivos quando se trata de agilidade e eficiência na manutenção e evolução dos sistemas de informação corporativos (SORDI; MARINHO; NAGY, 2006). Seu conceito refere-se a um modelo estrutural e organizacional que pode afetar fatores críticos (desempenho, segurança, manutenção etc.) associados ao funcionamento de uma aplicação.

SOA é uma analogia ao termo em inglês: *Service-Oriented Architecture*, que significa arquitetura orientada a serviços. Sua definição refere-se a um padrão que descreve os principais conceitos de uma arquitetura de *software* e suas relações, onde o serviço e sua utilização são os conceitos fundamentais implícitos (FONSECA, 2009).

Embora essa definição tenha sido vastamente abordada pelos principais autores da Engenharia de Software, ainda há confusões que dificultam o seu entendimento. SOA não deve ser entendido como um *software*, tecnologia ou até mesmo um produto, mas sim como um conceito relacionado à forma de projetar sistemas baseados na composição de serviços interoperáveis e reutilizáveis (JUNIOR, 2010).

4. AZEF: UM MODELO DE *FRAMEWORK* ORIENTADO A SERVIÇOS

Frameworks orientados a serviços podem ser definidos como um modelo de *software* que une conceitos da engenharia de *software* baseada em componentes e da arquitetura orientada a serviços para a obtenção de produtos de *softwares* reutilizáveis (FONSECA, 2009).

O principal objetivo do presente estudo é apresentar o AZEF, um *framework* orientado a serviços cujo objetivo é minimizar esforços no processo de desenvolvimento de aplicações *web*, ou seja, na criação de *web softwares* que trabalhem com dados que devem persistir em banco de dados.

A ideia para o nome AZEF deu-se a partir das iniciais ao termo em inglês *Almost Zero Effort Framework*, ou simplesmente *framework* de esforço quase zero. Essa definição deve-se ao fato de que o AZEF não tem natureza orientada a código-fonte, ou seja, o desenvolvimento de aplicações a partir dele só necessitará de desenvolvimento de código de sistema, em casos particulares.

Na verdade, o modelo de *framework* que propusemos e desenvolvemos torna-se bastante diferenciado, pois suas características são baseadas em serviços e, por isso, parte da sua manipulação depende exclusivamente de cadastros de funcionalidades.

ISBN 978-85-62830-10-5

Para um melhor entendimento sobre o funcionamento do AZEF, vamos imaginar uma ferramenta de interfaces bem interativas com um desenvolvedor de aplicação (usuário), que a partir de simples cadastros de serviços, possa mapear os dados para diferentes bancos de dados, interfaces e validações, e que, além disso, torne-o flexível para desenvolver novas funcionalidades. O AZEF enquadra-se perfeitamente nesse estilo de ferramenta.

De acordo com (MACHADO, 2004), a implementação dos *frameworks* orientados a serviços possibilita um reuso bem mais amplo do que os *frameworks* tradicionais, pois suas funcionalidades devem basear-se em configurações cada vez menores. Baseado nesse princípio, o processo de desenvolvimento do AZEF foi focado reunir conceitos capazes de tornar esse mesmo *framework* obediente à configuração de um número mínimo de parâmetros. Além disso, o AZEF conta com algumas características que, de acordo com (MACHADO, 2004) são fundamentais aos *frameworks* orientados a serviços:

a) Disponibilidade de reuso caixa-branca e caixa-preta: capacidade de viabilizar o desenvolvimento de novas funcionalidades de *software* a partir de unidades previamente criadas ou a partir de novas implementações a critério do desenvolvedor (RAVICHANDRAN; ROTHENBERGER, 2003);

b) Dinamismo: capacidade de ter suas funcionalidades e serviços passíveis de alteração, ou seja, o acoplamento de novas funcionalidades ou alteração de funcionalidades existentes não afeta o funcionamento da aplicação como um todo (MACHADO, 2004);

c) Heterogeneidade ambiental: capacidade de comunicação com diversas plataformas de sistemas operacionais e sistemas de banco de dados. Atualmente, o AZEF provê comunicação com os principais sistemas operacionais (Linux, Windows, Mac OS, FreeBSD etc.) e sistemas de banco de dados (MySQL, PostgreSQL, mSQL, Oracle, dbm, InterBase, FilePro etc.);

d) Padronização: familiarização das funcionalidades existentes com o desenvolvedor de aplicações, de modo que essa familiarização seja um fator importante que influencia na redução de erros por parte dos desenvolvedores de aplicação.

5. PROCESSO DE DESENVOLVIMENTO DO AZEF

O desenvolvimento de *software*, ao contrário do senso comum relativo a esse tópico, não corresponde apenas à etapa de implementação de código de sistema, mas sim a todo o conjunto de atividades, parcialmente ordenadas, cuja finalidade é a obtenção de um produto de *software*. Essas atividades compreendem, desde a extração dos requisitos desejados, até a validação e testes dos mesmos (SOMMERVILLE, 2011).

Em relação ao processo de desenvolvimento do AZEF, fez-se importante o cumprimento de uma série de etapas: análise de requisitos de software, definição da linguagem de programação e do banco de dados a serem utilizados, elaboração do diagrama de entidades relacionais, padronização de projeto e validação. Cada uma delas será amplamente abordada no decorrer do nosso trabalho.

5.1 ANÁLISE DE REQUISITOS DE *SOFTWARE*

A análise de requisitos de *software* compreendeu a primeira atividade do processo de desenvolvimento do AZEF. Nessa fase, foram identificadas as principais funcionalidades fornecidas pelos *frameworks* tradicionais, para que posteriormente, fosse elaborado um conjunto de procedimentos que tornassem viável a implementação das mesmas sem a presença de extensas codificações de sistema, mas sim através de cadastros de funcionalidades.

Atualmente, as funcionalidades desempenhadas pelos *frameworks* tradicionais ou não orientados a serviço, são baseadas no funcionamento de três camadas: interatividade do sistema com o usuário, aplicabilidade de regras de negócios e controle de dados.

5.2 LINGUAGEM DE PROGRAMAÇÃO E BANCO DE DADOS UTILIZADOS

O AZEF é um *framework* desenvolvido utilizando a linguagem PHP juntamente com o banco de dados MySQL.

ISBN 978-85-62830-10-5

O PHP é uma linguagem de criação de *scripts* do lado do servidor, que foi projetada especificamente para a web. Seu surgimento deu-se em 1994, através de Rasmus Lerdorf. As principais motivações para adoção do PHP na criação do nosso *framework* são justificadas por (WELLING; THOMSON, 2005) de acordo com os seguintes quesitos: alto desempenho, interfaces para muitos sistemas diferentes de banco de dados e portabilidade.

De acordo com (WELLING; THOMSON, 2005), o MySQL é um sistema de gerenciamento de banco de dados relacional (RDBMS – *Relational DataBase Management System*) bastante poderoso e eficiente. Suas principais vantagens são: alto desempenho, baixo custo e facilidade de configurar e aprender.

5.3 DIAGRAMAÇÃO DAS ENTIDADES RELACIONAIS

Uma das principais atividades do processo de desenvolvimento do AZEF compreendeu a modelagem e construção do diagrama das entidades relacionais do banco de dados.

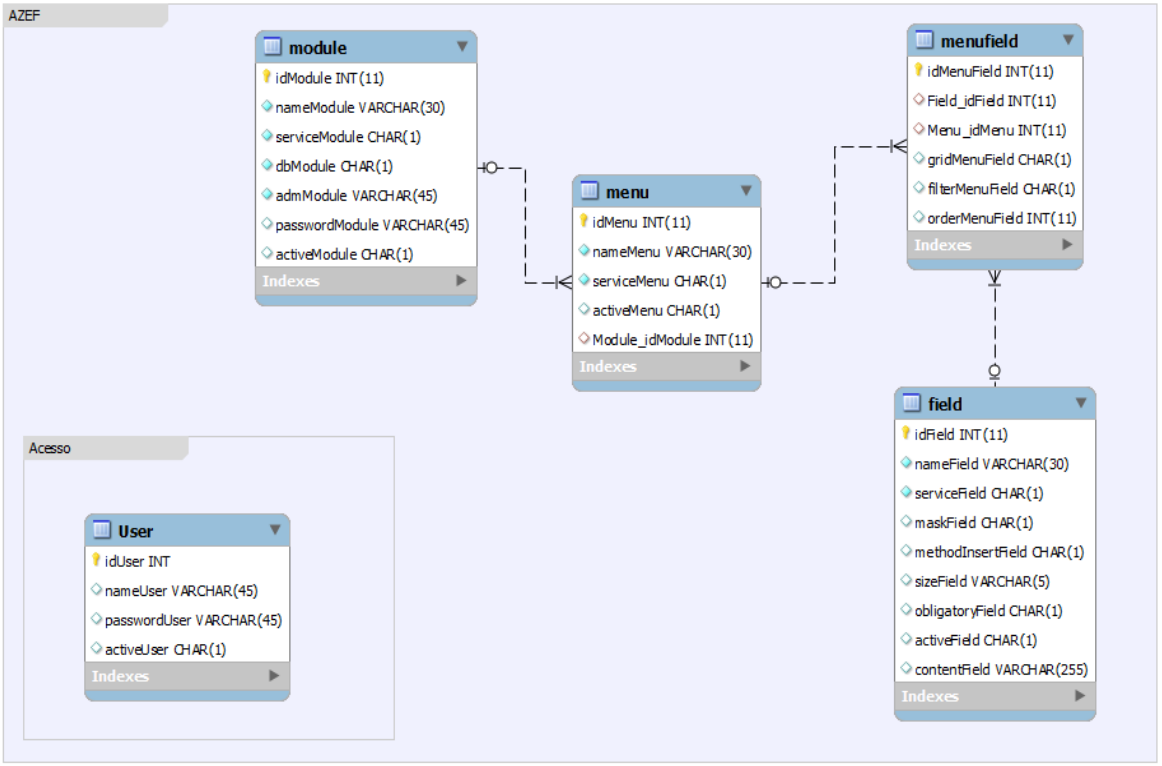

Figura 1 – Diagrama de entidades relacionais do banco (MySQL) de dados do AZEF

A tabela *User* contém os dados referentes aos usuários que terão acesso aos serviços do AZEF. As demais entidades relacionais (*module*, *menu*, *field* e *menufield*) representam os quatro serviços providos pelo AZEF:

a) Modulo (*module*): compreende as configurações iniciais do *software*, onde o desenvolvedor especificará se desenvolverá a aplicação a partir de uma base de dados previamente criada ou não. Além disso, será especificado o banco de dados a ser utilizado pela aplicação e informações para acesso ao banco de dados (usuário e senha);

b) Menu(*menu*): compreende à criação das tabelas e dos formulários que farão parte da aplicação. Ao cadastrar um menu, o AZEF automaticamente criará a tabela e o formulário para a figuração e cadastro dos dados respectivamente;

c) Campo (*field*): compreende a configuração de campos para os formulários da aplicação. Ao criar um campo, o desenvolvedor torna esse mesmo campo apto a figurar em qualquer menu da aplicação;

ISBN 978-85-62830-10-5

d) Vinculação de campo (*menufield*): compreende a ação de vincular um campo a um menu. Para que um campo (*field*) recém-criado figure em um formulário (*menu*), faz-se necessário que o mesmo seja vinculado na tabela *menufield*. Para cada campo vinculado, automaticamente seu respectivo atributo de tabela é inserido na entidade do menu especificado.

5.4 PADRÃO DE PROJETO

O padrão de projeto utilizado na criação do AZEF foi o MVC – um acrônimo para *Model*, *View* e *Controller* (Modelo, Visão e Controlador). Esse padrão de projeto tem como objetivo separar todo o desenvolvimento de uma aplicação em três partes:

a) Modelo: gerencia o comportamento dos dados da aplicação;
b) Visão: gerencia a saída gráfica e textual da parte da aplicação visível ao usuário;
c) Controlador: interpreta as entradas de dados, comandando a Visão e o Modelo;

A grande vantagem de se utilizar o padrão MVC é a separação de lógica e apresentação, sendo que isso favorece o trabalho em equipe. Um *designer* poderia trabalhar na apresentação, definindo HTML, CSS e *Flash*, enquanto um *Database Administrator* (DBA) poderia trabalhar com o modelo, e outro programador poderia se concentrar nas regras de negócios inseridas no controlador. Dessa forma, qualquer mudança, por exemplo, na apresentação, teria pouco ou nenhum impacto nas demais camadas de aplicação (MINETTO, 2007).

5.5 VERIFICAÇÃO E VALIDAÇÃO DE TESTES

A última fase do processo de desenvolvimento do AZEF compreendeu a verificação e validação de *software*. Essa atividade contou com quatro etapas: planejamento de testes, projeto de casos de testes, execução de testes e avaliação dos resultados de testes. Cada uma delas foi desenvolvida ao longo do próprio processo de desenvolvimento de *software*, concentrando-se em testes de unidade, testes de integração e testes de sistema.

Os testes de unidade concentraram-se na procura de erros de lógica e de implementação em cada módulo de *software*, separadamente. Os testes de integração compreenderam as atividades de descoberta de erros associados à comunicação entre os serviços do AZEF. Por fim, nos testes de sistema, foram identificados erros de funções e características de desempenho que não estavam de acordo com a especificação de *software*.

6. CONCLUSÕES

Frameworks orientados a serviços surgem como soluções capazes de minimizar gargalos inerentes ao uso dos *frameworks* tradicionais. Essa afirmação deve-se ao fato de que os *frameworks* orientados a serviço independem ao máximo de codificações de sistema, o que não ocorre nos *frameworks* tradicionais.

De acordo com a proposta apresentada no presente estudo, o processo de prototipação de *software* passará a ser familiar tanto ao desenvolvedor de aplicação quanto ao seu usuário final principalmente, para este último, no que tange ao processo de construção das aplicações em si.

REFERÊNCIAS

CARNEIRO, C. **Frameworks de Aplicações Orientadas a Objetos – Uma Abordagem Interativa e Incremental**. 2003. Dissertação (Mestrado em Redes de Computadores) – Universidade de Salvador, Salvador, 2003.

FONSECA, J. S. **Frameworks de Aplicações Moveis com Segurança em SOA**. 2009. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal do Maranhão, São Luis, 2009.

GOULÃO, M. C. P. A.. **Engenharia de Software Baseado em Componentes: uma Abordagem Quantitativa**. Dissertação de Mestrado, Universidade Nova de Lisboa, Caparica, 2002.

ISBN 978-85-62830-10-5

JUNIOR, J. D. **Arquitetura Orientada a Serviços**, Em: Revista Engenharia de Software, 22. ed. Disponível em http://www.devmedia.com.br/revista-engenharia-de-software-22/16114. Acesso em 03 de março de 2012

MACHADO, J. C. **Um Estudo Sobre o Desenvolvimento Orientado a Serviços**. 2003. Dissertação (Mestrado em Ciência da Computação) – Pontifica Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2004.

MINETTO, E. L. **Framework Para Desenvolvimento em PHP**. 2007.

RAVICHANDRAN, T.; ROTHENBERGER, M. A. ***Software Reuse Strategies and Component Markets***, *Association for Computing Machinery*, *August*, 2003.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Addison-Wesley, 2011.

SORDI, J. O.; MARINHO, B. L.; NAGY, M. **Benefícios da Arquitetura de Software Orientada a Serviços para Empresas: Análise da Experiência do ABN AMRO Brasil**. Em Revista de Gestão da Tecnologia e Sistemas de Informação, 2006.

TALIGENT, Inc. ***Building Object-Oriented Frameworks***.2011.

WELLING, L.; THOMSON, L. ***PHP & MySQL Web Development***. 3.ed. 2005.

ISBN 978-85-62830-10-5