Guiwe: um Framework para Construção da Camada de Interface Gráfica Web de Sistemas e de Frameworks

Matheus C. Viana[1,2], Ricardo A. Ramos[3], Rosangela A. D. Penteado[2]

[2]Departamento de Computação (DC) - Universidade Federal de São Carlos (UFSCar)
Caixa Postal 676 – 13565-905 – São Carlos – SP – Brasil
[3]Centro de Informática - Universidade Federal de Pernambuco (UFPE)
Cidade Universitária - 50740-540 - Recife, Pernambuco, Brasil
[3]Colegiado de Eng. da Computação - UNIVASF
Santo Antônio - 48902-300 -Juazeiro, Bahia, Brasil
matheus_viana@dc.ufscar.br, ricargentonramos@gmail.com, rosangel@dc.ufscar.br

Abstract. The use of frameworks for system instantiation is desirable, since it provides code and design reuse and decreases the effort spent on software development. However, enterprise application frameworks which do not have a graphical user interface layer demand extra effort to implement this layer during a system instantiation. This paper presents Guiwe framework (Graphical user interface for web)**,** which supports the development of a graphical user interface layer for systems and enterprise application frameworks. Guiwe framework architecture uses web technologies and applies Model-View-Controller (MVC) pattern. A case study was performed with an instantiation of Guiwe framework in order to develop the graphical user interface layer of GRENJ framework.

Palavras-chave: Framework, Guiwe, camada de interface gráfica.

1 Introdução

A camada de interface gráfica requer grande parte do esforço gasto para o desenvolvimento de alguns sistemas. Esse esforço se deve tanto à sua construção quanto à sua interligação com a camada de negócios (modelo). Alguns ambientes de desenvolvimento integrado (do inglês, *Integrated Development Environment* – IDE) possuem recursos que apóiam a construção da camada de interface gráfica de sistemas, com o intuito de facilitar o trabalho do desenvolvedor. Entretanto, as IDEs geram códigos complexos, dependentes de bibliotecas próprias e não estimulam o desenvolvedor a implementar o sistema com base em técnicas que objetivam a qualidade como, por exemplo, padrões de software [1].

O uso de algumas técnicas baseadas em reúso pode reduzir o esforço de desenvolvimento e, simultaneamente, aumentar a qualidade do software produzido. O conceito de reúso consiste na reutilização de artefatos previamente construídos e testados [2].

[1] Apoio financeiro CNPq.

Frameworks orientados a objetos são soluções reutilizáveis em que entidades abstratas são customizadas com o propósito de definir um comportamento específico [3]. Além de reúso de código, frameworks fornecem reúso de projeto. Um *Enterprise Application Framework* (EAF) é utilizado para apoiar o desenvolvimento sistemas de domínio específico, voltado a atividades de indústria ou de prestação de serviços como, por exemplo, sistemas para aeroportos, manufatura, e gestão de negócios [4].

Este artigo tem como objetivo apresentar o framework Guiwe (do inglês, ***G****raphical* ***u****ser* ***i****nterface for* ***we****b*), que apóia a construção da camada de interface gráfica de sistemas voltados para a web. Esse framework também pode ser instanciado para dar origem à camada de interface gráfica de um EAF, devido a sua capacidade de adaptação aos pontos variáveis do EAF em tempo de execução.

Além desta Seção, as demais estão organizadas da seguinte maneira: na Seção 2 é abordado o desenvolvimento de software com o uso de frameworks orientados a objetos; na Seção 3 é apresentado o processo de desenvolvimento do framework Guiwe, bem como são descritas suas principais características; na Seção 4 é apresentado um estudo de caso em que o framework Guiwe foi utilizado para apoiar a construção da camada de interface gráfica de um EAF; na Seção 5 são abordados alguns trabalhos relacionados; e na Seção 6 são apresentadas as considerações finais.

2 Frameworks Orientados a Objetos

Frameworks são constituídos de um conjunto de classes, assim como uma biblioteca de classes, porém as classes de um framework são dependentes entre si e não podem ser utilizadas separadamente como as de uma biblioteca [5]. Isso se deve, pois, além de reúso de código, frameworks também fornecem reúso de projeto. Outra diferença existente entre biblioteca de classes e frameworks é quanto à ordem de controle do software. O código específico da aplicação é que determina quando e como as classes de uma biblioteca são utilizadas. Contudo, quando um framework é utilizado, ocorre uma inversão de controle, pois é o framework que invoca o código específico da aplicação. Essa inversão de controle é conhecida como Princípio de Hollywood [6].

Todo framework é composto de duas partes [7]: os **pontos fixos** (*frozen spots*), que constituem a parte imutável e independente da aplicação em desenvolvimento e os **pontos variáveis** (*hot spots*), que representam a parte utilizada pelo desenvolvedor para instanciá-lo de acordo com as especificações da aplicação. Os pontos variáveis são customizados por meio de **métodos gancho** (*hook methods*).

Um framework no qual o acesso aos pontos variáveis ocorre por meio da herança de suas classes é classificado como **caixa branca**. Quando esse acesso se dá por meio de composição, é classificado como **caixa preta**. Um framework que pode ser instanciado das duas maneiras é classificado como **caixa cinza**. Outra classificação indica o seu propósito [4]: 1) *Enterprise Application Frameworks* (EAF), 2) *System Infrastructure Frameworks* (SIF) e 3) *Middleware Integration Frameworks* (MIF). SIF simplificam o desenvolvimento de softwares que controlam operações de baixo nível como, por exemplo, sistemas operacionais, gerenciadores de janelas gráficas e compiladores. MIF aumentam a capacidade de modularização e integração de aplicações distribuídas. *Struts* [8] e *Spring* [9] são exemplos de MIF.

3 O Framework Guiwe

Com o crescimento do uso de sistemas web, há a necessidade de construir interfaces gráficas que possam ser reutilizadas para reduzir o esforço de desenvolvimento desses sistemas. Dessa forma, o framework Guiwe foi concebido para ser utilizado na construção da camada de interface gráfica web de aplicações e de EAFs.

O framework Guiwe é um MIF caixa branca com duas camadas: de interface gráfica e de controle. A primeira tem o objetivo de prover uma interface gráfica web e a segunda realiza a interligação entre essa interface e a camada de negócios do sistema ou EAF. Assim, o framework Guiwe compõe uma arquitetura baseada no padrão *Model-View-Controller* (MVC) [10].

A interface gráfica fornecida pelo framework Guiwe é organizada em uma tela com cabeçalho, menu principal, painel principal e rodapé. No cabeçalho e no rodapé e no painel principal podem ser carregadas páginas web com imagens e textos. O menu principal contém as opções de páginas que podem ser carregadas no painel principal. A organização da interface gráfica criada pelo framework Guiwe é apresentada na Figura 1 (a). No painel principal da interface gráfica podem ser carregados formulários, tabelas, relatórios e outras páginas do sistema em desenvolvimento, Figura 1 (b). Cada formulário está relacionado com uma classe da camada de negócios do sistema em desenvolvimento e os campos desse formulário correspondem aos atributos dessa classe.

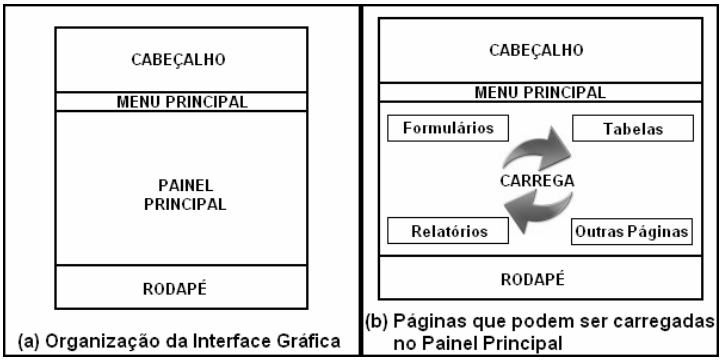

Figura 1. Organização da interface gráfica do framework Guiwe.

Os painéis representam classes que mantém um relacionamento de composição ou agregação com outras e, portanto, não possuem um formulário próprio. Um exemplo desse tipo de relacionamento existe entre uma classe que representa uma venda e outra que representa os itens de uma venda. Também é possível construir tabelas e relatórios que carregam informações provenientes da base de dados da aplicação.

A camada de controle do framework Guiwe gerencia a execução da funcionalidade da aplicação ou do EAF em desenvolvimento por meio da manipulação de objetos das classes da camada de negócios. A camada de controle identifica e acessa os atributos dos objetos em tempo de execução, pois, de outra forma, não seria possível manipular objetos de classes que foram implementadas durante a instanciação de um sistema a partir de um EAF que utiliza o framework Guiwe.

3.1 Construção do Framework Guiwe

O desenvolvimento de sistemas voltados para a web engloba várias tecnologias diferentes como, por exemplo, Java, JSP, HTML, CSS e JavaScript. Além de exigir que se tenha conhecimento sobre essas tecnologias, são necessários esforços extras para organizá-las e interligá-las. No caso de um framework voltado para a web, além dessas dificuldades, existe a complexidade de customização dos seus pontos variáveis. Desse modo, na camada de interface gráfica do framework Guiwe foram implementadas classes para a geração do código HTML das páginas dos sistemas e *servlets* Java foram utilizados no lugar de JSP para que seja priorizado o uso da linguagem Java durante a instanciação desse framework. O uso das linguagens CSS e JavaScript não pôde ser completamente evitado na implementação do framework Guiwe, devido a particularidades dessas tecnologias e, também, para evitar restrições à flexibilidade da interface gráfica. Porém, documentos dessas tecnologias já são providos pelo framework para definir a aparência de suas páginas e para gerenciar os eventos do usuário. O desenvolvedor não necessita customizá-los a menos que queira adicionar alguma funcionalidade não prevista pelo framework ou alterar a apresentação da interface gráfica do sistema em desenvolvimento.

Na camada de interface gráfica, as classes que estendem *HTMLDocument* são responsáveis pela geração do código HTML das páginas web. Cada servlet utiliza uma dessas classes para criar um tipo de página: simples (*PageServlet*), formulário (*PersistentFormServlet*), tabela (*TableServlet*), painel (*PanelServlet*) ou relatório (*ReportServlet*). A Figura 2 apresenta o modelo de classes da camada de interface gráfica do framework Guiwe.

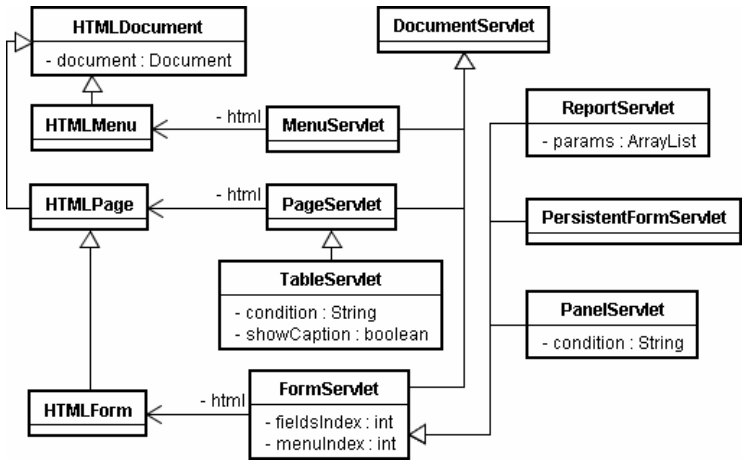

Figura 2. Modelo de classes da camada de interface gráfica do framework Guiwe.

Na camada de controle, a classe *ControllerServlet* recebe e responde às requisições provenientes da camada de interface gráfica. A classe *ControllerPerformer* identifica e efetua a operação indicada em cada requisição. Essas operações são efetuadas sobre objetos persistentes das classes da camada de modelo de um sistema desenvolvido com o apoio do framework Guiwe. As classes da camada de modelo do sistema

devem implementar a interface *PersistentObject*, que indica quais operações podem ser efetuadas pelo framework. As classes *PersistentObjectManager* e *ObjectManager* são responsáveis por inicializar e atribuir valores aos objetos persistentes das classes da camada de modelo para que as operações possam ser efetuadas. A Figura 3 apresenta o modelo de classes da camada de controle do framework Guiwe. As classes *ControllerPerformer* e *PersistentObjectManager* podem ser estendidas para que o framework Guiwe efetue novas operações sobre outros tipos de objetos.

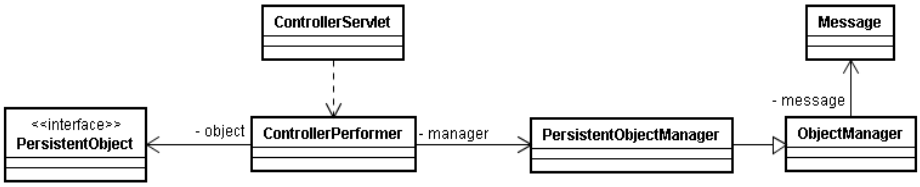

Figura 3. Modelo de classes da camada de controle do framework Guiwe.

As mensagens apresentadas aos usuários, quando uma operação é concluída ou quando ocorre uma falha na sua execução, são definidas em arquivos que podem ser customizados pelo desenvolvedor. Esses arquivos permitem a internacionalização da interface gráfica dos sistemas construídos com o apoio do framework Guiwe.

As classes do framework Guiwe foram construídas com a prática do *Test-Driven Development* (TDD) [11]. Essa prática define que os detalhes de implementação de um software podem ser determinados com a criação prévia de testes. Dessa forma, para cada classe do framework, os seguintes passos foram realizados: 1) criação de uma lista de casos de testes; 2) implementação dos testes; 3) implementação dos métodos da classe; 4) execução dos testes, correção e validação dos métodos. A criação da lista de casos de testes corresponde à criação da classe, de seus atributos, de seu *javadoc* e da assinatura de seus métodos. Os testes foram implementados em Java com o apoio da ferramenta JUnit [12].

Os códigos HTML gerados pelas classes do framework Guiwe definem a estrutura (conteúdo) das páginas web. Para a definição da apresentação (cores, fontes, etc.) dessas páginas foi necessária a criação de documentos CSS que mantêm a padronização da interface gráfica e auxiliam quanto à usabilidade dos sistemas.

Para o tratamento de eventos na interface gráfica do framework Guiwe foram criados quatro arquivos JavaScript: 1) o arquivo *jquery* [13] fornece uma interface para execução dos comandos assíncronos; 2) o arquivo *main* controla o funcionamento do menu principal e utiliza a interface do arquivo *jquery* para carregar o cabeçalho, o menu principal e o rodapé, assim como os formulários, as tabelas, os relatórios, a página inicial e outras páginas no painel principal; 3) o arquivo *persistentform* envia os dados de um formulário, painel ou relatório que estiver carregado na tela para a camada de controle efetuar uma operação, recebe a resposta e disponibiliza-a para o usuário; 4) o arquivo *transaction* realiza funções semelhantes às do arquivo *persistentform*, porém, específicas dos painéis.

4 Estudo de Caso: Construção da Camada de Interface Gráfica do Framework GRENJ

O framework Gestão de REcursos de Negócios com implementação em Java (GRENJ) [14] é um EAF caixa branca que pertence ao domínio definido pela linguagem de padrões de análise Gestão de Recursos de Negócios (GRN) [15]. Esse domínio está relacionado com sistemas que contemplam transações de aluguel, comercialização e manutenção de recursos de negócios. A arquitetura do framework GRENJ era composta de duas camadas: de persistência e de negócios (modelo). A camada de negócios possui uma hierarquia de classes que representam entidades do domínio do framework, por exemplo, a classe *Resource* que representa um recurso.

A construção da camada de interface gráfica do framework GRENJ foi realizada com a instanciação do framework Guiwe. Essa instanciação ocorreu com a construção de classes que dependem das classes da camada de negócios do framework GRENJ e estendem, direta ou indiretamente, classes das camadas de interface gráfica ou de controle do framework Guiwe. Por exemplo, foram construídas as classes *StaticObjectManager* e *StaticObjectFormServlet*. Ambas são dependentes da classe *StaticObject* da camada de negócios do GRENJ e herdam, respectivamente, as classes *PersitentObjectManager* e *PesistentFormServlet* do framework Guiwe. Essa hierarquia é semelhante a do framework GRENJ, cuja classe *StaticObject* estende *PersistentObject*. A Figura 4 apresenta o modelo de classes da camada de interface gráfica do framework GRENJ. As classes destacadas na cor cinza pertencem ao framework Guiwe e as demais são dependentes de classes da camada de negócios do framework GRENJ que possuem nomes e hierarquia semelhantes.

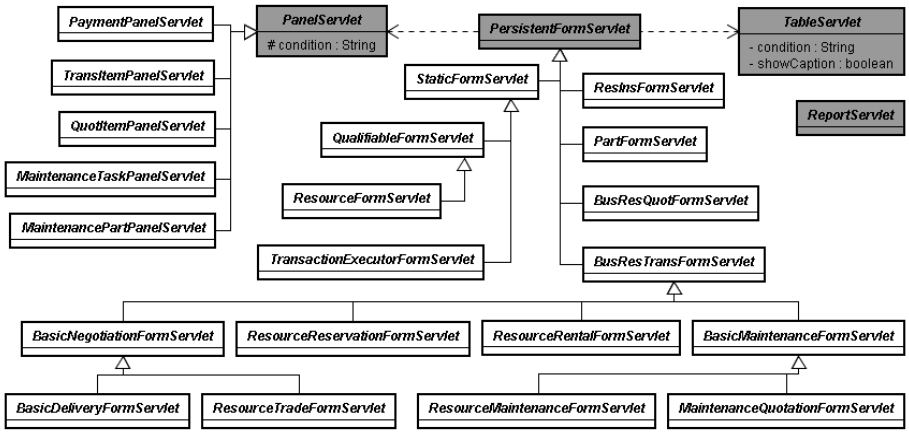

Figura 4. Modelo de classes da camada de interface gráfica do framework GRENJ.

A camada de controle do framework GRENJ é formada por um conjunto de classes que recebem e respondem às solicitações da camada de interface gráfica. Essas classes são responsáveis por manipular objetos das classes da camada de negócios com as quais estão relacionadas. A Figura 5 apresenta o modelo de classes da camada de controle do framework GRENJ.

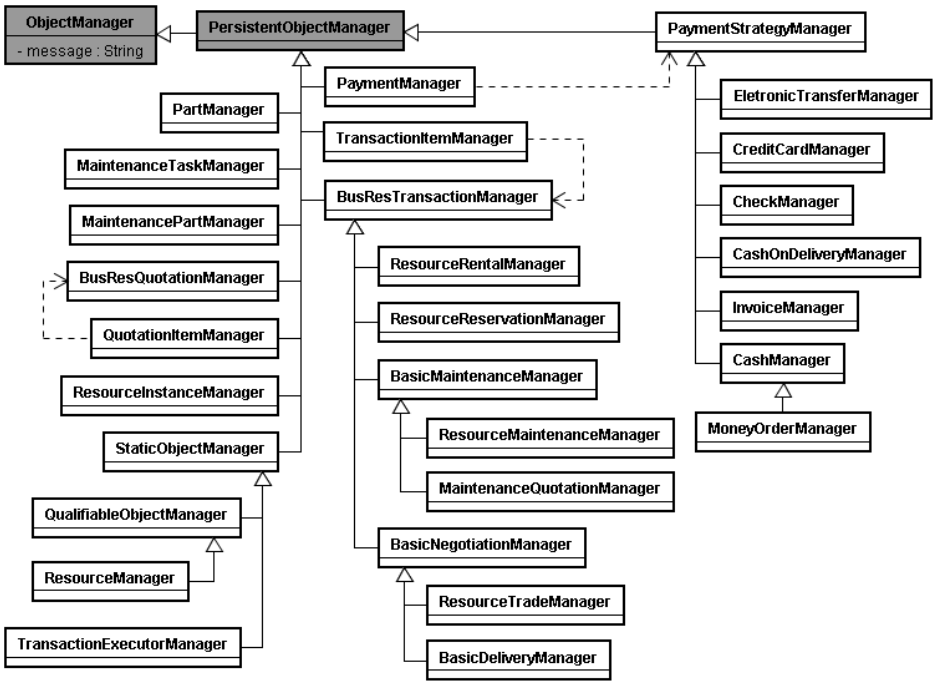

Figura 5. Modelo de classes da camada de controle do framework GRENJ.

4.1 Instanciação de um Sistema a Partir do Framwork GRENJ

Um sistema para uma locadora de DVDs foi desenvolvido com a instanciação do framework GRENJ e, consequentemente, do framework Guiwe. A Tabela 1 contém os requisitos desse sistema.

Tabela 1. Requisitos do sistema para uma locadora de DVDs.

#	**Descrição**
1	A locadora realiza o aluguel de DVDs de filmes que podem ter uma ou mais cópias.
2	Cada filme possui um código, título e ano.
3	Cada DVD possui código que identifica sua posição na prateleira, informação que indica se está disponível ou alugado e o título do filme nele contido.
4	Os filmes são classificados por gênero (comédia, terror, ação, etc.).
5	Os DVDs são alugados para os clientes cadastrados da locadora. As informações que o sistema deve manter sobre o cliente são: código, nome, telefone e CPF.
6	As informações de locação são: código, data de locação, data de devolução prevista, código do cliente, DVDs alugados, data de devolução efetiva e valor. Um cliente pode alugar mais de um DVD em uma mesma locação.

As classes do sistema para uma locadora de DVDs estendem classes das camadas de negócios e de interface gráfica do framework GRENJ para criar a lógica e a interface específica do sistema. Não é necessário estender as classes da camada de controle para a instanciação de sistemas a partir do framework GRENJ. A Tabela 2 contém a relação de classes específicas do sistema para uma locadora de DVDs.

Tabela 2. Classes específicas do sistema para uma locadora de DVDs.

Classes da Camada de Negócios	Classes da Camada de Interface Gráfica
Genero	GeneroFormServlet
Filme	FilmeFormServlet
DVD	DVDFormServlet, DVDTableServlet
Locacao	LocacaoFormServlet, LocacaoTableServlet
Cliente	ClienteFormServlet
ItemLocacao	ItemPanelServlet

O Formulário de Locações é gerado pela classe *LocacaoFormServlet*. Essa classe estende *ResourceRentalFormServlet*, que é responsável por gerar formulários para transações de aluguel. A Figura 6 apresenta a tela do sistema para uma locadora de DVDs com o Formulário de Locações carregado no painel principal, em que são destacados: a) a tabela de locações construída pela classe *LocacaoTableServlet* e b) o painel de itens da locação construído pela classe *ItemPanelServlet*.

Figura 6. Modelo de classes da camada de controle do framework GRENJ.

A construção da camada de interface gráfica do sistema para uma locadora de DVDs com o apoio do framework Guiwe foi realizada rapidamente, em torno de uma hora, somente com a criação de algumas classes que possuem alguns métodos gancho que indicam detalhes específicos do sistema. A construção da mesma camada em outra versão desse sistema, sem o apoio do framework Guiwe, foi realizada em maior tempo, em torno de oito horas.

5 Trabalhos Relacionados

O framework Fast Interface Build (FIB) foi desenvolvido com o intuito de apoiar a construção da camada de interface gráfica de sistemas implementados em linguagem Java [16]. A interface criada por esse framework fornece meios para efetuação de operações de persistência de dados, assim como o framework Guiwe. Entretanto, o framework FIB é baseado na biblioteca Swing da linguagem Java e, portanto, é capaz de criar interfaces somente para sistemas *stand-alone*. O framework Guiwe faz uso de tecnologias web e pode ser acessado remotamente por meio de um navegador.

Frameworks utilizados na indústria, como o *Struts* [8] e o *Spring* [9], atuam como intermediadores entre a camada de negócio e a de interface gráfica com o usuário em aplicações voltadas para a web. Apesar de apoiarem com sucesso o desenvolvimento de sistemas com requisitos específicos, esses frameworks apresentam limitações quando utilizados no desenvolvimento de softwares genéricos como, por exemplo, outros frameworks. Além disso, esses frameworks exigem do desenvolvedor conhecimento tanto de sua arquitetura quanto de tecnologias web. A arquitetura do framework Guiwe é simplificada, devido ao número reduzido de classes e, em geral, o desenvolvedor se baseia apenas na linguagem Java para utilizá-lo.

6 Considerações Finais

Este trabalho apresentou o framework Guiwe para a construção da camada de interface gráfica de sistemas de informação e de *Enterprise Application Frameworks* voltados para web. A instanciação desse framework define um processo em que algumas de suas classes são estendidas para contemplar as características específicas das classes da camada de negócios do software alvo. Esse processo favorece a aplicação de práticas relacionadas com métodos ágeis, como iterações incrementais, testes, entre outras.

O número reduzido de classes do framework Guiwe facilita sua aprendizagem e sua utilização. Apesar de sua simplicidade, sua característica caixa branca permite que suas classes sejam estendidas e lhe garante flexibilidade no número de operações, tipos de dados e formulários contemplados. Seus arquivos CSS e de script podem ser modificados para que a apresentação de sua interface gráfica seja aperfeiçoada.

Durante o desenvolvimento do framework Guiwe e da interface gráfica do framework GRENJ, as listas de casos de teste definiram os detalhes da implementação das classes e deram origem a documentação *javadoc* desses framewoks. Isso reduziu o número de artefatos de documentação do processo de

desenvolvimento. Além de descrever os métodos, a documentação *javadoc* define o que os testes devem verificar e quais as interfaces do projeto. Essa abordagem tornou o trabalho de construção das classes mais eficiente, uma vez que não era necessário consultar e manter a consistência de muitos artefatos. Além disso, a verificação do funcionamento dos métodos por meio dos testes garantiu melhor qualidade aos frameworks Guiwe e GRENJ.

Alguns trabalhos futuros podem ser realizados com o intuito de adicionar componentes com mais recursos na camada de interface gráfica dos framework Guiwe. Por exemplo, instruções podem ser adicionadas nos documentos JavaScript para fornecer aos campos de texto máscaras para números de telefone, números de documentos, dados monetários e outros valores. Podem, inclusive, ser implementados componentes com calendários para os campos com datas.

Referências

1. Gamma, E., Helm, R., Johnson, R. e Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1995)
2. Shiva, S. G. e Shala, L. A.: Software Reuse: Research and Pratice. In: 4th International Conference on Information Technology, pp. 603--609 (2007)
3. Johnson, R. E.: Frameworks = (Components + Patterns). Communications of the ACM, vol. 40, no. 10, pp. 39--42 (1997)
4. Abi-Antoun, M.: Making Frameworks Work: A Project Retrospective. In: Conference on Object Oriented Programming Systems Languages and Applications, pp. 1004--1018 (2007)
5. Liem, I. e Nugroho, Y.: An application generator framelet. In: 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD '08), pp. 794--799 (2008)
6. Larman, C.: Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design and the Unified Process. 2nd Edition, Prenctice-Hall (2002)
7. Brugali, D. e Sycara, K.: Frameworks and Pattern Languages: an Intriguing Relationship. ACM Computing Surveys, vol. 32, no. 2 (2000)
8. Apache Struts, http://struts.apache.org
9. Spring Source, http://www.springsource.com
10. Harrison, N. B., Avgeriou, P., e Zdun, U.: Using Patterns to Capture Architectural Decisions. IEEE Software, vol. 24, No. 4, pp. 38--45 (2007)
11. Janzen, D. e Saiedian, H.: Does Test-Driven Development Really Improve Software Quality. IEEE Software, vol. 25, no. 2, pp. 77--84 (2008)
12. JUnit.org Resources for Test Driven Development, http://www.junit.org
13. JQuery JavaScript Library, http://jquery.com
14. Durelli, V. H. S., Viana, M. C. e Penteado, R. A. D.: Uma Proposta de Reúso de Interface Gráfica com o Usuário Baseada no Padrão Arquitetural MVC. In: IV Simpósio Brasileiro de Sistemas de Informação (SBSI'08), pp. 48--59, Rio de Janeiro (2008)
15. Braga, R. T. V. e Masiero, P. C.: A Process for Framework Construction Based on a Pattern Language. In: 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment (COMPSAC'02), p. 615--622 (2002)
16. Chaves, A.P.; Leal, G.C.L.; Huzita, E.H.M.: An Experimental Study of the FIB Framework Driven by the PDCA Cycle. In: International Conference of the Chilean Computer Science Society (SCCC '08), pp. 23--31 (2008)