PARTENON: GESTÃO INTEGRADA DE EVENTOS

Roberson Junior Fernandes Alves*
Luiz Felipe Bartz**
Robson Gian Perassoli***

Resumo

Este artigo tem por objetivo apresentar o Partenon (Sistema para Gestão Integrada de Eventos), desenvolvido sobre a modalidade de *software* livre, oferecendo aos usuários uma ferramenta completa para o gerenciamento e controle de eventos. Não se prendendo a um tipo específico de evento, o Partenon contém funcionalidades genéricas para tipologias distintas. Entre as principais funcionalidades se destaca a integração com redes sociais auxiliando na divulgação e marketing do evento. Para isso, foram utilizadas ferramentas de desenvolvimento com suporte ao ambiente *web* e linguagem de programação Java, por meio do *framework* integrador *Demoiselle*, desenvolvido pelo Governo Federal na versão 2.0 tendo como finalidade facilitar o desenvolvimento de aplicações, trazendo como padrão uma arquitetura baseada no modelo *Model-View-Controller* (MVC). O *Demoiselle* possui suporte como a *Java Persistence API* (JPA), utilizada para facilitar a persistência das informações dos objetos no Sistema Gerenciador de Bancos de Dados (SGBD) *PostgreSQL*.
Palavras-chave: *Demoiselle*. *Framework*. Partenon. Java *Web*. Eventos.

1 INTRODUÇÃO

O processo de informatizar se tornou indispensável e fundamental para qualquer ramo de atividade, assim como a necessidade de um *software* que inclua várias aplicações e processos que permitam a uma organização realizar e controlar seus negócios. Porém, um simples aplicativo não significa obter sucesso no concorrente mundo da tecnologia de informação. Nesse ponto, a escolha de tecnologias é fundamental para garantir um desenvolvimento eficaz, trazendo como resultado um produto mais completo.

Assim, a linguagem Java está cada vez mais popular em aplicações cotidianas em razão das características como ser livre e com várias plataformas para desenvolvimento. Desse modo, o *Demoiselle* 2.0, assim como a sua primeira versão, é voltado para o desenvolvimento *web* com grande número de *frameworks* acoplados, garantindo várias características indispensáveis para uma atual aplicação.

A aplicação desenvolvida utilizou a linguagem de programação orientada a objetos Java mediante o *framework Demoiselle* 2.0, executado no ambiente *web*. Permitindo agilizar e automa-

* Especialista em Ciência da Computação pela Universidade Federal de Santa Catarina. Professor dos Cursos de Sistemas de Informação e Ciência da Computação da Universidade do Oeste de Santa Catarina de São Miguel do Oeste; Coordenador dos Laboratórios de Tecnologia da Informação da Universidade do Oeste de Santa Catarina Campus de São Miguel do Oeste; roberson.alves@unoesc.edu.br
** Graduando do Curso de Bacharelado em Sistemas de Informação; lipemh@mhnet.com.br
*** Graduando do Curso de Bacharelado em Sistemas de Informação; robsonperassoli@gmail.com

tizar os processos envolvidos nos eventos e suas tipologias. Como o *framework* e os demais *frameworks* acoplados a ele seguem o conceito de *software* livre, a aplicação também adota este conceito.

2 *FRAMEWORKS* E O *JAVA PERSISTENCE API* (JPA)

Quando a reusabilidade se torna fator de influência sobre o tempo, deve-se avaliar a necessidade de um *framework*, pois segundo Braude (2005, p. 566), o objetivo mais importante no desenvolvimento de *software* é o reuso. Falando de *framework* e reuso, surge a ideia de um conjunto de classes e métodos em que o autor explica que "Um *framework* é uma coleção de artefatos de *software* que é utilizado por várias aplicações diferentes. Esses artefatos são em geral classes, juntamente com o software exigindo para utilizá-las." (BRAUDE, 2005).

Um *framework* tenta resolver problemas de baixo nível como segurança, acesso a dados, comunicação. Então, "No desenvolvimento do software, um framework ou arcabouço é uma estrutura de suporte definida em que um outro projeto de software pode ser organizado e desenvolvido." (BRASIL, 2009) .

Persistência é a ação de armazenar dados em bancos de dados relacionais utilizando a linguagem *Structured Query Language* (SQL). O Mapeamento Objeto-Relacional (ORM) é a persistência de objetos acontecendo de forma transparente e automatizada para tabelas do banco de dados relacional, mapeamento este feito usando metadados, os quais ficam localizados na aplicação e descrevem as tabelas. Para suprir a necessidade de utilizar linguagens de programação orientadas a objetos, em conjunto com bases de dados relacionais, surgiram ferramentas que facilitam este trabalho, tornando possível armazenar objetos em bancos de dados relacionais de forma simples, utilizando-se de vários padrões (*design-patterns*) e também alguns *frameworks*, que podem desempenhar esse papel de forma automatizada (BAUER; KING, 2005).

A tecnologia JPA não é um *framework* de persistência propriamente dito, mas uma especificação da comunidade Java. Esta especificação pode ser implementada por qualquer empresa, desde que siga os padrões definidos pela comunidade, como é caso das grandes fabricantes de ferramentas de persistência: *JBoss Hibernate*, *Oracle Toplink* e *EclipseLink*. Esta forma de trabalhar permite que a aplicação trabalhe com qualquer *framework* de persistência que siga os padrões JPA de forma desacoplada, permitindo a troca quando for necessária sem maiores danos à aplicação (KEITH; SCHNICARIOL, 2009).

3 **DEMOISELLE 2.0**

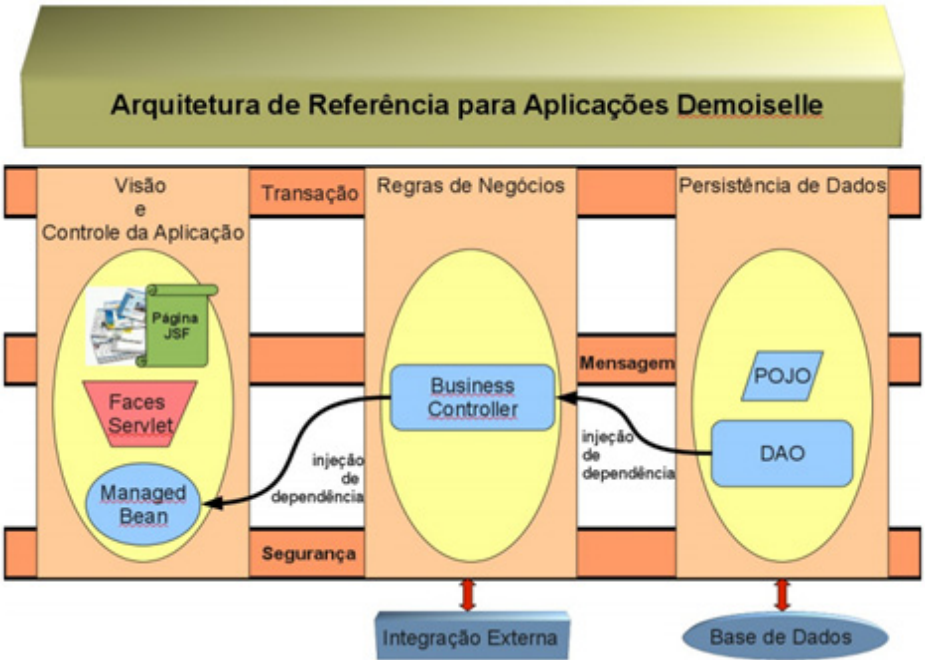
O *Demoiselle framework* é construído a partir do conceito de *framework* integrador, incorporando diversas ferramentas utilizadas no mercado Java. Objetiva facilitar o desenvolvimento de aplicações, privando o desenvolvedor de perder tempo escolhendo os *frameworks* especialistas que serão usados no seu projeto, resultando em grande aumento da produtividade, além de facilitar a manutenção dos sistemas. Possui mecanismos facilitadores voltados à resolução dos problemas mais comuns em uma aplicação. Entre eles estão a arquitetura, a segurança e a configuração (SACRAMENTO et al., 2011).

A estrutura do *Demoiselle* é dividida em *Core*, que contém as funcionalidades comuns a todas as aplicações; é a base, o núcleo propriamente dito. Extensões, por sua vez, são funcionalidades extras extremamente ligadas ao núcleo, porém, específicas a um domínio, como é o caso de JPA e JSF, pois algumas aplicações não fazem uso de persistência, não fazendo sentido estarem no núcleo. Por fim, os componentes, que são artefatos independentes do núcleo, não precisam estender as funcionalidades do *core*, têm ciclo de vida próprio e não precisam necessariamente fazerem uso do *Demoiselle* (SACRAMENTO et al., 2011).

Além da estrutura bem definida, o *Demoiselle* adota padrões aceitos mundialmente, o que possibilita ao desenvolvedor estar desacoplado de produtos específicos, pois a implementação do *framework* se baseia em especificações definidas pela comunidade Java (LISBOA, 2010). Apesar do *framework* não forçar o desenvolvedor a utilizar nenhum tipo de arquitetura, o *Demoiselle* fornece uma arquitetura que pode ser usada como referência (Esquema 1) na criação de uma aplicação *web*. Esta arquitetura é constituída de três camadas, seguindo o padrão MVC, tendo na parte de modelo os Pojos (*Plain Old Java Objects*) e o *Data Access Objects* (DAO), um padrão usado em Java para auxiliar na persistência de dados.

Na parte de visão podem ser citadas as páginas *Java Server Faces* (JSF), e, por fim, pode ser citado o Controlador, que são os *Managed Beans* (Beans Gerenciados) os quais interagem com a Visão e são integrados aos *Business Controllers*, que têm como objetivo efetuar a lógica de negócios da aplicação (SACRAMENTO et al., 2011).

Esquema 1 – Estrutura geral do *Demoiselle*



Fonte: *Demoiselle framework* (2010).

A arquitetura MVC é um conceito que tem como objetivo a separação da aplicação em três partes distintas o Modelo, que nada mais é do que o meio de acesso e persistência dos dados de negócio; a Visão, que pode ser descrita como a parte relacionada à exibição dos dados, e também é responsável por receber as entradas do usuário; e, por fim o Controlador, que faz o

trabalho de aplicar as regras de negócio da aplicação e tratar os dados recebidos da Visão, dados estes que poderão ser persistidos no Modelo, e após processados e persistidos podem retornar informação para o usuário (GONÇALVES, 2007). A união de tecnologias possibilita ao *Demoiselle* um ambiente completo para o desenvolvimento de aplicações voltadas para a *web*, ganhando tempo, produtividade e redução do retrabalho.

4 **PARTENON**

Em conversas informais com possíveis usuários do sistema, foram elencados os seguintes requisitos macro para o desenvolvimento do Partenon:

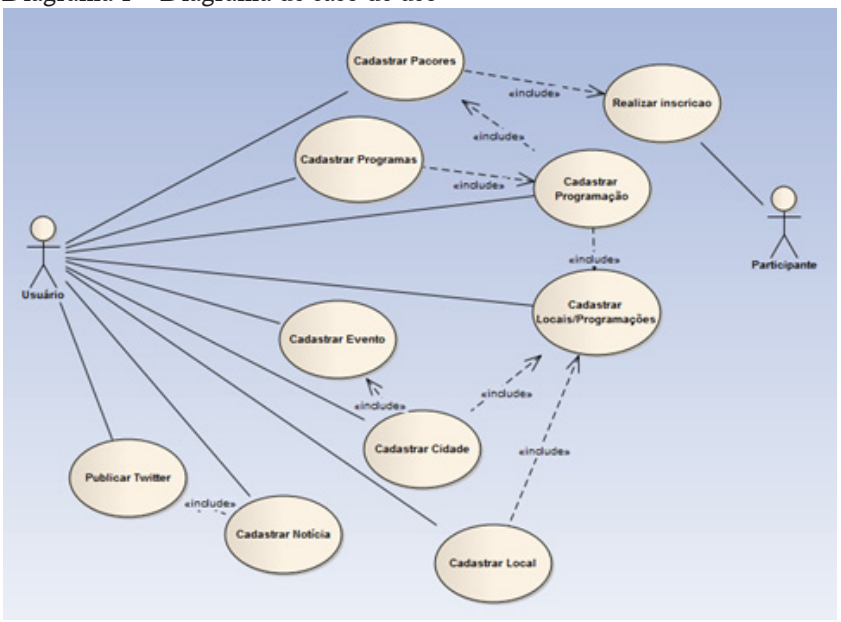
a) criação de portal em que será possível visualizar notícias e páginas as quais serão cadastradas no sistema e atualizadas dinamicamente;
b) no portal, possibilitar aos participantes a visualização de imóveis para a hospedagem, que devem ser cadastrados no sistema;
c) criação de módulo para controle das movimentações financeiras;
d) módulo para o controle da locação de *stands*.

Para simplificar e permitir o entendimento da equipe sobre a aplicação, foram definidos que seriam usados diagramas de caso (Diagrama 1) de uso para modelar as principais funcionalidades do sistema.

Para o desenvolvimento da aplicação, utilizou-se da *Integrated Development Environmen*t NetBeans (IDE) 7.0, integrado com o *framework Demoiselle* 2.0 no qual estão compostos também os *frameworks* jUnit e JSF, sendo jUnit ferramenta para a automatização de testes em projetos Java e JSF para o desenvolvimento da *interface* com o usuário. A base de dados foi desenvolvida com PostgreSQL 9.0, sendo este um SGBD caracterizado por ser robusto e gratuito.

Assim, para a gestão no desenvolvimento, utilizaram diversas tecnologias, sendo elas dotProject para facilitar a gerência e acompanhamento do até então desenvolvido; e o que está pendente desenvolver, assim como parte da documentação do sistema SVN (*Subversion*) permitindo o controle e o gerenciamento de versão com a equipe de desenvolvimento e Apache Maven auxiliando no gerenciamento de automação de projetos em Java.

Diagrama 1 – Diagrama de caso de uso

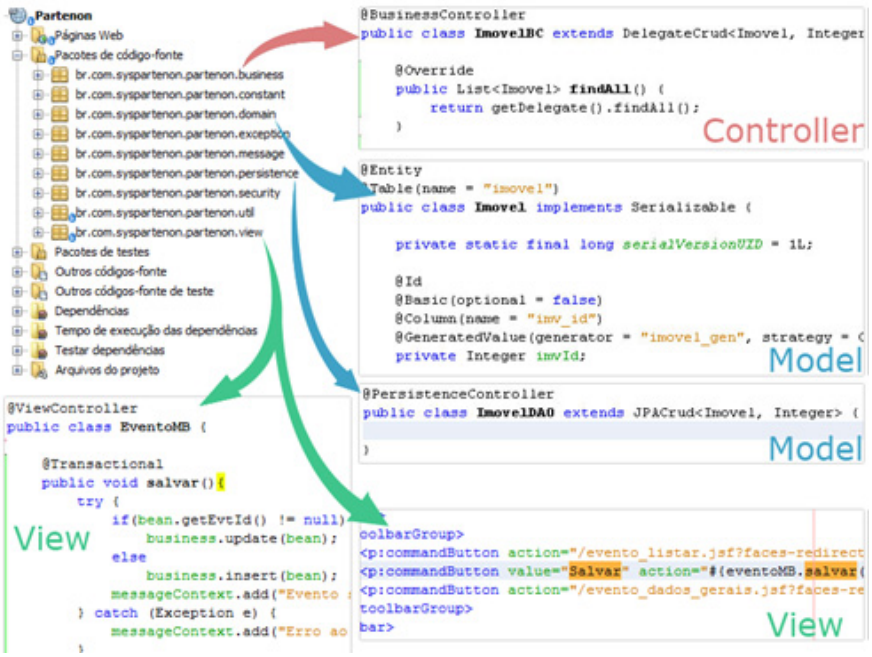


Fonte: os autores.

O desenvolvimento é uma tarefa que está muito suscetível a erros, e, para que essas falhas possam ser corrigidas, devem ser realizados testes. Estes podem ser manuais, mediante ferramentas de depuração por meio de ferramentas para realizar testes automatizados. Tais ferramentas de testes automatizados possibilitam que um teste, uma vez escrito, possa sempre ser executado, diminuindo o tempo gasto com testes. Para a implementação das rotinas de teste foi utilizada a ferramenta jUnit. Já para o desenvolvimento das telas foi utilizada uma biblioteca de componentes para JSF chamada *PrimeFaces*, que, além de deixar o sistema mais atraente e usável ao usuário final, minimiza o esforço na hora do desenvolvimento.

Como o Partenon é um sistema de médio porte, caso não houvesse organização na estrutura de arquivos e códigos-fonte, o desenvolvimento poderia se tornar caótico, dificultando o andamento do projeto e das manutenções futuras. O Esquema 2 apresenta a estrutura de organização do projeto na IDE Netbeans.

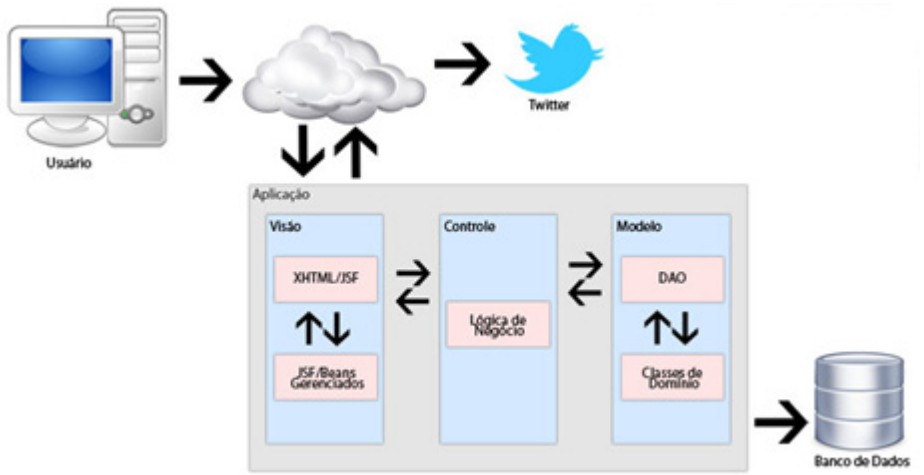
Esquema 2 – Estrutura da aplicação Netbeans (MVC)



Fonte: os autores.

Para facilitar o entendimento do funcionamento da aplicação, foi elaborado o Esquema 3, que tem como objetivo exemplificar o fluxo do sistema desde a interação do usuário até a integração entre os componentes internos do Partenon.

Esquema 3 – Arquitetura da aplicação



Fonte: os autores.

O Sistema Partenon ficou dividido em módulos (Esquema 4), que contêm características semanticamente similares. Esse estilo de organização facilita o entendimento do usuário sobre o Sistema.

Esquema 4 – Módulos da aplicação

Fonte: os autores.

Na Imagem 1 é apresentada a visualização do usuário após efetuar o *login* onde no *Menu* central são disponibilizados ícones grandes com atalho para algumas funcionalidades do sistema.

Imagem 1 – Tela principal do Sistema Partenon

Fonte: os autores.

A barra do *Menu* superior sempre fica visível ao usuário onde as demais páginas são carregadas dinamicamente na parte central, tornando o sistema mais fácil no caso de o usuário necessitar acessar outras telas para as demais funcionalidades.

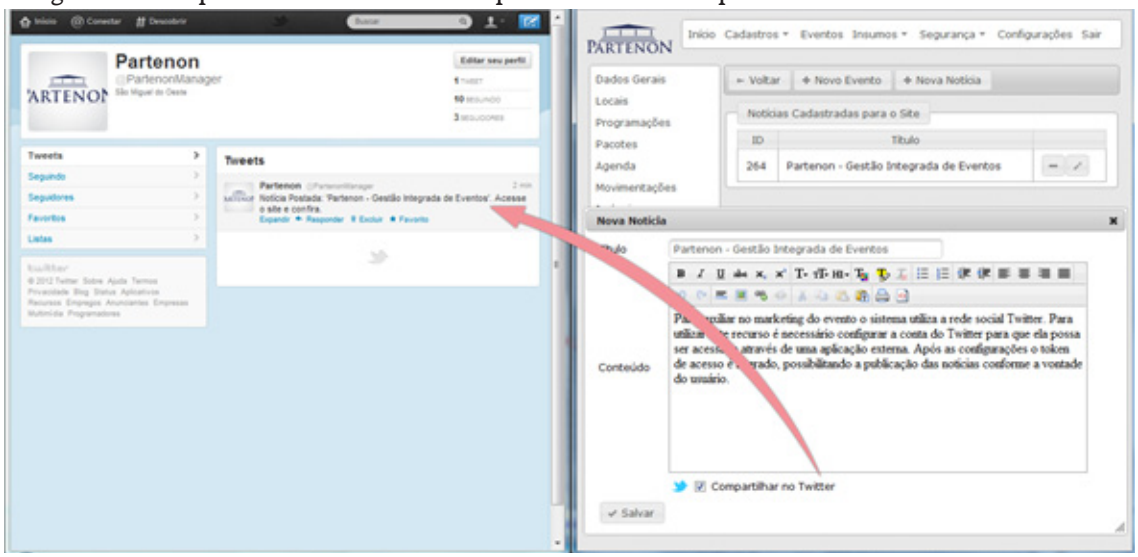
Dentro dos módulos específicos do Evento, para a divulgação deste, foi desenvolvido um *site* (Imagem 2) que pode ser alimentado direto da aplicação, cadastrando páginas, notícias, galeria de fotos, *banners* e enquetes. O gerenciamento e a manutenção do *site* podem ser efetuados pelo próprio usuário da aplicação, pois após salvar as informações, estas já ficam disponíveis nele.

Imagem 2 – *Site* do Evento gerado e gerenciado pelo Sistema Partenon

Fonte: os autores.

Outra funcionalidade no gerenciamento do *site* são as notícias, que além de estarem disponíveis, podem também ser propagadas pela rede social *Twitter* auxiliando no marketing do próprio evento. Para utilizar esse recurso é necessário configurar a conta do *Twitter* para que ela possa ser acessada por meio de uma aplicação externa. Após as configurações, o *token* de acesso é liberado, possibilitando a publicação das notícias conforme a vontade do usuário (Imagem 3).

Imagem 3 – Compartilhamento de notícias por meio do *Twitter* pelo sistema Partenon



Fonte: os autores.

Com os recursos do Partenon, o usuário tem a possibilidade de controlar o seu evento de forma eficaz, em ambiente *web* e, assim, realizar sua divulgação por meio de mecanismos avançados aceitos e utilizados globalmente.

5 CONCLUSÃO

Constatou-se que a utilização de um *framework* pode auxiliar no desenvolvimento de tarefas que se tornariam repetitivas, tendo maior agilidade no desenvolvimento. Não foram encontradas dificuldades na utilização do *framework* em razão do grande número de pessoas contribuindo para a comunidade do *Demoiselle*. A documentação é muito simples e completa, favorecendo uma pequena curva de aprendizado, possibilitando ter conhecimentos avançados em pouco tempo. Essa facilidade proporcionou a um dos integrantes acesso à publicação no *blog* oficial do *Demoiselle*, por causa da divulgação de um tutorial mediante a lista de *e-mails* oficiais do *framework*.

O desenvolvimento de uma aplicação informatizada no ambiente *web* não garante apenas a realização e o controle de negócios, mas também a disponibilidade das informações. O Partenon oferece uma solução que permite o controle e a gestão dos processos envolvidos em eventos, não sendo restrito a uma tipologia específica.

Atenção a vários tipos de eventos, permissão de um total controle de pessoas e processos envolvidos no evento, geração de um *site* alimentado pelo próprio usuário, integração com redes sociais e um ambiente *web* agregaram valores à aplicação, tornando o Partenon um potencial *software* para o mercado.

A escolha de tecnologias foi fator determinante para o desenvolvimento eficaz, pois estas garantiram a separação, a padronização e a centralização de funções e operações ganhando tempo com o desenvolvimento e facilitando a manutenção do sistema. Um ponto fundamental para o sucesso do trabalho em equipe foi o gerenciamento de projetos, em que todo o processo de desenvolvimento foi planejado e dividido em tarefas, e cada componente do grupo realizou-as podendo adicionar anotações, tendo um maior controle do que foi feito e a quantidade de tempo para realizá-las.

Contudo, a elaboração do trabalho foi um grande desafio aos desenvolvedores, sendo associado ao ganho de conhecimento, possibilitando crescimento pessoal e servindo como importante experiência profissional.

Partenon: integrated event management

Abstract

This article aims to present the Partenon (Integrated Management System for Events) developed on open source modality, giving for users a complete tool for managing and controlling events. The Partenon contains generic functionality for different typologies of events. Among the main features highlight the integration with social networks aiding in the dissemination and marketing of the event. For this we used development tools to support web environment and the Java programming language, using the framework integrator Demoiselle developed by the Governo Federal in version 2.0. Intended to facilitate the development of applications, bringing as standard architecture model based on Model-View-Controller (MVC). The Demoiselle has support for Java Persistence API (JPA), used to facilitate the persistence of objects in DataBase Management System (DBMS)PostgreSQL.
Keywords: Demoiselle. Framework. Partenon. Java Web. Events.

REFERÊNCIAS

BAUER, Christian; KING, Gavin. **Hibernate em Ação**. 2. ed. Rio de Janeiro: Ciência Moderna, 2005. 530 p.

BOTELHO. Vanderson. **Desenvolvimento Web com Framework Demoiselle versão 1.0**. Curitiba, 2009. Disponível em:<http://www.inf.ufpr.br/andrey/desWeb/aula9/Demoiselle-Tutorial-Modulo01-Arquitetura-Apresentacao.pdf>. Acesso em: 31 out. 2011.

BRAUDE, Eric. **Projeto de Software**: Da programação à arquitetura: uma abordagem baseada em Java. Porto Alegre: Bookman, 2005. 619 p.

DEMOISELLE FRAMEWORK. **Arquitetura de Referência para Aplicações Demoiselle**. 2010. Disponível em: <http://demoiselle.svn.sourceforge.net/viewvc/demoiselle/framework/trunk/docs/analysisdesign/dodo/demoiselle-aplicacoes_documento_arquitetura_software%28DAS%29.pdf>. Acesso em: 30 maio 2011.

GONÇALVES, Edson. **Desenvolvendo Aplicações Web com JSP, Servlets, Javaserver Faces, Hibernate, EJB 3 Persistence e Ajax**. Rio de Janeiro: Ciência Moderna, 2007. 736 p.

KEITH, Mike; SCHNICARIOL, Merrick. **Pro JPA 2**: Mastering the Java™ Persistence API. New York: Apress, 2009. 481 p.

LISBOA, Flavio Gomes da Silva. **Introdução ao Demoiselle Framework**: uma abordagem comparativa de utilização do padrão MVC para o desenvolvimento de aplicações Web em Java orientada ao reuso. 2010. Disponível em: <http://www.frameworkdemoiselle.gov.br/>. Acesso em: 28 maio 2011.

MACORATTI, José Carlos. **Padrões de Projeto**: o modelo MVC – Model View Controller. Disponível em: <http://www.macoratti.net/vbn_mvc.htm>. Acesso em: 22 maio 2011.

SACRAMENTO, Cleverson et al. **Framework Demoiselle 2.0**: guia de Referência, 2011. Disponível em: <http://demoiselle.sourceforge.net/docs/reference/2.0-v6/pdf/demoiselle-reference.pdf>. Acesso em: 24 maio 2011.