UNIVERSIDADE DO OESTE DE SANTA CATARINA – UNOESC
CAMPUS DE SÃO MIGUEL DO OESTE

LUIZ FELIPE BARTZ
ROBSON GIAN PERASSOLI

PARTENON:
GESTÃO INTEGRADA DE EVENTOS

São Miguel do Oeste (SC)
2012

LUIZ FELIPE BARTZ
ROBSON GIAN PERASSOLI

PARTENON:
GESTÃO INTEGRADA DE EVENTOS

Monografia apresentada à Universidade do Oeste de Santa Catarina – Campus de São Miguel do Oeste como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Esp. Roberson Junior Fernandes Alves

São Miguel do Oeste (SC)
2012

Dedicamos este trabalho aos nossos Pais, pois sempre nos deram apoio e força necessária e não mediram esforços pra concluirmos mais esta etapa de nossas vidas.

AGRADECIMENTOS

Agradecemos primeiramente a nossas famílias por acreditarem em nossa capacidade assim como todo incentivo, compreensão e dedicação que nos concederam.

Aos amigos de longa caminhada e aos que adicionamos em nossas vidas no decorrer do curso, aqueles que estiveram conosco em muitas situações dando-nos força e apoio quando necessitamos.

Ao professor e orientador Roberson J. F. Alves por acreditar, colaborar e incentivar a ideia assim como na execução e conclusão deste trabalho.

Agradecimentos à coordenadora do curso de Sistemas de Informação professora Otília Donato Barbosa por trazer até nós a idéia do presente Trabalho e diretamente os dirigentes do Moto Grupo Cães do Asfalto por confiar no desenvolvimento do mesmo.

A Comunidade *Demoiselle* por auxílio prestado em sua lista de discussões e reconhecimento de nossos conhecimentos com o *framework* através do acesso a publicações em seu blog oficial.

Agradecer aos organizadores do I BootCamp OeSC-Livre em Xanxerê – SC, pela oportunidade de explanar nosso conhecimentos com o *framework Demoiselle* e divulgar o Partenon para sua lista.

Também, agradecimentos especiais aos demais professores de Sistemas de Informação que repassarão seus ensinamentos auxiliando indiretamente na execução deste projeto, mas também em nossa formação profissional.

RESUMO

Este trabalho 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. Tem como objetivo primordial suprir as necessidades gerenciais do evento Motocão, organizado pelo Moto Grupo Cães do Asfalto de São Miguel do Oeste – SC. Não se prendendo a um tipo especifico de evento, o Partenon contém funcionalidades genéricas para tipologias distintas. Dentre as principais funcionalidades destaca-se 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, através 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 MVC (*Model-View-Controller***)**. O *Demoiselle* possui suporte a JPA (*Java Persistence API),* uma API utilizada para facilitar a persistência das informações dos objetos no SGBD (Sistema Gerenciador de Bancos de Dados) *PostgreSQL*.

Palavras-chave: Partenon, *Demoiselle*, *Framework.*

ABSTRACT

This work aims at presenting the Partenon (Integrated Management System for Events) developed based on open source software, and provides to users a complete tool for managing and controlling events. Like primary needs have the management of Motocão organized by Moto Grupo Cães do Asfalto from São Miguel do Oeste – SC. The Application does not focus on a specific type of event, containing basic features of other types of events. Among the main features can be highlighted the integration with social networks assisting in the dissemination and event marketing. For this project some tools were used, providing a complete environment with support for Web and Java programming language through the Demoiselle integrative framework developed by the Federal Government in version 2.0. With premises of facilitate application development, bringing as a standard architecture based on MVC (Model-View-Controller). The Demoiselle has support for JPA (Java Persistence API), an API used to provide facilities for persistence of the information of the objects in the DBMS (Database Management System) PostgreSQL.

Keywords: Partenon, Demoiselle, Framework.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

A&POO	*Object-Oriented Analysis and Desing*
HTML	*HyperText Markup Language*
JEE	*Java Enterprise Edition*
JME	*Java Micro Edition*
JPA	*Java Persistence Api*
JSE	*Java Standard Edition*
JSF	*Java Server Faces*
JSP	*Java Server Pages*
MVC	Modelo, Visão e Controlador
OO	Orientado a Objetos
ORM	*Object Relational Mapping*
SCM	*Software Configuration Managment*
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
SVN	*Subversion*
XML	*Extensible Markup Language*
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

O processo de informatizar se tornou indispensável e fundamental para qualquer ramo de atividade, assim como o aumento da 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. Neste ponto a escolha de tecnologias é fundamental para garantir um desenvolvimento eficaz.

Sendo assim, a linguagem Java está cada vez mais popular em aplicações cotidianas por ser livre e contar com grande variedade de ferramentas. Em consonância, o *framework Demoiselle* 2.0 assim como sua primeira versão integra esta gama de ferramentas conhecidas e largamente utilizadas pela comunidade *Java*, garantindo várias características indispensáveis para uma aplicação nos dias de hoje.

O presente trabalho fez uso da linguagem de programação orientada a objetos *Java* através *do framework Demoiselle* 2.0 sendo executado no ambiente web. Esta aplicação permite agilizar e automatizar os processos envolvidos nos eventos e suas tipologias. Como o *framework* e suas demais ferramentas seguem o conceito de software livre, a aplicação também adotou o mesmo conceito.

No decorrer do trabalho serão abordados conceitos relacionados a eventos bem como suas tipologias. Também, serão descritas as tecnologias que auxiliaram o desenvolvimento e a metodologia de trabalho aplicada. Após isso, serão apresentados os resultados obtidos e em seguida as diretrizes para continuidade do mesmo.

1.1 OBJETIVOS

A seguir são descritos os objetivos direcionam o presente trabalho.

1.1.1 Objetivo Geral

Desenvolver uma aplicação explorando a modalidade de software livre através do *framework Demoiselle*, oferecendo aos usuários uma ferramenta completa para o gerenciamento e controle de eventos.

1.1.2 Objetivos Específicos

- Implementar e modelar a aplicação utilizando o paradigma orientado a objetos através da linguagem Java;
- Utilizar *frameworks*, padrões de projeto e boas práticas para o desenvolvimento da aplicação;
- Desenvolver a aplicação visando a web como seu ambiente operacional;
- Adotar ciclo de vida da engenharia de software adequado para o desenvolvimento de aplicações web utilizando as metodologias ágeis Scrum e XP;
- Testar a aplicação implementando testes unitários através da ferramenta *JUnit* que compõe o *framework Demoiselle*;
- Integrar a aplicação as redes sociais e permitir a disponibilização de blogs relacionados aos eventos gerenciados;
- Validar a aplicação junto aos usuários finais.

1.2 JUSTIFICATIVA/PROBLEMATIZAÇÃO

Mudanças ocorrem a todo o momento na sociedade, e os indivíduos devem adaptar-se a elas. Processos que já foram realizados manualmente hoje são realizados através de máquinas com poucos cliques. Com a facilidade de acesso houve um aumento proporcional do volume da informação, juntamente com a necessidade de armazenar e transformar estes dados em informação útil.

Para suprir estas necessidades é possível fazer uso de sistemas de informação, que possibilitem armazenar e tratar dados que sejam pertinentes a uma organização. Visto que os sistemas de informação são ponto chave para tratar dados. Pode-se dizer que um sistema de informação na web além de tratar a informação, possibilita que ela esteja disponível para várias pessoas em qualquer parte do mundo, algo pertinente quando se fala de comunicação e informação.

Tendo em vista o grande potencial de aplicações web para divulgação e compartilhamento de informação, características estas que podem ser usadas em benefício da aplicação proposta que tem como finalidade simplificar a tarefa de gerenciar eventos usando os recursos que a internet proporciona para dissipação da informação. Qualidades estas que serão usadas para realizar tarefas que são feitas de forma manual, em planilhas do Excel ou até mesmo em papel. Sterene (2000) explica que os negócios encontraram várias formas para a comunicação eletrônica através da internet, desde a redução em custos com papéis até o aperfeiçoamento das comunicações internas, causando assim satisfação dos clientes e facilitando o *feedback* aos gerentes. O desenvolvimento deste sistema foi tulizado o *Demoiselle framework.*

O uso do *framework* citado partindo do pressuposto que o código desenvolvido está relacionado com o desenvolvimento de regras de negócio e não de tarefas genéricas e custosas que são praticamente iguais na maioria os sistemas, pois de acordo com Lisboa (2010) um *framework* tem como principal propósito auxiliar no processo de desenvolvimento de aplicações, permitindo que sejam desenvolvidas mais rapidamente e facilmente. O *Demoiselle* integra várias tecnologias e meios que facilitam o desenvolvimento de aplicações que fazem uso de internet.

O sistema proposto tem a capacidade de auxiliar o controle de vários tipos de eventos, fechados ou abertos ao público. Dentre eles estão: congressos, seminários, workshops,

encontros, feiras, cursos e eventos em geral. Implementou-se funcionalidades que auxiliam o organizador do evento a controlar de forma geral os gastos relacionados ao evento. Para maior segurança o administrador pode controlar os usuários através de perfis pré-definidos.

A aplicação possibilita a integração com um blog criado para o evento, que terá como finalidade a divulgação de notícias, fotos, e outras informações relacionadas que pode ser cadastradas por um usuário que tenha acesso. Outra funcionalidade é a integração com o microblog Twitter, possibilitando a propagação das informações do evento.

2 REVISÃO DA LITERATURA

A revisão literária a seguir citará alguns tipos de eventos juntamente com suas características, e também conceitos básicos relacionados à realização de eventos, e os requisitos que eles precisam preencher para serem bem sucedidos. Em seguida serão abordados padrões de projeto e de arquitetura, logo após uma breve abordagem sobre *frameworks* que auxiliam o desenvolvimento de projetos.

Serão abordados tópicos relacionados a metodologias de desenvolvimento de aplicações, enfatizando o desenvolvimento de aplicações Web.

Os principais conceitos da linguagem Java serão abordados em seguida, bem como, fundamentos dos principais *frameworks* para desenvolver aplicações Java na Web e tecnologias utilizadas atualmente no desenvolvimento de aplicações.

2.1 EVENTOS

Inicialmente será abordado o assunto de eventos mediante a apresentação e definição dos conceitos e também os diversos tipos de Eventos.

2.1.1 Definição de Eventos

Segundo Melo Neto (2003) um evento é algum fato que possa gerar sensação e, por isso, ser motivo de notícia. Descrevendo mais a fundo, um fato é algo que acontece, tem datas e horários de início e fim. A realização de um evento está associada a um momento de tempo, e a um local determinado, por isso deve ser planejado corretamente e estar enquadrado às características do realizador, do público que irá prestigiar o evento e do ambiente em que será realizado. Se o evento for um acontecimento bem-sucedido ele virará notícia na mídia.

Cesca (1997) complementa, “Evento é um fato que desperta a atenção, podendo ser notícia e com isso divulgar o organizador”.

Pressman (2006, p.02) afirma que um o software assume um duplo papel.

> Ele é o produto e, ao mesmo tempo, o veículo para entrega de produto. Como produto ele disponibiliza o potencial de computação presente no hardware local, Quer resida em um telefone celular, que opere em um computador de grande porte, o software é um transformador de informações [...] Como veículo usado para entrega do produto, o software age como base para controle do computador (sistemas operacionais), para comunicação da informação (redes) e para criação e o controle de outros programas (ferramentas e ambientes de software).

Segundo Giácomo (1993), "Evento é um componente do mix da comunicação, que tem por objetivo minimizar esforços, fazendo uso da capacidade sinérgica da qual dispõe o poder expressivo no intuito de engajar pessoas numa idéia ou ação".

A Internet é atualmente um dos meios mais utilizados para comunicação e propagação da informação, sendo um dos desenvolvimentos mais importantes na história da computação justamente por possibilitar a conexão com bilhões de pessoas ao redor do mundo. Ressaltando que essa tecnologia está integrada ao cotidiano das pessoas no século XXI (PRESSMAN, 2006).

Atualmente uma das formas adotadas para o relacionamento entre pessoas através da internet são as redes sociais, para Alcará, Chiara e Tomaél (2005, p. 1), através das relações desenvolvidas durante a vida no âmbito familiar, escolar, comunidade e no trabalho as pessoas estão inseridas na sociedade, onde a própria natureza humana os liga e organiza a sociedade em rede. Nas redes sociais cada indivíduo possui uma função e identidade cultural, sua relação com os outros indivíduos forma a união que representa a rede.

A busca de relacionamento através da internet fez com que as redes sociais acumulassem grande número de pessoas, segundo Instituto Evaldo Lodi (2009, p. 13):

> Uma pesquisa da ComScore, companhia internacional de medição digital, mostra que, no ano passado,85% dos 41,5 milhões de usuários de internet no Brasil com mais de 15 anos entraram por alguma rede social. Projeções do Ibope Nielsen Online indicam que o número este ano deve chegar a 62,3 milhões de pessoas com mais de 16 anos. O Orkut, rede social do Google, era o mais acessado com 21 milhões de internautas em setembro de 2008, segundo a ComScore. Outro estudo realizado pela E.Life, empresa brasileira líder na monitoração e análise da mídia gerada pelo consumidor, o boca a boca, confirrma o favoritismo do Orkut em relação aos outros sites de relacionamento com 45% da preferência dos entrevistados. Mas mostra também o avanço de outros no período de janeiro a abril de 2009, em relação ao ano passado. O acesso ao Twitter, página pessoal ou de empresa, que permite enviar e ler atualizações em tempo real em textos de até 140 caracteres, cresceu de 3,8% em 2008 para 23% este ano. O Blogspot é o terceiro colocado com 12% de participação total desse mercado.

Descritos os conceitos básicos relacionados à realização de eventos, e a capacidade da web como ambiente principal para a divulgação e disponibilização da informação, a seguir serão citados os tipos de eventos que a aplicação suporta e quais suas principais características.

2.1.2 Tipos de Eventos

Dentre os vários tipos de eventos, de forma genérica podem ser descritos alguns dos tipos mais abrangentes, e que serão usados no decorrer deste trabalho.

- Congresso: Reunião formal e periódica de pessoas que pertencem a grupos profissionais com interesse em comum. Promovido geralmente por entidades associativas com objetivo de estudar, debater e chegar a conclusões de um tema geral sendo exposto em subtemas. (MEIRELLES, 2003).
- Seminários: Exposição verbal feita por pessoas colocadas no mesmo plano, onde os participantes possuem prévio conhecimento do assunto, com objetivo de fornecer e somar informações de temas já pesquisados. (MATIAS, 2004).
- Workshops: Encontro onde há parte expositiva e demonstrações do objeto que gerou o evento. (CESCA, 1997).
- Encontros: Reunião de pessoas de uma mesma categoria profissional, com intuito de debater temas polêmicos apresentados geralmente por representantes dos grupos participantes. (MEIRELLES, 2003).
- Feira: Ampla, fixa e visa vender, onde expositor organiza a participação adquirindo um espaço físico transformando em um *stand* estando juntamente com outros expositores transformando em amplo e concorrido evento. (CESCA, 1997).
- Curso: Evento educativo, marcado pela apresentação de um tema específico visando o conhecimento, treinamento ou reciclagem dos participantes, capacitando os mesmos para exercer atividades relacionadas ao assunto proposto. (MEIRELLES, 2003).

Explicados os principais conceitos relacionados a eventos, seguido de seus tipos com suas principais características, seguindo para a análise e projeto orientado a objetos.

2.2 ANÁLISE E PROJETO ORIENTADO A OBJETOS

Análise e Projeto Orientados a Objetos – A&POO (*Object-Oriented Analysis and Desing*) segundo Braude (2005, p. 528) "é uma maneira de especificar e projetar aplicações OO (Orientado a Objetos). Sua principal característica é abordar o projeto utilizando termos que ocorrem naturalmente na aplicação".

2.2.1 Padrões de Projeto

Para Braude (2005, p. 158) "os padrões de projetos são combinações de classes e algoritmos associados que cumprem com propósitos comuns de projeto. Um padrão de projeto expressa uma ideia em vez de uma combinação fixa de classes. Os algoritmos associados expressam a operação básica do padrão."

Gamma (1998, p. 15) divide em três tipos as técnicas de projeto conhecidas como padrão de projetos. Sendo elas os padrões criacionais, estruturais e comportamentais.

2.2.1.1 Padrões de Projeto Criacionais

Braude (2005, p.163) define que os padrões de projetos criacionais auxiliam na criação do projetos de aplicações que envolvem coleções de objetos, permitindo a criação de várias coleções possíveis a partir de um único bloco de código. Tendo como propriedades a criação em tempo de execução de muitas versões da coleção e a restrição dos objetos criados.

Gamma (1998, p. 91) afirma que os padrões de criação abstraem o processo de instanciação, tornando um sistema independente de como os objetos serão criados, compostos e representados. Utiliza também como padrão de criação de classes a herança virando assim a classe que é instanciada.

2.2.1.2 Padrões de Projeto Estruturais

Segundo Braude (2005, p.242) “os propósitos do projeto de padrões estruturais são representar objetos complexos (o ponto de vista estático) e obter funcionalidades a partir deles de maneira a utilizar seus objetos agregados (o ponto de vista dinâmico)”.

Gamma (1998, p. 118) complementa que ao contrário de compor interfaces ou implementações os padrões de projetos também devem descrever maneiras de compor objetos para realizar novas funcionalidades.

2.2.1.3 Padrões de Projeto Comportamentais

Braude (2005, p.321) afirma que o propósito dos padrões de projeto comportamental é encapsular o comportamento entre objetos, permitindo múltiplas ações de comportamento em tempo de execução assim reutilizando este comportamento em outras aplicações ou codificá-lo efetivamente.

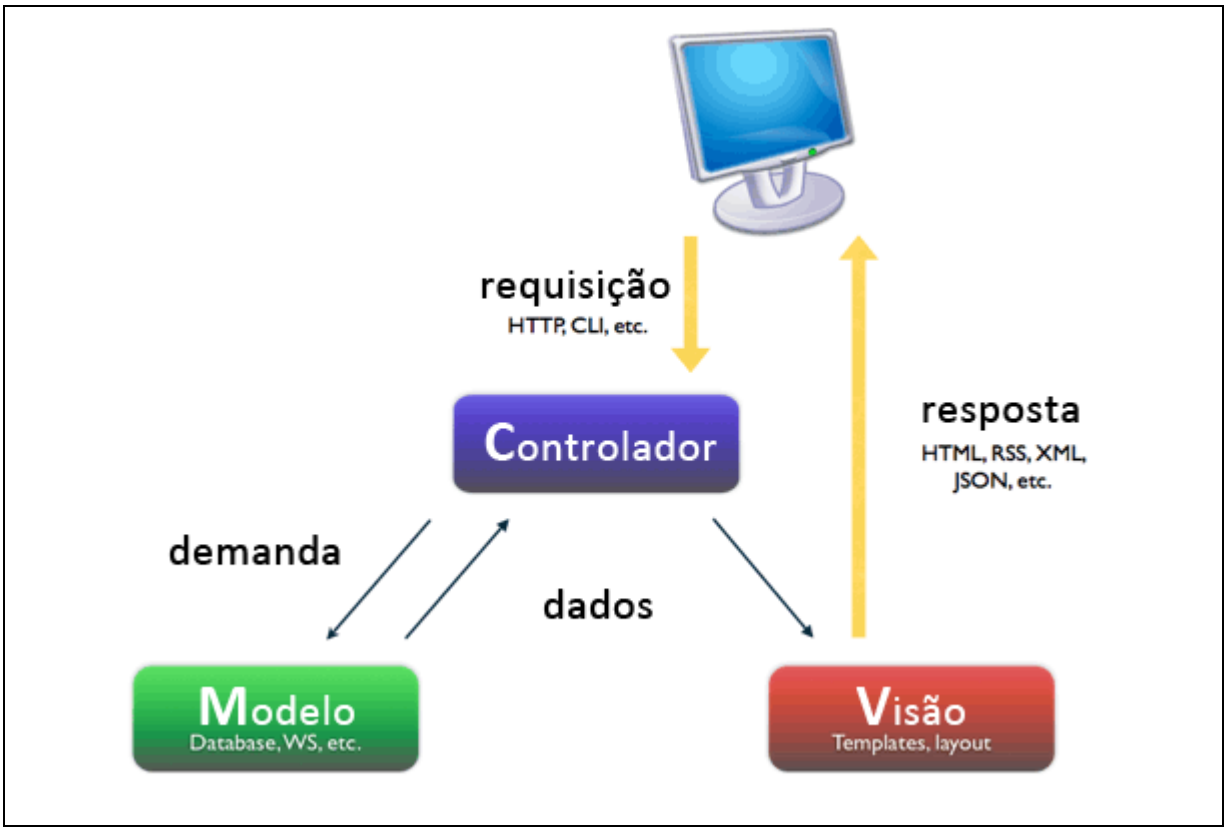
Gamma (1998, p. 188) define que os padrões de projetos comportamentais estão preocupados com algoritmos e responsabilidades entre os objetos e não descrevem apenas um padrão para objetos ou classes, mas também a comunicação entre eles utilizando assim a herança para distribuição do comportamento entre as classes.

2.2.2 Padrões de Arquitetura

Pressman (2006, p. 443) descreve a arquitetura de uma aplicação Web como uma infra-estrutura que possibilite o sistema a atingir os objetivos de negócio. Existem várias arquiteturas que podem ser usadas, no entanto, o autor sugere que seja usada uma arquitetura em três camadas nomeada MVC (Modelo, Visão e Controlador).

2.2.2.1 Modelo, Visão e Controlador (MVC)

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 final 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 pode retornar informação para o usuário. (GONÇALVES, 2007).



Esquema 1: Fluxo do modelo MVC
Fonte: Valente (2011).

De acordo com Marcoratti (2002), a divisão em camadas lógicas torna o sistema mais flexível, permitindo que cada camada possa ser alterada de forma independente. Este modelo de trabalho permite que a lógica do sistema seja separada em pacotes, o que reduz a dependência entre partes distintas, proporcionando que cada camada possa trabalhar de forma autônoma, facilitando o reuso dos componentes em vários sistemas. O autor ressalta que este modelo se tornou padrão em sistemas que rodam na Web.

2.2.3 Frameworks

Braude (2005, p. 566) explica que o objetivo mais importante no desenvolvimento de software é o reuso. Sendo assim, caso uma organização não consiga se beneficiar do investimento nas habilidades dos projetistas e programadores, utilizando várias vezes o trabalho dos mesmos, concorrentes acabam fazendo isso se antecipando com produtos superiores.

Appleton (1998 apud PRESSMANN, 2006, p. 203, grifo do autor) afirma que:

> Em alguns casos pode ser necessário fornecer uma infra-estrutura do esqueleto de implementação específica, chamada de arcabouço (*framework*), para o trabalho de projeto. O projetista pode selecionar uma "*miniarquitetura reusável* que fornece a estrutura e o comportamento genéricos para uma família de abstrações de software, dentro de um contexto(...) que especifica sua colaboração e uso em determinado domínio"

Braude (2005, p. 567, grifo do 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." Ainda afirma que um *framework* é um denominador comum para família de aplicações, surgindo geralmente após o desenvolvimento da segunda, terceira ou quarta aplicação.

Vistos os principais conceitos relacionados a padrões de projetos e arquitetura serão descritas as metodologias de desenvolvimento de aplicações, com uma abordagem relacionada ao desenvolvimento de aplicações web.

2.3 METODOLOGIAS PARA DESENVOLVIMENTO DE APLICAÇÕES WEB

Nos primórdios da internet os sites eram arquivos estáticos ligados por meio de hipertexto que eram mostrados usando poucos recursos visuais como texto e gráficos de baixa qualidade. Segundo Pressman (2006, p. 379) o HTML (*Hyper Text Markup Language*) foi crescendo e sendo integrado a novas ferramentas, XML (*Extensible Markup Language*) e Java, essas ferramentas proporcionaram aos engenheiros web o fornecimento de mais recursos computacionais junto da informação. O grande amadurecimento das ferramentas possibilitou o nascimento de sistemas e aplicações baseados em Web, hoje em dia estas aplicações evoluíram para ferramentas muito sofisticadas e complexas que são integradas com bancos de dados e aplicações de negócio.

Com base no grande crescimento das aplicações que fazem uso da internet, Pressman (2009, p. 379) sugere que projetos de grande escala que utilizam a Web como ambiente, tem a necessidade de ser desenvolvidos utilizando abordagens disciplinadas de engenharia, segundo ele novos métodos e ferramentas devem ser usados, levando em conta características especiais deste meio. Segundo o autor os modelos de processo que se encaixam neste cenário são os que adotam a filosofia de desenvolvimento ágil.

As metodologias ágeis de desenvolvimento têm como valores. Indivíduos e Interações mais que de processos e ferramentas, Software Funcional mais que documentação abrangente, Colaboração com o cliente mais que negociação de contratos e Responder a mudanças mais que seguir um plano. (BECK et al., 2001).

Trabalhando diretamente na satisfação do cliente através da entrega de software, onde mudanças são sempre bem vindas, independente da fase do desenvolvimento, os processos ágeis são formulados baseados nas mudanças que podem ocorrer durante o processo. O software é entregue ao cliente em um período que pode ir de semanas a poucos meses, dando preferência ao menor tempo. As pessoas relacionadas tanto às regras de negócio quanto desenvolvimento devem estar integradas, contornar problemas e achar soluções em conversas face a face. (BECK et al., 2001).

Dentre as várias metodologias ágeis existentes Koscianski e Soares (2006) citam que as mais conhecidas são *Scrum* e *Extreme Programming* (XP).

2.3.1 Scrum

Metodologia focada no projeto e desenvolvimento orientado a objetos. O *Scrum* aplica algumas ideias do controle de processos que ocorrem em indústrias ao desenvolvimento de software. Baseando-se nas premissas de flexibilidade, adaptabilidade e produtividade. O principal foco é achar uma forma de trabalho onde os participantes da equipe consigam produzir software de maneira flexível em ambientes mutáveis e dinâmicos. (KOSCIANSKI; SOARES, 2006).

As práticas do *Scrum*, de acordo com *Advanced Development Methods* (1996 apud PRESSMAN, 2006, p. 69), as equipes de trabalho devem ser pequenas, desta forma a comunicação entre a equipe é priorizada diminuindo a supervisão e aumentando a dissipação do conhecimento da equipe. O processo deve possibilitar alterações técnicas e de negócios, para que o produto seja produzido com maior qualidade. O software é entregue em pequenos incrementos que podem ser ajustados, testados documentados e expandidos em caso de necessidade.

O desenvolvimento é dividido em ciclos de geralmente trinta dias chamados *Sprints*. No início de cada *Sprint* a equipe trabalha definindo os requisitos do sistema para aquele ciclo. Ocorrem pequenas reuniões diariamente, onde é discutido o que aconteceu desde a última reunião, eliminando dificuldades ou impedimentos. Essa resolução diária de problemas faz com que seja possível conseguir agilidade no desenvolvimento. (KOSCIANSKI; SOARES, 2006).

2.3.2 Extreme Programming

Este metodologia é recomendada para ser usada em pequenas ou médias empresas, que desenvolvem aplicações onde os requisitos são vagos e modificados com muita frequência. A metodologia XP enfatiza o desenvolvimento rápido, com o maior foco na satisfação do cliente, favorecendo o comprimento das estimativas. (KOSCIANSKI; SOARES, 2006).

De Acordo com Wells (2009), Extreme Programming enfatiza o trabalho em grupo, ou seja, gerentes, clientes e desenvolvedores são parceiros para criar um time colaborativo. Esta metodologia preza um ambiente simples, porém efetivo, que possibilita ao time se tornar altamente produtivo. Este time auto-organizado resolve o problema da forma mais eficiente possível.

O código desenvolvido na metodologia XP deve ser limpo e enxuto, com o menor número possível de componentes, sem se preocupar com requisitos futuros. Entre as práticas do XP está a programação em pares, onde o código é criado em dupla, projeto simples e propriedade coletiva, ou seja, todos os desenvolvedores da equipe devem ter o mesmo conhecimento do código, tendo liberdade para agregar conhecimento ao código, isso é proporcionado pelo uso de outra prática, a padronização de código. (KOSCIANSKI; SOARES, 2006).

Após terem sido abordados os conceitos principais de metodologias de desenvolvimento de aplicações Web, serão esplanadas as principais tecnologias relacionadas ao desenvolvimento deste tipo de aplicações.

2.4 TECNOLOGIAS

Nesta seção serão abordados tópicos relacionados as principais tecnologias utilizadas no desenvolvimento de aplicações web e que compõem o *framework Demoiselle*.

2.4.1 Java

Em maio de 1995 foi lançada a linguagem Java, uma linguagem que trabalharia com sites produzidos para a web. Um grande atrativo da linguagem é o fato de ser portável para vários sistemas operacionais. A aceitação de Java no mercado cresceu rapidamente, pois foi lançada em épocas em que a web estava em ascensão e a linguagem possibilitava a utilização de vários recursos, como animações que até então não eram possíveis de fazer nas páginas existentes. (GONÇALVES, 2007).

De acordo com Gonçalves (2007, p. 07), referindo-se a Java:

> O Java amadureceu e com as mudanças ocorridas no mundo, muitas implementações foram criadas, com o intuito de abranger essas mudanças. Hoje você pode estar usando tanto aplicativos para desktop, páginas para a internet ou até mesmo aplicativos pequenos em celulares. Todos criados com a linguagem Java.

A grande aceitação de Java no mercado fez com que vários *frameworks* que facilitam e automatizam o processo de desenvolvimento nesta linguagem fossem desenvolvidos. A plataforma Java está dividida em quatro áreas (Quadro 1), cada uma específica para um ambiente. (FURGERI, 2010).

Divisão da Arquitetura Java	
JSE **(*Java Standard Edition*)**	Considerada a base da linguagem, foi desenvolvida para ser executada em máquinas simples e estações de trabalho.
JEE **(*Java Enterprise Edition*)**	Direcionada ao desenvolvimento de aplicações que fazem uso de servidores trabalhando em redes e internet.
JME ***(Java Micro Edition*)**	Projetada para dispositivos com baixo poder computacional, como dispositivos móveis e celulares.
JavaFX	Plataforma criada para suportar o desenvolvimento de aplicações ricas, que são interfaces muito dinâmicas. Oferece suporte a qualquer biblioteca de Java, entre eles estão desktops, navegadores de internet, celulares, TVs, entre outros.

Quadro 1: Divisão da arquitetura Java
Fonte: Adaptado de Furgeri (2010).

A arquitetura JEE possibilita a criação de aplicações de negócio e *websites* dinâmicos. Estas aplicações podem ser divididas em camadas, requisito fundamental para este tipo de aplicação. Estas camadas dividem-se em cliente de apresentação, que pode ser um navegador web ou uma aplicação JSE, também pode existir a camada de apresentação no lado servidor. Esta camada constrói a interface do cliente, no caso de ser uma aplicação Web, a camada de lógica de negócio é onde está contida toda a lógica da aplicação, como cadastros, condições para realizar ações, etc. Por fim, pode-se citar a camada de modelo de domínio. Esta camada inclui toda a lógica para interação com o banco de dados e com a lógica de negócios, como inserção, remoção, procura e atualização de registros. (CRAWFORD; KAPLAN, 2003).

A biblioteca de classes do JEE é formada por uma arquitetura baseada em componentes, dentre eles estão os *Servlets*, que são classes Java que estão armazenadas em *containers* JEE, podem receber requisições HTTP feitas ao servidor Web e responder a elas gerando HTML de forma dinâmica. Pode-se citar também a *JavaServer Pages* (JSP) que são muito similares ao conceito de *Servlets*, mas a diferença é que o código Java é embutido em um documento HTML. Quando uma JSP é requisitada o servidor compila o código deste arquivo para uma *Servlet* que retornará algum conteúdo para o usuário, este tipo de componente é usado para construir a interface da aplicação. (CRAWFORD; KAPLAN, 2003).

Como ponto central do JEE pode-se citar os *Entrerprise JavaBeans* que são o modelo de componentes para desenvolver regras de negócio, estes componentes podem ser

configurados para, manter ou não, um estado durante a interação com um cliente. JEE também contém em seu núcleo extenso suporte a XML, além disso, suporta a chamada de procedimentos remotos baseados em XML. Outro ponto forte desta plataforma é a integração e criação de *webservices*, na maioria das vezes através de XML, com a possibilidade de descrever, registrar, procurar e invocar objetos de serviços através da Web. (WEAVER; MUKHAR; CRUME, 2004).

Esta plataforma define um grande número de possibilidades que facilitam o desenvolvimento de aplicações empresariais, que necessitam de uma infra-estrutura bem formada, necessária para desenvolver aplicações que devem ser escaláveis, robustas, seguras e de fácil manutenção. (WEAVER; MUKHAR; CRUME, 2004).

2.4.2 *Java Persistence API* (JPA)

Bauer e King (2005, p. 5) definem persistência como a ação de armazenar dados em bancos de dados relacionais utilizando a linguagem SQL (*Structured Query Language*), e que Mapeamento Objeto-Relacional (ORM) é a persistência de objetos acontecendo de forma transparente e automatizada para tabelas do banco de dados relacional, mapeamento este, que é feito usando metadados que ficam localizados na aplicação e descrevem as tabelas.

Dall'Oglio (2009, p. 222) afirma que a crescente adoção do desenvolvimento orientado a objetos incentivou o surgimento de bancos de dados que utilizam este paradigma para persistir dados, porém o baixo desempenho, fez com que os modelos relacionais de bancos de dados continuassem sendo o mais viáveis, pelo fato de que suas ferramentas já eram aceitas e detinham um alto nível de maturidade, também pelo fato de ser a maior fatia no mercado de bancos de dados.

Ainda segundo o autor, 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 de armazenar objetos em bancos de dados relacionais de forma simples, utilizando-se de vários padrões (*design-patterns*) e também alguns *frameworks* que podem desempenhar este papel de forma automatizada.

Apesar de o SQL estar presente em muitas aplicações Dall'Oglio (2009, p. 222) afirma que muitos programadores usam o SQL de forma errada, segundo o autor não há

entendimento suficiente, ou o uso dos recursos acontece de forma incorreta, o que leva a problemas de performance e manutenção, como também é citado por Bauer e King (2005, p. 37). Entre as vantagens de usar uma ferramenta para ORM pode-se destacar a abstração de código SQL, que entre meio ao código Java pode se tornar confuso para o desenvolvedor. Ainda seguindo o que foi citado pelo autor a criação de código relacionado com a persistência de dados pode ser a mais tediosa no desenvolvimento de software, uma ferramenta ORM permite que se gaste o tempo que gastaria escrevendo SQL, trabalhando na estratégia de negócios.

Em Java existem vários *frameworks* que podem ser utilizados para realizar a persistência de objetos, alguns pagos, outros não, porém a maioria trabalha de forma não padronizada, o que torna a aplicação muito dependente da tecnologia de persistência. A tecnologia JPA não é um *framework* de persistência propriamente dito, é 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*.

Dentre as vantagens de usar JPA, Google Code (2010), descreve a independência do banco de dados.

> A JPA (API persistente Java) é uma interface padrão para armazenar objetos de dados em um banco de dados relacional. O padrão define as interfaces para anotar objetos Java, recuperar objetos com consultas e interagir com um banco de dados usando transações. Um aplicativo que usa a interface JPA pode funcionar com bancos de dados diferentes sem usar qualquer código específico do fornecedor do banco de dados. A JPA facilita o transporte do seu aplicativo entre bancos de dados de fornecedores diferentes.

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 a aplicação. (KEITH; SCHNICARIOL, 2009, p. 11).

2.4.3 JavaServer Faces (JSF)

As aplicações Web geralmente são compostas por duas partes principais, a apresentação e a lógica de negócios. A apresentação é referente à aparência, ou seja, a marcação HTML que especifica o layout da aplicação, como: imagens, links, fontes, etc. A

lógica de negócios por sua vez é desenvolvida utilizando alguma linguagem de programação, como Java, que determina o comportamento da aplicação. Algumas aplicações misturam essa lógica com a marcação HTML. (GEARY; HORSTMANN, 2005, p. 7).

JSF é um *framework* Java, que simplifica o desenvolvimento de interfaces do usuário, parte complexa no desenvolvimento de aplicações Web. (BURNS; SCHALK, 2010, p. 3).

Segundo Constantino (2008), é uma arquitetura poderosa que suporta o desenvolvimento de aplicações de grande porte com qualidade, além disso, de acordo com Pitanga (2004) implementa características de um *framework* MVC desenvolvido para Web, o que é uma grande vantagem, pois há clara separação entre visualização e regras de negócio.

De acordo com Geary e Horstmann (2005, p. 21) para separar a camada de aplicação da lógica de negócios JSF utiliza *Beans*, que conforme Flanagan (2006, p. 300) são componentes de software reutilizáveis que podem ser manipulados com uma ferramenta de desenvolvimento. Geary e Horstmann (2005, p. 21) explicam que em Java um *bean* é simplesmente uma classe que segue certas convenções de código. Dentre as convenções estão: manter um construtor padrão na classe e seguir um padrão para nomenclatura de métodos de acesso as propriedades. Para cada propriedade existirá um par de métodos, um para inserir valor, e outro para recuperar. Além dos métodos usados para manipular as propriedades um *bean* pode conter outros métodos.

Página JSF renderizada

Executar Ação

Código da página JSF

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Aplicação JSF</title>
    </h:head>
    <h:body>
        <h:form>
            <h:inputText value="#{meuBean.propriedade}" />
            <h:commandButton action="#{meuBean.executarAcao()}"
                             value="Executar Ação" />
        </h:form>
    </h:body>
</html>
```

Bean de ações

```
public class MeuBean {
    private String propriedade;

    public MeuBean() {
    }

    public String getPropriedade() {
        return propriedade;
    }

    public void setPropriedade(String propriedade) {
        this.propriedade = propriedade;
    }

    public void executarAcao(){

    }
}
```

Listagem 1: Implementação de um *Bean* integrado a uma página JSF
Fonte: Os autores (2012).

O *framework* JSF foi modelado para simplificar o desenvolvimento de aplicações por meio de componentes de interface com o usuário que são conectados a componentes de lógica

de negócio. A utilização deste *framework* é recomendada pela *Sun Microsystems* para o desenvolvimento Web com Java atualmente. (GONÇALVES, 2008).

2.5 FRAMEWORK DEMOISELLE

O *Demoiselle Framework* é construído a partir do conceito de *framework* integrador, integrando diversas ferramentas utilizadas no mercado Java. Tem como objetivo 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 comum em uma aplicação. Entre eles estão a arquitetura, segurança e configuração. (SACRAMENTO et al., [2011]).

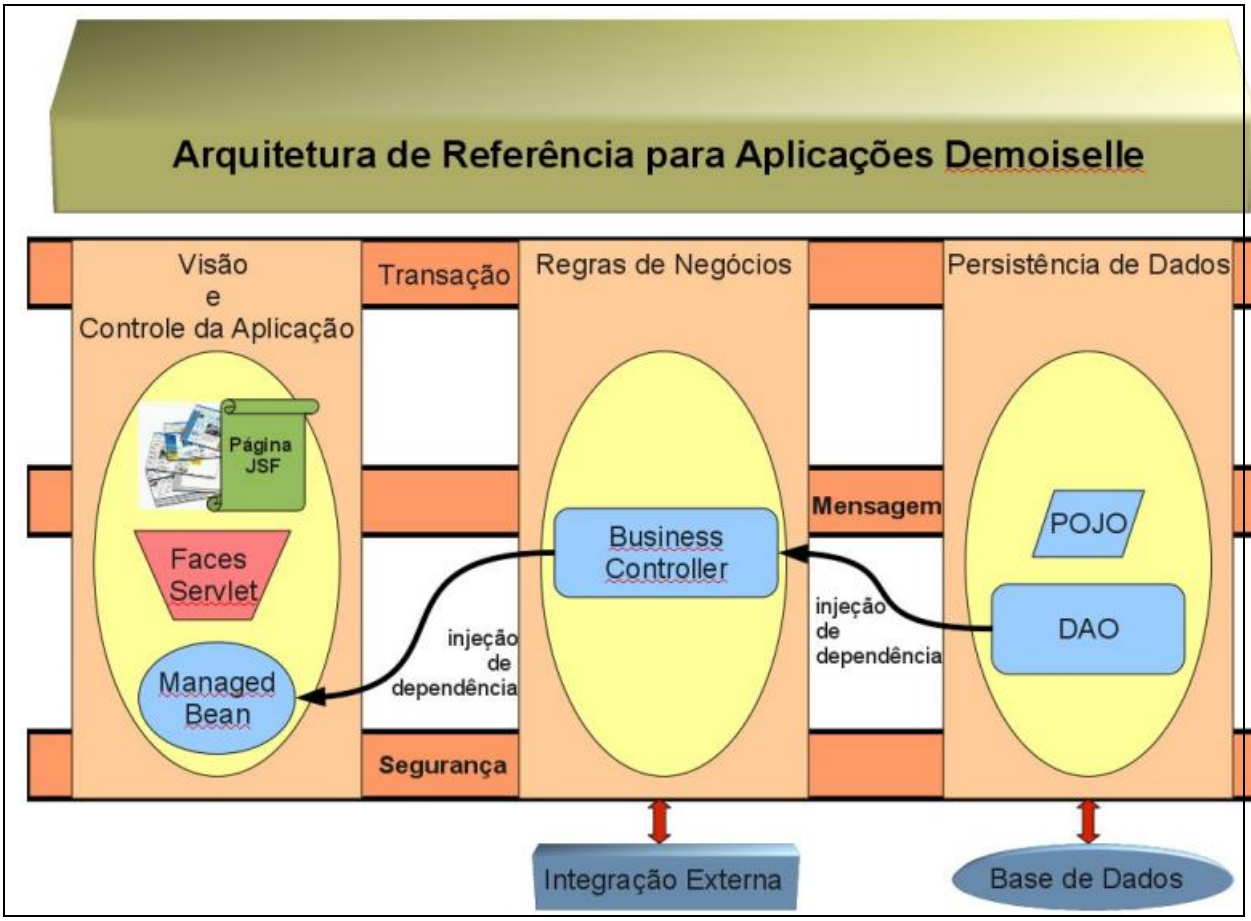
O *framework* contém uma estrutura não monolítica, ou seja, as funcionalidades estão separadas do núcleo principal, esta forma de organização permite que aplicações específicas não necessitem compor dependências que não serão usadas. (SACRAMENTO et al., [2011]).

A estrutura do *Demoiselle* é dividida em *Core*, que contém as funcionalidades que são comuns a todas 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 estar no núcleo. Por fim os componentes, que são artefatos independentes do núcleo, não precisam estender as funcionalidades do *core*, têm ciclo de vida próprio, não precisam necessariamente fazer 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* baseia-se em especificações que são definidas pela comunidade Java. (LISBOA, 2010).

Apesar de o *framework* não forçar o desenvolvedor a utilizar nenhum tipo de arquitetura, o *demoiselle* fornece uma arquitetura que pode ser usada como referência (Esquema 3) 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 DAO (*Data Access Objects*) que é um padrão usado em Java para auxiliar na persistência de dados. Na parte de visão podem ser citadas as páginas JSF (*JavaServer Faces*), e por fim pode

ser citado o Controlador que são os *Managed Beans* (Beans Gerenciados) que interagem com a Visão e são integrados aos *Business Controllers*, que tem como objetivo efetuar a lógica de negócios da aplicação. (SACRAMENTO et al., [2011]).



Esquema 2: Estrutura Geral
Fonte: *Demoiselle Framework* (2010).

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.

2.6 GESTÃO DO PROJETO

Agora além de seus conceitos serão apresentado algumas tecnologias utilizadas para o alcance dos objetivos.

2.6.1 Controle de versão

Segundo Pressman (2006, p. 608), controle de versão é uma combinação de procedimentos e ferramentas para gerir diferentes versões de objetos de configuração. Define quatro capacidades para um sistema de controle de versão, um banco de dados de projeto guardando todos os objetos de configurações necessários; uma capacidade de gestão de versão guardando todas as versões para que seja possível construir uma versão usando diferenças de versões anteriores; uma facilidade de construir que permita ao engenheiro de software coletar todos os objetos de configurações necessárias para gerar através deles uma versão específica do software e por último devem implementar a capacidade de acompanhamento de tópicos ou *bugs*, para registrar e acompanhar o estado dos tópicos associados a cada objeto de configuração.

Sendo assim, o *Apache Subversion* segundo Collins-Sussman, Fitzpatrick e Pilato ([20--], p. 18) é um sistema livre para controle de versão, gerenciando arquivos e diretórios e as alteração realizadas ao longo do tempo. Permite também que se recupere versões antigas e examine o histórico de alterações.

2.6.2 Gestão de configuração de software (SCM)

Pressman (2006, p. 600) informa que a "gestão de configuração de software é um conjunto de atividades para administrar modificações ao longo do ciclo de vida do software de computador". A SCM (*Software Configuration Management*) pode ser entendida como uma atividade para garantia de qualidade de software.

Sonatype (2008) define o *Apache Maven* como uma ferramenta de gerenciamento de projeto que compreende um modelo de objeto do projeto, um conjunto de normas, um ciclo de

vida, um sistema de gerenciamento de dependências e uma lógica para execução de *plugins* definidos no ciclo de vida do projeto.

2.6.3 Teste unitário

Pressman (2006, p. 295) explica que "O teste de unidade focaliza o esforço de verificação na menor unidade de projeto do software ." Assim, baseado em uma descrição do projeto no nível de componentes os caminhos de controles são testados com a finalidade de descobrir erros dentro dos limites do módulo.

Paula Neto ([20--]) define que *JUnit* "é um framework open-source, criado por Eric Gamma e Kent Beck, com suporte à criação de testes automatizados na linguagem de programação Java". Visando a intenção de facilitar a criação de um código para a automação de testes unitários e apresentação de resultados. Testando cada método de uma determinada classe com a apresentação de possíveis erros ou falhas.

Assim, conforme citado acima segue demonstração de uso da ferramenta no presente projeto (Listagem 2), onde há duas situações de teste, no quadro da esquerda superior onde ocorre o erro, não permitindo a compilação do sistema. Já no quadro superior direito o teste é executado sem erros, compilando o projeto.

Listagem 2: Caso de teste Cadastro de Evento
Fonte: Os autores (2012).

O desenvolvimento é uma tarefa que está muito suscetível a erros, para que estas falhas possam ser corrigidas devem ser realizados testes. Os testes podem ser manuais, através de ferramentas de depuração, ou através de ferramentas para realizar testes automatizados. As ferramentas de testes automatizados possibilitam que um teste, uma vez escrito, possa sempre ser executado, diminuindo o tempo gasto com testes.

2.6.4 Gerenciamento do Projeto

Para Costa e Guedes (2012), "Atualmente uma ferramenta que faça a gerência dos projetos de uma empresa reflete significativamente no seu desempenho em cada projeto e também no seu sucesso perante o mercado". Sendo assim, define o *dotProject* como um *software* capaz

de suprir as necessidades em empresas de TI devido a diversidade de funcionalidades e facilidade na utilização.

Quanto ao *dotProject*, o mesmo foi citado como o melhor e mais usado software para gerenciamento de projetos, pois através dele a equipe de desenvolvimento tem acesso a tudo tendo em vista que o mesmo é *on-line* e engloba várias utilidades. (CENTRO DE DIFUSÃO DE TECNOLOGIA E CONHECIMENTO, 2012).

Assim, no desenvolvimento conforme visto acima uma das funcionalidades disponível é o gráfico de *Gantt* (Gráfico 1), utilizado no presente projeto para demonstrar o avanço das diferentes etapas do projeto.

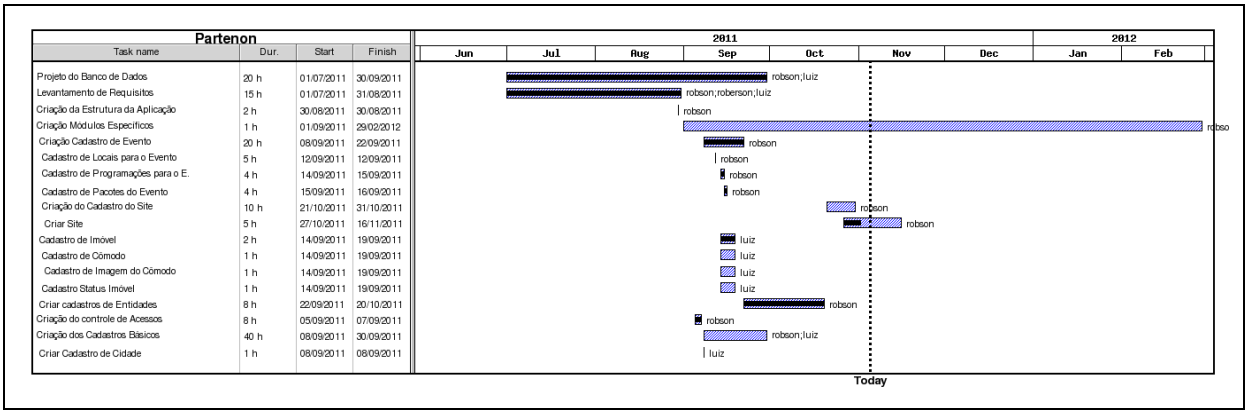


Gráfico 1: Gráfico de *Gantt* - Andamento do Projeto (*dotProject*)
Fonte: Os autores (2012).

Assim, para a gestão no desenvolvimento utilizou-se diversas tecnologias para facilitar a gerência e acompanhamento do até então desenvolvido e o que está pendente desenvolver.

3 DESENVOLVIMENTO

O trabalho teve como objetivo o desenvolvimento de um *software* para controle e gestão de eventos através do *framework Demoiselle* 2.0. A seguir serão descritos os meios envolvidos no desenvolvimento.

Primeiramente realizou-se um estudo sobre as tipologias de eventos e os processos que compõem os mesmos. Também, realizamos uma reunião com os futuros usuários com a finalidade de entender suas necessidades e assim definidos os requisitos do sistema, iniciando a modelagem do banco de dados.

Posteriormente foi realizado um estudo sobre a linguagem Java e o *Framework Demoiselle* na sua versão 2.0 assim como os *frameworks* que estão acoplados junto ao mesmo, como o JSF para desenvolvimento da interface com usuário; *Maven* como ferramenta para gerenciamento de automação de projetos em Java e *JUnit* como ferramenta para automatização de testes.

No desenvolvimento do projeto utilizamos a IDE *(Integrated Development Environmen*t) *Netbeans*, ferramenta com suporte a Java e ao *Demoiselle 2.0*.

Para realizar a persistência dos dados foi utilizado o padrão JPA já embutido no *framework* e como Sistema Gerenciador de Banco de Dados (SGBD) utilizamos o *PostgreSQL* 9.0, sendo caracterizado por ser gratuito e robusto. A modelagem da base de dados foi elaborada com *DB Designer Fork*, que permitindo a exportação do script para criação da própria base de dados no SGBD.

Para disponibilizarmos a aplicação ao usuário final necessitamos que a mesma esteja rodando em um servidor Java. Assim, utilizamos o servidor de aplicação *Apache Tomcat* 7 sendo um do servidores mais utilizados em aplicação Java.

A verificação inicial da aplicação foi efetuado através da ferramenta *JUnit*, que vem acoplada ao *framework Demoiselle* e em seguida validada junto aos usuários.

Para o controle e gerenciamento de versões no trabalho em equipe foi utilizado SVN *Subversion* e na gerência do projeto o *dotProject.*

Como metodologia de desenvolvimento foi realizada uma mescla entre alguns conceitos de SCRUM e XP que mais se adequaram as características de realização e execução do trabalho.

3.1 METODOLOGIA DE DESENVOLVIMENTO

Pelo fato do *Scrum* e *XP* se tratarem de metodologias ágeis para gestão do *software* além do projeto possuir características muito específicas e também o ambiente trabalhado não é de total desenvolvimento, optou-se por uma mescla das duas metodologias adaptadas ao projeto.

Com *Scrum*, a adoção de tarefas no *dotProject* foi tratado como *Product Backlog*, sendo uma lista de funcionalidades desejadas para o *software.*

Já com *XP*, em muitos mementos a programação em par permitiu maior qualidade no código não alterando o tempo de entrega, pois diminui as correções e reduz os erros.

3.1.1 Modelagem e Diagramas

Em reunião realizada com os dirigentes do Moto Grupo Cães do Asfalto de São Miguel do Oeste - SC, através de conversa informal, foi observado que não há uma política documentada de controle dos processos do evento. Estas práticas impossibilitam a geração de relatórios, bem como a análise de dados para auxiliar a tomada de decisões em eventos futuros.

Nesta conversa foram elencados os seguintes requisitos:

- Criação de portal onde será possível visualizar notícias e páginas que serão cadastradas no sistema e atualizadas dinamicamente.
- No portal possibilitar aos participantes a visualização de imóveis para hospedagem, estes imóveis devem ser cadastrados no sistema.
- Criação de módulo para controle das movimentações financeiras.
- Módulo para controle da locação de stands.

Após a definição dos requisitos foi iniciado o desenvolvimento, porém algumas ideias sobre o funcionamento do sistema ficaram abstratas. Para simplificar o entendimento da equipe sobre a aplicação foram elaborados casos de uso que foram utilizados para criar diagramas de caso de uso. Segue baixo dois casos de uso (Quadro 2 e Quadro 3).

PARTENON	**Partenon** *Gestão Integrada de Eventos*

Caso de Uso Cadastro de Evento

Ator Primário	Usuário do Sistema
Escopo	Iniciar o Gerenciamento de um evento
Nível	Objetivo do Usuário
Stakeholders	Organização do Evento
Pré Condição	- Estar logado no Sistema; -Ter acesso ao módulo de eventos; - Possuir um município cadastrado.
Cenário Principal	1. Selecionar Módulo de eventos; 2. Clicar em Cadastrar Novo Evento; 3. Informar dados Gerais do Evento; a. Nome b. Descrição c. Data de Início d. Data de Término 4. Informar Local sede do Evento; a. Descrição b. Rua c. Bairro d. Cidade e. Número f. Cep g. Complemento 5. Clicar em Salvar; 6. Sistema libera opções para gerenciar o evento.
Extensões	4.1 Cidade não foi informada; 4.1.1 Sistema gera mensagem de erro solicitando que a cidade seja informada. 4.1.2 Usuário aborta o processo e cadastra uma cidade.
Variações	Sem variação

Quadro 2: Caso de uso cadastrar evento
Fonte: Os autores (2012)

Da mesma forma, além do caso de uso cadastrar evento (Quadro 2) foi criado o caso de uso cadastro gerar site (Quadro 3) que demonstra a forma como deverá ocorrer a geração do site.

PARTENON	**Partenon** *Gestão Integrada de Eventos*
Caso de Uso Gerar Site	
Ator Primário	Usuário do Sistema
Escopo	Gerar um site com as informações do evento
Nível	Objetivo do Usuário
Stakeholders	Coordenador de Marketing do evento
Pré Condição	Evento estar cadastrado
Cenário Principal	1. Selecionar o Módulo de Eventos 2. Selecionar o evento ao qual o site se refere 3. Acessar o menu "Site" 4. Informar configurações para o site a. Caminho da url desejado para acessar o Site; b. Disponibilidade das inscrições pelo site ; c. Disponibilidade de visualização dos imóveis; d. Informar Dados do Twitter. 5. Clicar em Salvar
Extensões	Sem extensões
Variações	4' Com ou sem Dados do Twitter.

Quadro 3: Caso de uso gerar site
Fonte: Os autores (2012)

Como resultado do caso de uso cadastrar evento (Quadro 2) foi desenvolvido o diagrama cadastrar evento (Diagrama 1).

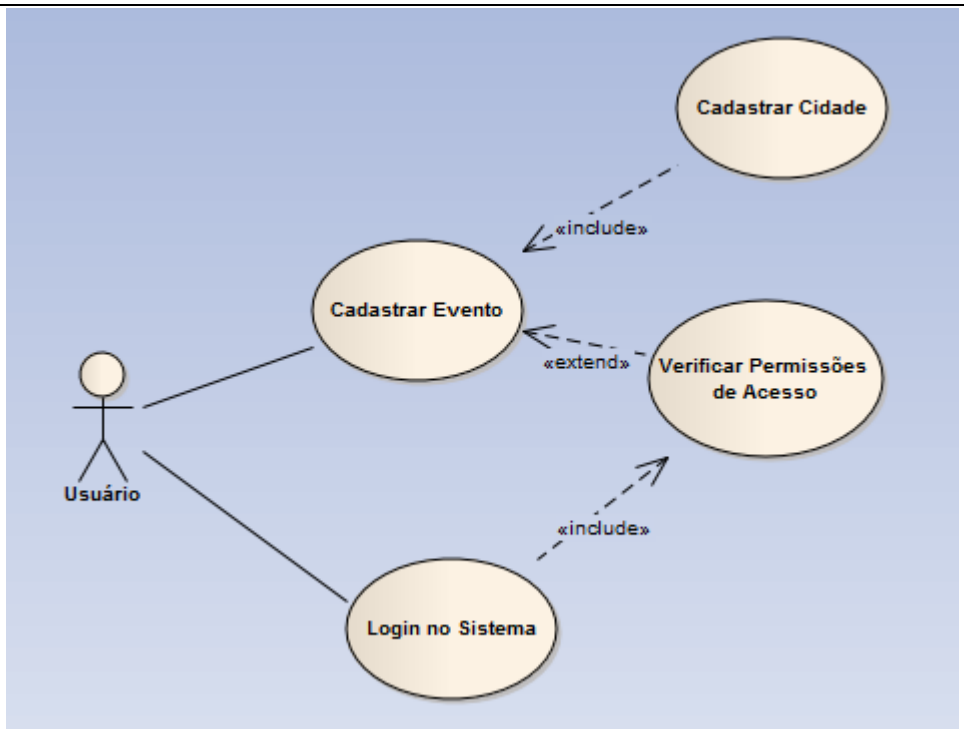


Diagrama 1: Diagrama do caso de uso cadastrar evento
Fonte: Os autores (2012).

Na seqüência, baseado no caso de uso gerar site (Quadro 3), foi criado o diagrama gerar site (Diagrama 2) que ilustra o respectivo caso de uso.

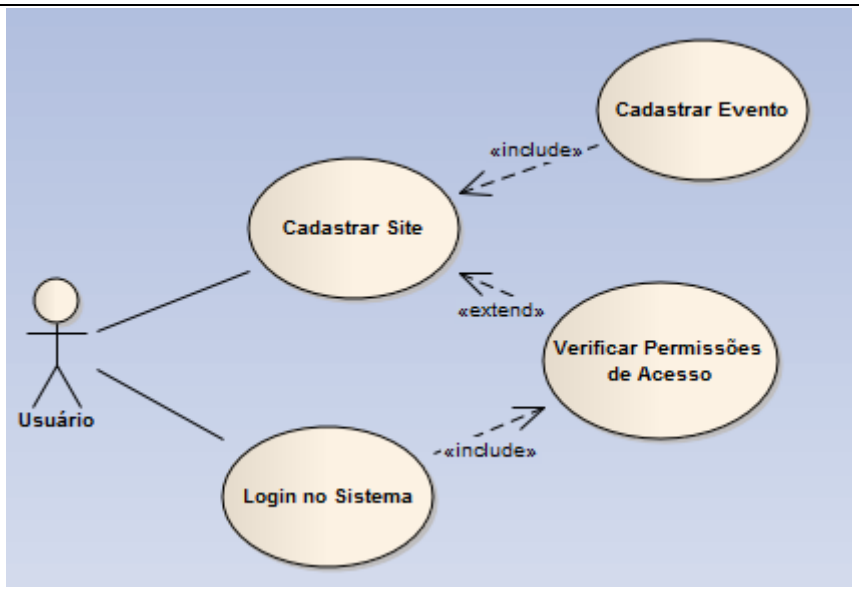


Diagrama 2: Diagrama do caso de uso gerar site
Fonte: Os autores (2012).

Além dos Diagramas de Caso de Uso foi utilizado outro diagrama da UML (*Unified Modeling Language*), o diagrama de classes, que é muito utilizado no desenvolvimento de sistemas orientados a objetos. Os diagramas de classe podem ser utilizados para gerar as classes que compõem o sistema minimizando o esforço desnecessário no desenvolvimento do projeto. O Diagrama 3 é uma parcial do núcleo do sistema.

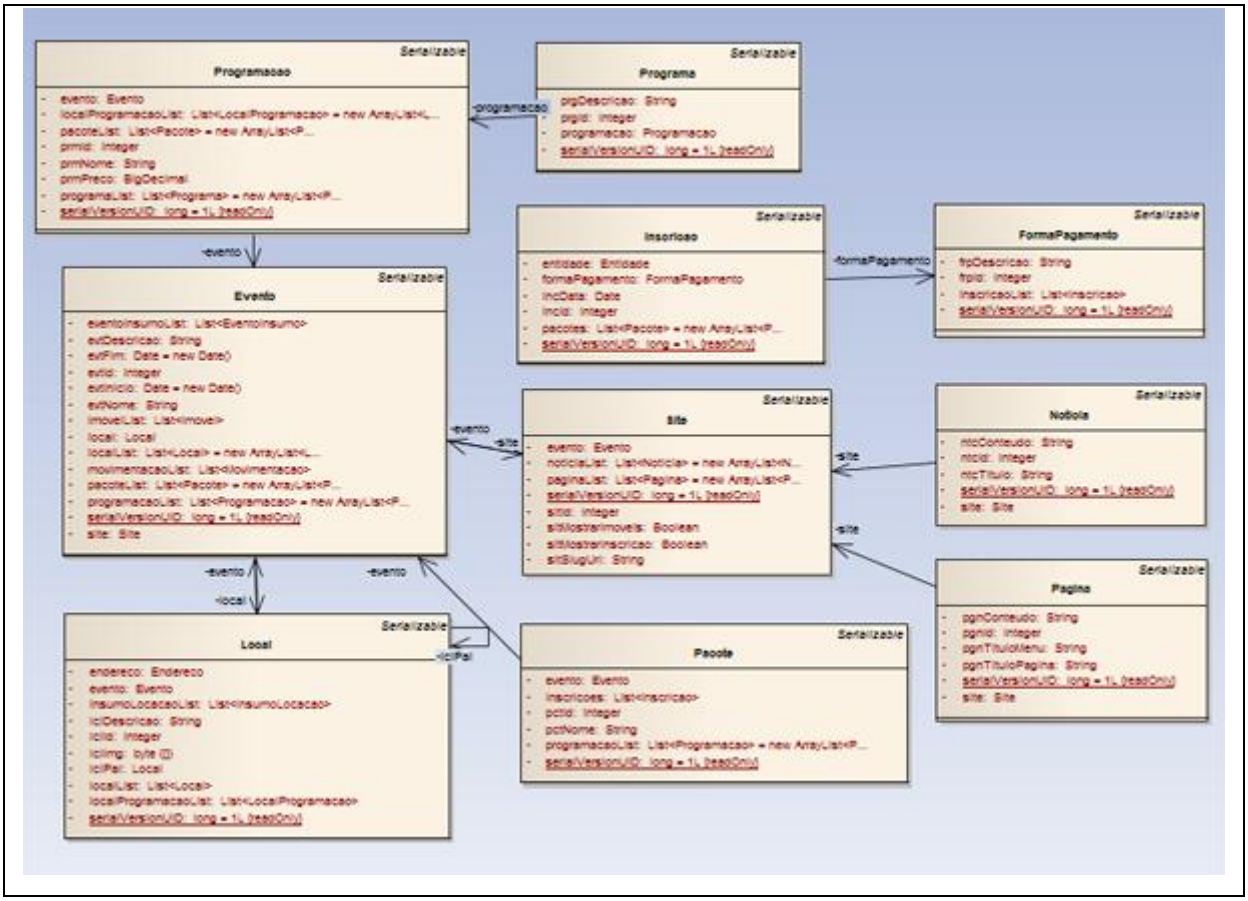


Diagrama 3: Diagrama de Classes do núcleo do evento
Fonte: Os autores (2012).

Para a modelagem de dados foi utilizado o diagrama de Entidade/Relacionamento elaborado através da ferramenta DB Designer Fork versão 4, a partir deste diagrama foi gerado o *script* SQL para a criação da base de dados e com ela foi feito o mapeamento para as classes do sistema. Segue resumo do modelo Entidade/Relacionamento (Diagrama 4).

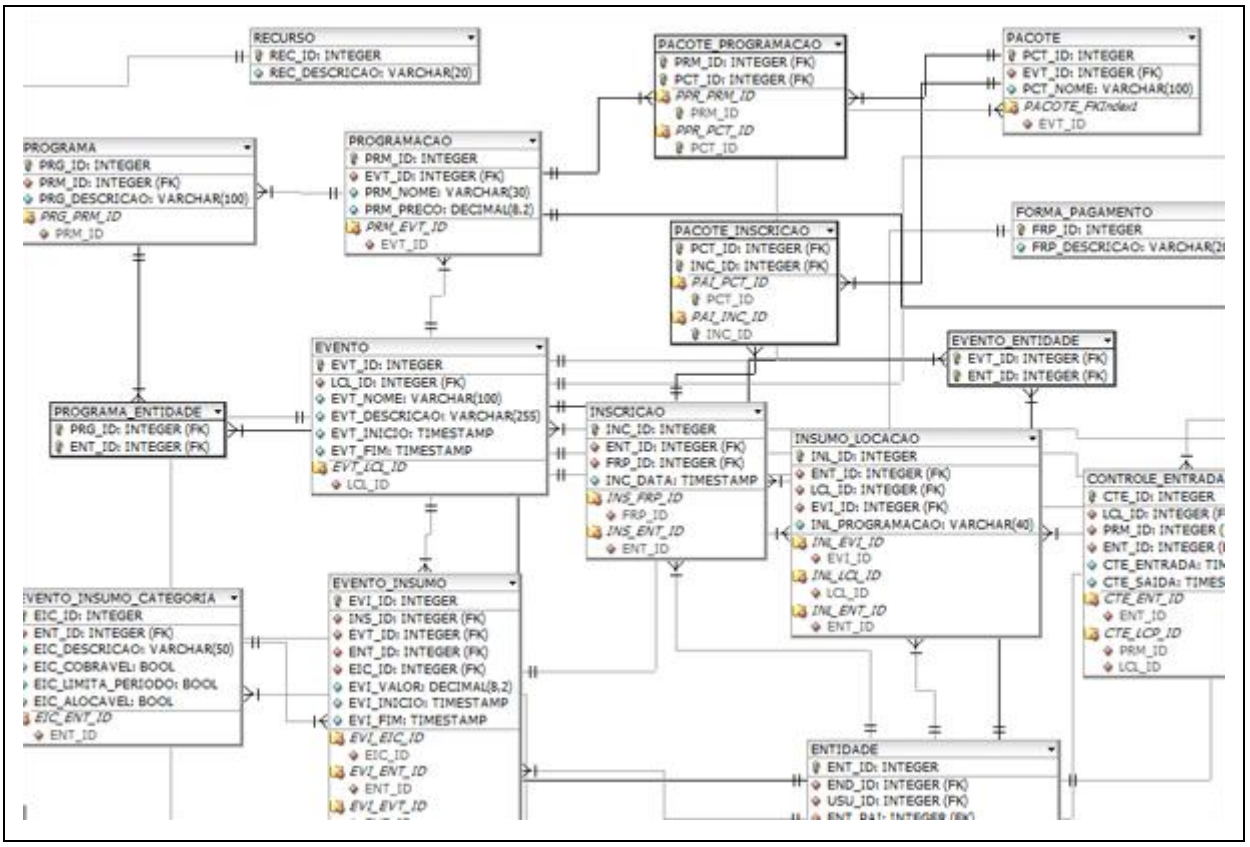


Diagrama 4: Modelo Entidade/Relacionamento parcial do sistema
Fonte: Os autores (2012).

Juntamente com os demais digramas elaborados previstos pela UML, também foi criado o Diagrama de Implantação (Diagrama 5). Com ele é representado à configuração e arquitetura do projeto.

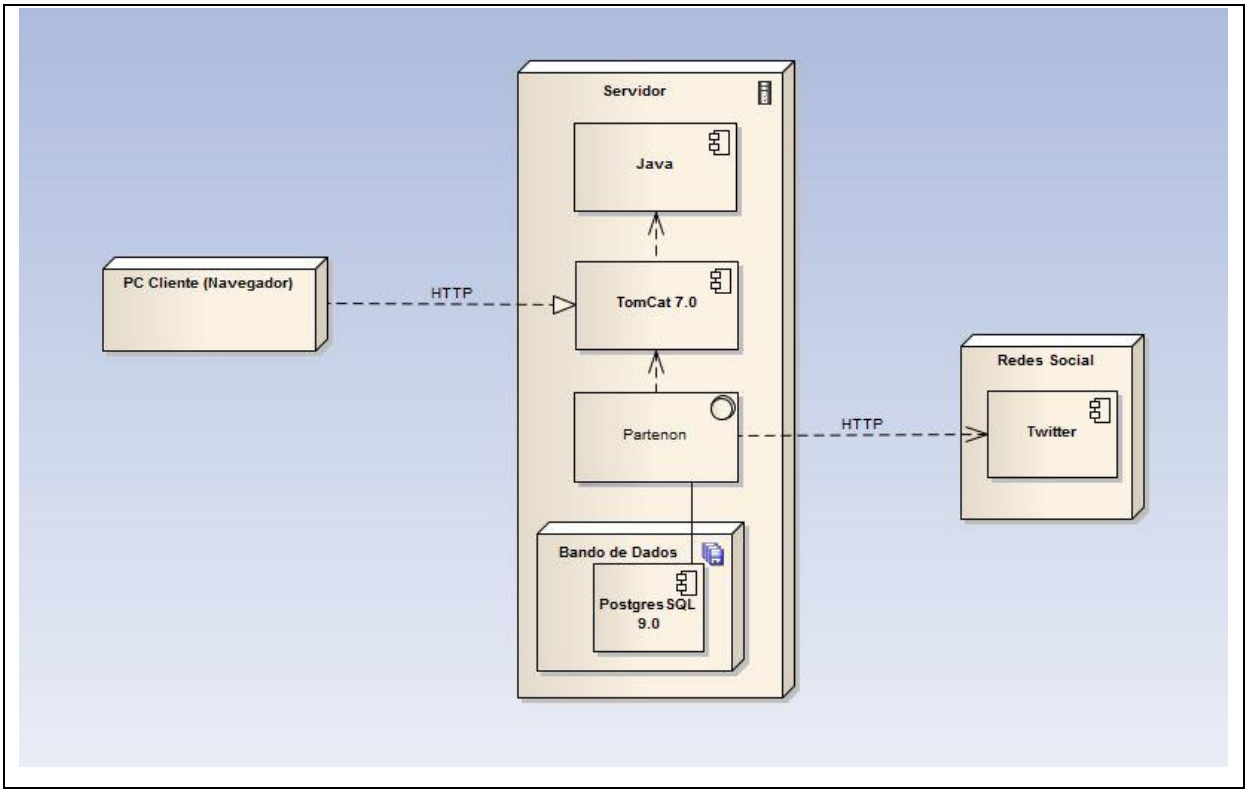


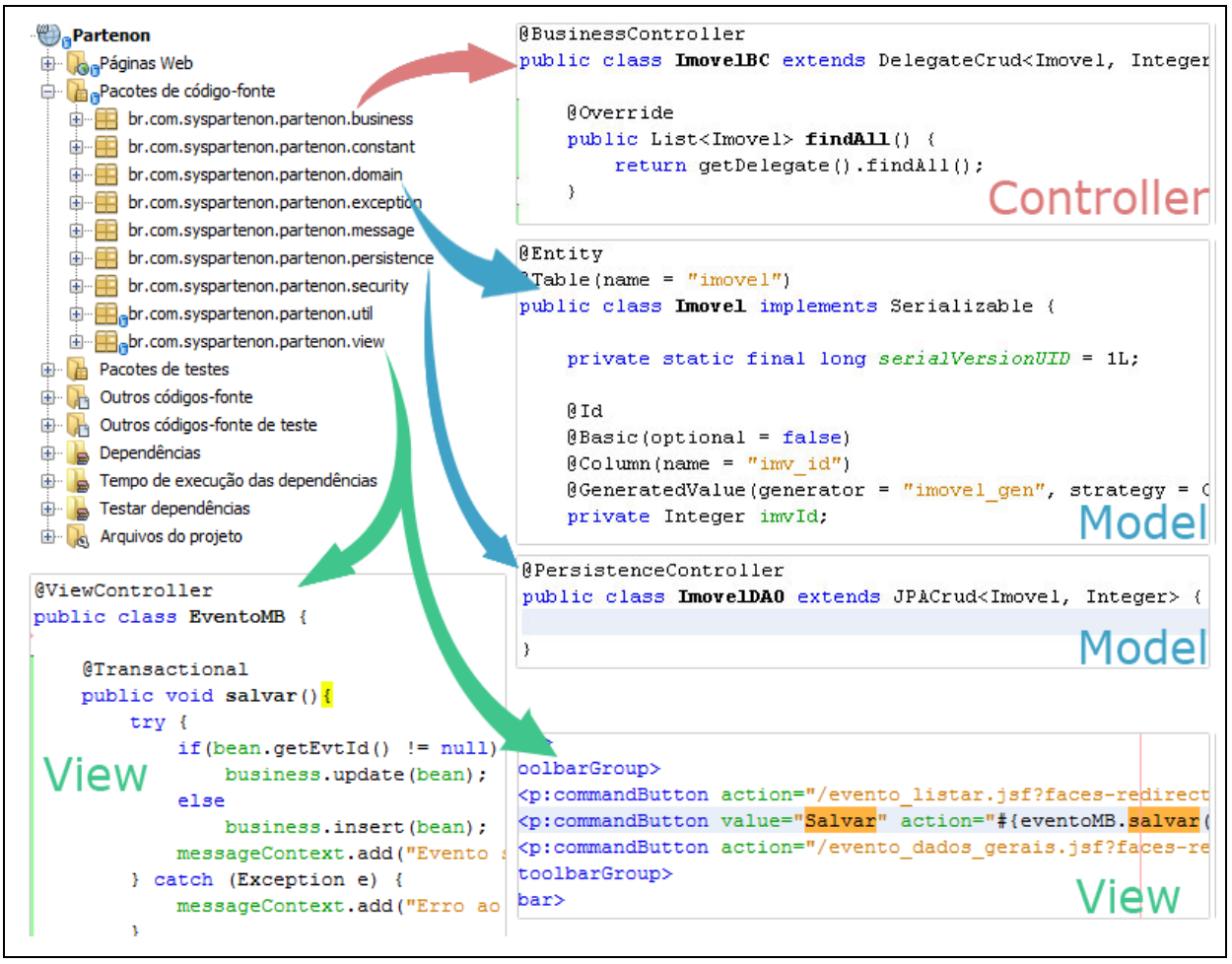
Diagrama 5: Diagrama de Implantação
Fonte: Os autores (2012).

Após o levantamento de requisitos para o Motocão, foram pesquisadas as necessidades dos demais tipos de eventos que a aplicação engloba e elaborados os diagramas necessários. Então, iniciou-se o desenvolvimento da aplicação em questão.

3.2 PARTENON

O Partenon foi desenvolvido através da utilização dos recursos e arquitetura oferecidos pelo Demoiselle Framework 2.0 permitindo que com esta tecnologia seja garantida maior padronização, agilidade e qualidade no desenvolvimento, tornando-o de fácil entendimento e manutenção.

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 manutenções futuras. O Listagem 3 apresenta a estrutura de organização do projeto na IDE Netbeans juntamente com trechos de código fonte exemplificando o estrutura MVC da aplicação.



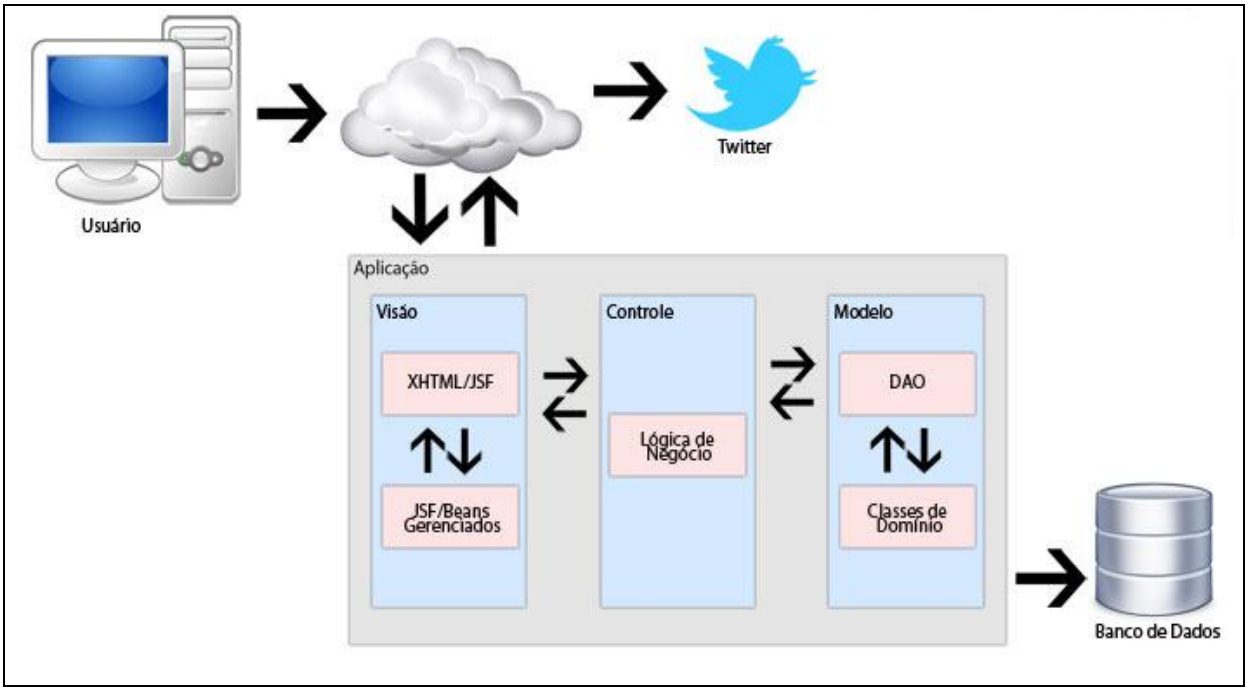
Listagem 3: Estrutura da aplicação Netbeans (MVC)
Fonte: Os autores (2012).

A listagem acima (Listagem 3) demonstra as partes que compõe a arquitetura MVC do sistema, iniciando pelo *Model* que é composto por entidades JPA e podem ser identificadas com a anotação *@Entity* juntamente com suas configurações. Também, existe o DAO que é um objeto utilizado para acesso a dados sendo ele a ponte entre as entidades e o *Controller* identificado com a anotação *@PersistenceController*.

Já o *Controller* é onde estão localizadas as regras de negócio da aplicação atuando como uma ponte entre a *View* e o *Model*. Pode ser identificado com a anotação *@BusinessController*.

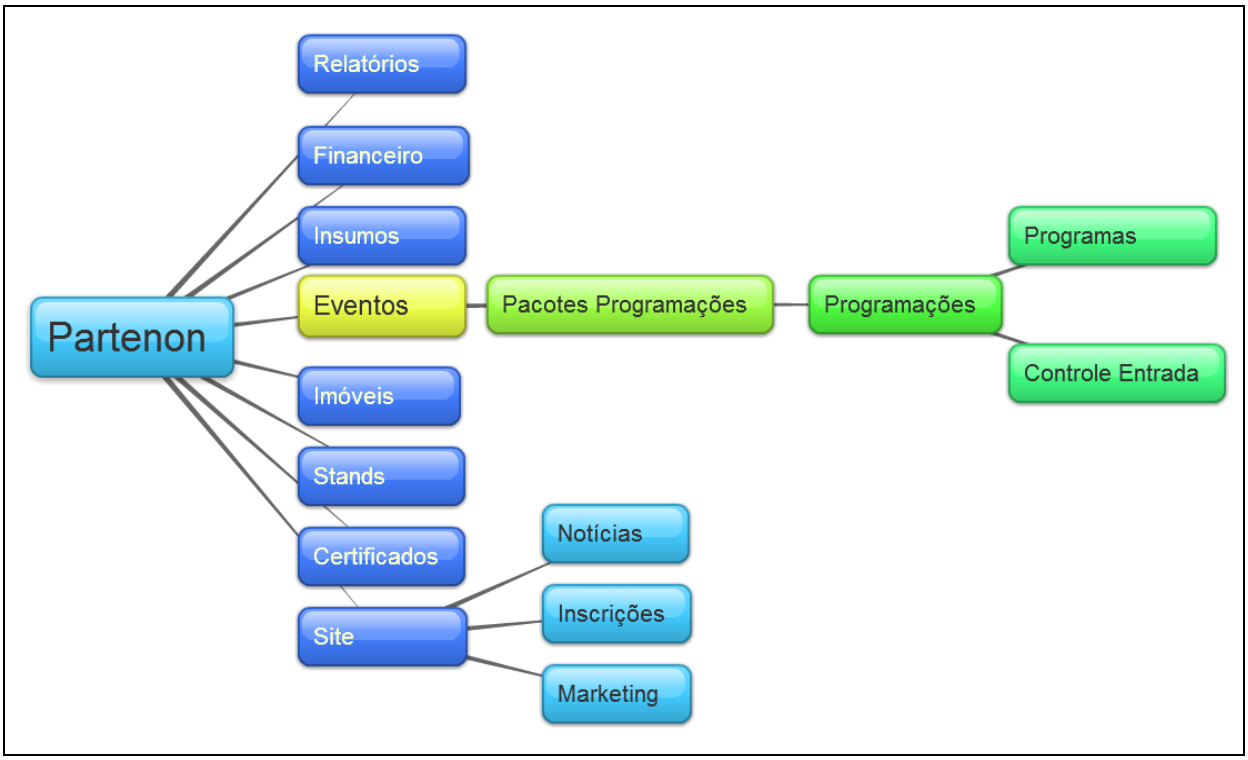
Por fim, pode-se citar a *View* que atua como uma camada que recebe as interações do usuário e repassa ao *Controller*, da mesma forma, exibe o que foi atualizado. É composta por dois componentes sendo eles uma página com código HTML e uma classe com código *Java*, que pode ser identificada pela anotação *@ViewController*.

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 citados acima.



Esquema 3: Arquitetura da Aplicação
Fonte: Os autores (2012).

O sistema é dividido em módulos que contém características semanticamente parecidas. Este estilo de organização facilita o entendimento do usuário sobre o sistema. Os principais módulos da aplicação são: Evento, Financeiro, Stands, Imóveis e Site. Conforme Esquema 4.



Esquema 4: Módulos da Aplicação
Fonte: Os autores (2012).

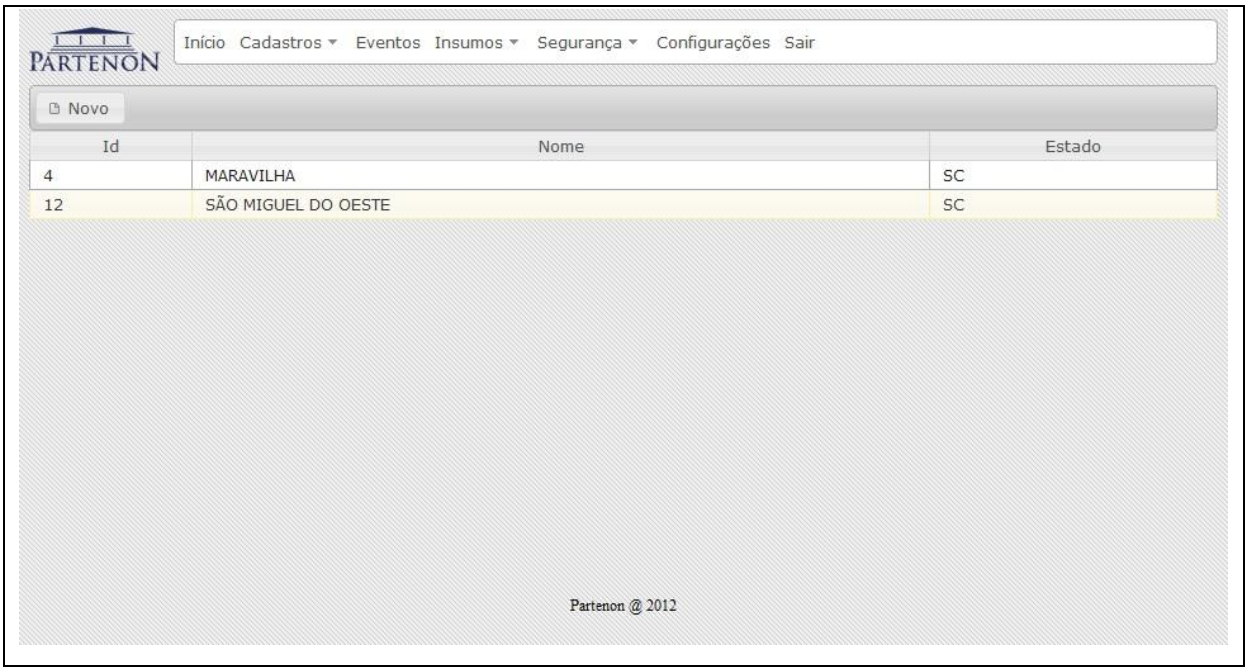
Para acessar o sistema Partenon é necessário estar logado por meio da tela de acesso ao sistema conforme Tela 1. O projeto também possui uma área de Segurança, onde além do seu *login* e senha cada usuário pertence a um Grupo de Usuários e o determinado Grupo possui os Recursos de Sistema.

Tela 1: Tela de acesso ao sistema Partenon
Fonte: Os autores (2012).

Na Tela 2 é apresentado a visualização do usuário após efetuar o *login* onde o "Menu" central é disponibilizado ícones grandes com atalho para algumas funcionalidades do sistema. O barra do "Menu" superior sempre fica visível ao usuário onde as demais paginas são carregadas dinamicamente na parte central. Assim, tornando o sistema mas fácil caso o usuário necessite acessar outras tela para demais funcionalidades.

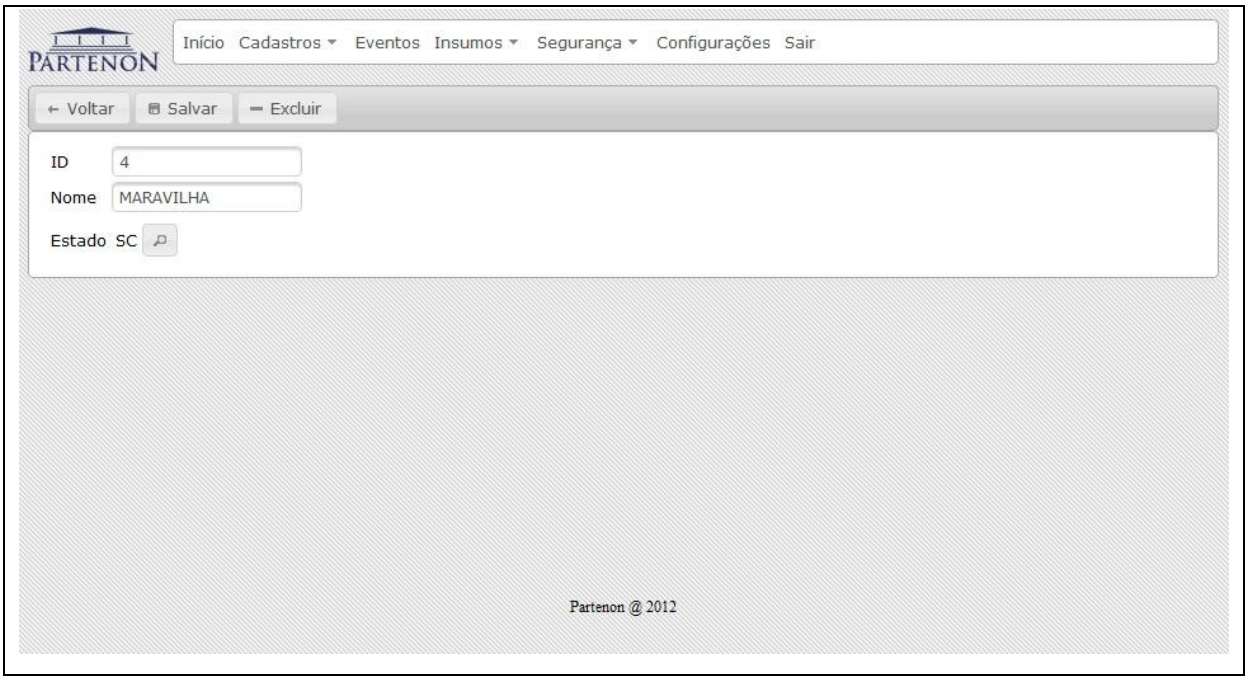
Tela 2: Tela principal do sistema Partenon
Fonte: Os autores (2012).

Exceto as telas que possuem dependência a um Evento especifico, as demais telas do sistema obedecem ao mesmo padrão para lançamento de informações conforme Tela 3. Ao acessar o módulo, primeira mente e feio uma listagem dos dados do determinado módulo que além de permitir a inclusão com um novo registro no sub-menu superior também permite selecionar um registro para efetuar demais operações no mesmo.



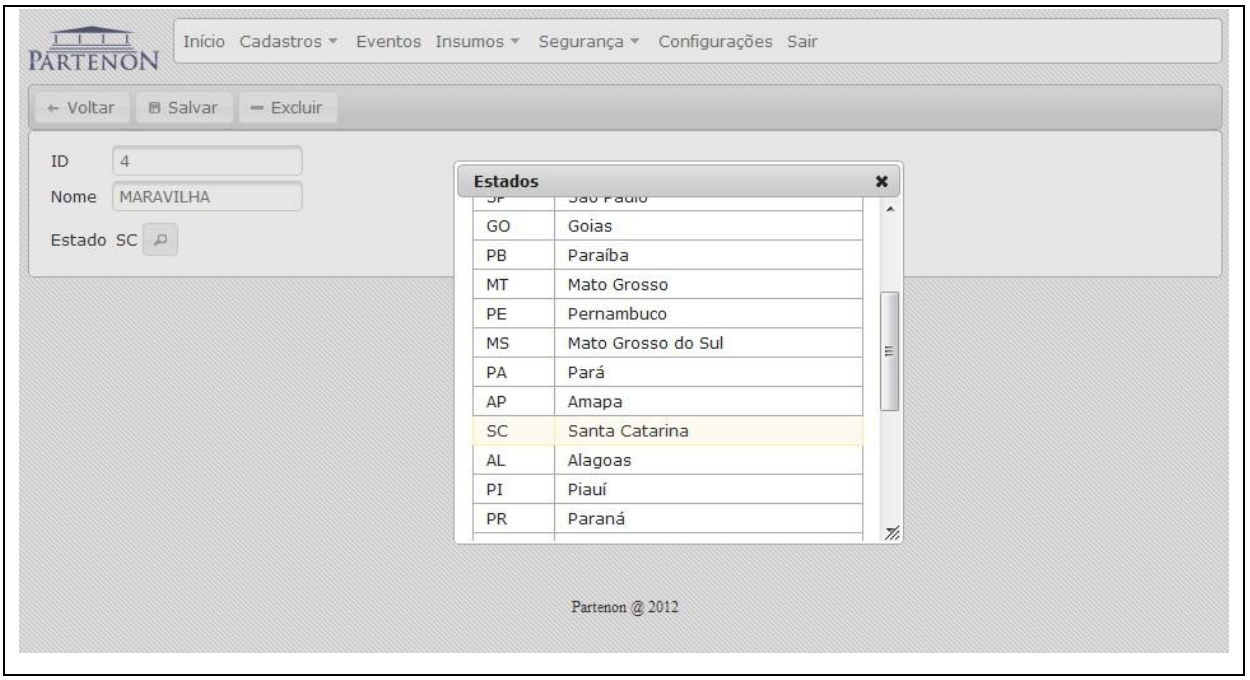
Tela 3: Tela padrão para listagem de registros do sistema Partenon
Fonte: Os autores (2012).

Ao selecionar um registro na tela de listagem ou mesmo clicando um “Novo”, o sistema redireciona automaticamente para uma nova tela, onde nela constam os demais campos do cadastro conforme Tela 4. No sub-menu superior conforme a operação, ficam disponível os demais botões como, “Salvar” garantido que seja salvo as informações, “Excluir” permitindo a exclusão do registro e “Voltar” para retomar a tela anterior (Desenho 10) .



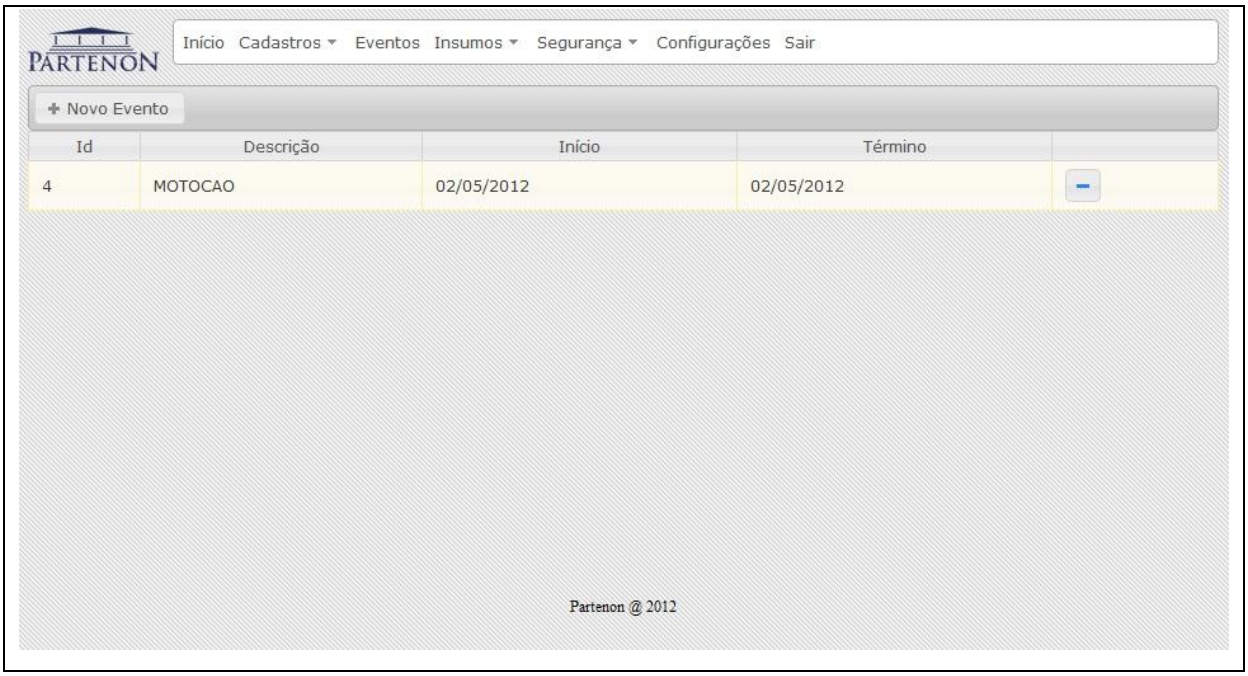
Tela 4: Tela padrão para cadastros do sistema Partenon
Fonte: Os autores (2012).

Quando é solicitada informação valida e previamente cadastrada, como o caso de Município em questão (Tela 4) necessitada de um Estado. Então, é disponibilizado um busca nas informações necessárias e persistir a informação no respectivo campo. Assim ao solicitar a pesquisa, clicando no botão quem possui a “Lupa”, abre-se então uma nova tela de pesquisa padrão do sistema, conforme Tela 5, para realizar a seleção da informação desejada.



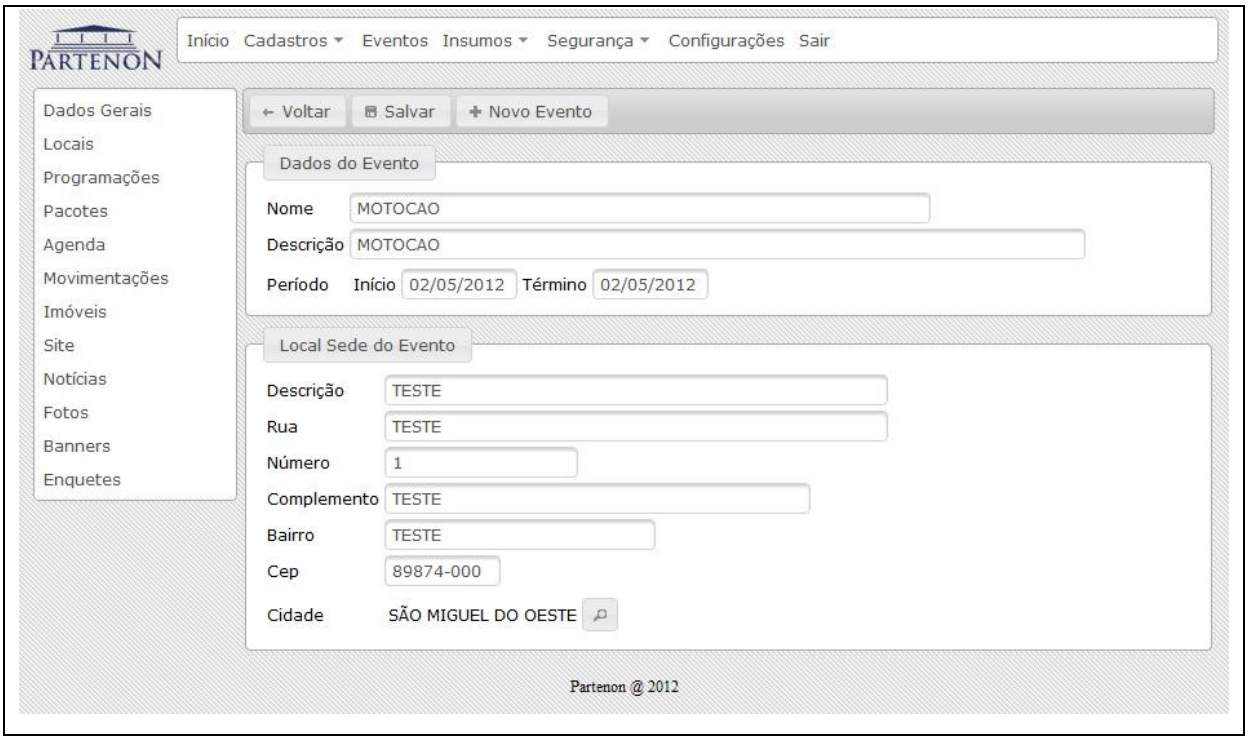
Tela 5: Tela padrão para consulta de informações do sistema Partenon
Fonte: Os autores (2012).

O módulo de Eventos é aonde detém maior dependência com demais módulos, pois cada Evento possui suas movimentações específicas e também, permite ser trabalhado com mais Eventos no próprio sistema. Assim, para acessar estes módulos específicos necessita-se selecionar em qual evento será trabalhado, conforme Tela 6.



Tela 6: Tela para seleção de Evento do sistema Partenon
Fonte: Os autores (2012).

Após selecionar o Evento desejado (Desenho 13), o sistema direciona para uma nova página sendo ela o módulo de Eventos onde engloba o cadastro das entidades relacionadas diretamente ao evento. Um evento é composto por várias programações, que podem ser realizadas em lugares distintos que foram previamente cadastrados no sistema. Uma programação pode estar em vários pacotes. Um pacote é um conjunto de programações tendo como finalidade facilitar as inscrições agrupando programações com características em comum, conforme Tela 7.

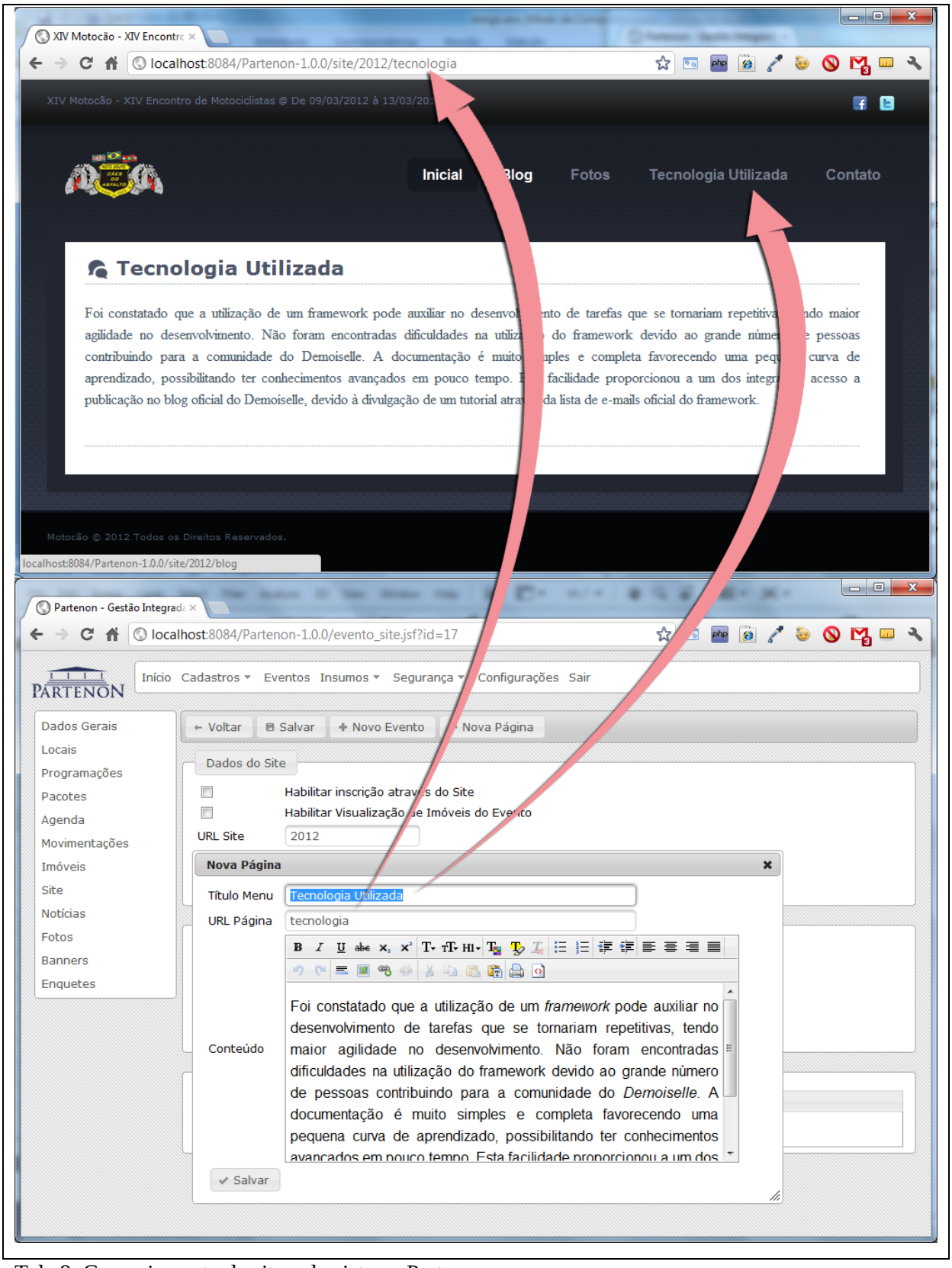


Tela 7: Tela para cadastro de Evento do sistema Partenon
Fonte: Os autores (2012).

Dentro dos módulos específicos do Evento, para a divulgação do meso foi desenvolvido um site que pode ser alimentado direto da aplicação, cadastrando páginas, notícias, galeria de fotos, banners e enquetes. Segue Tela 8, mostrando pagina gerado pelo sistema.

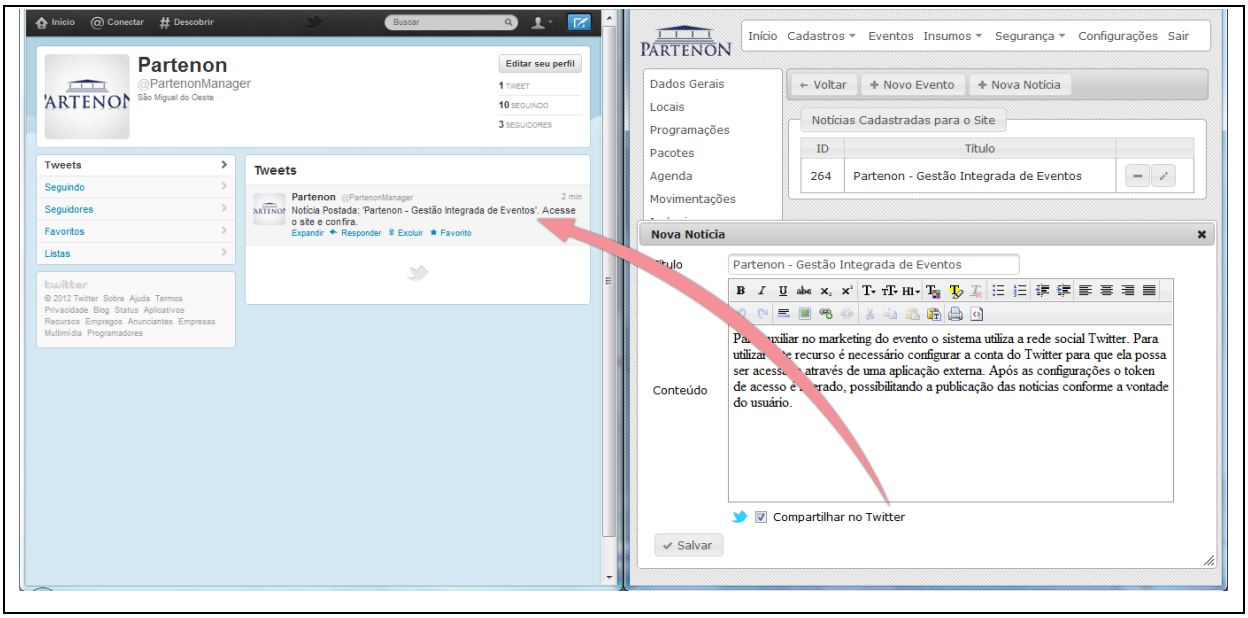
Tela 8: Site do Evento gerado e gerenciado pelo sistema Partenon
Fonte: Os autores (2012).

O gerenciamento e manutenção do site podem ser efetuados através do próprio usuário da aplicação, pois após salvar as informações as mesmas já ficam disponíveis no site. Também, as URLs que compões o site são amigáveis, facilitando a navegação de mecanismos de buscas bem como do usuário final, como pode ser observado no Tela 9.



Tela 9: Gerenciamento do site pelo sistema Partenon
Fonte: Os autores (2012).

Outra funcionalidade no gerenciamento do site são as notícias que além de estarem disponíveis no mesmo podem também ser propagadas pela rede social *Twitter* auxiliando no marketing do próprio Evento. Para utilizar este recurso é necessário configurar a conta do *Twitter* para que ela possa ser acessada através de uma aplicação externa. Após as configurações o *token* de acesso é liberado, possibilitando a publicação das notícias conforme a vontade do usuário (Tela 10).



Tela 10: Compartilhamento de notícias através do *Twitter* pelo sistema Partenon
Fonte: Os autores (2012).

Com os recursos mostrados acima o usuário tem a possibilidade de controlar o seu evento de forma eficaz, em ambiente *web* e assim realizar a divulgação do mesmo através de mecanismos avançados que são aceitos e utilizados globalmente.

3.3 RESULTADOS

A execução do presente projeto através do uso da arquitetura proposta pelo *framework,* proporcionou ao Partenon um ambiente de fácil manutenção e reutilização de códigos, diminuindo assim a probabilidade de erros e possíveis impactos com mudança ou implementação de novos requisitos.

Para demonstrar que os objetivos foram atingidos, foi elaborado um questionário respondido por usuários reais que fizeram uso da aplicação durante certo período. Esta avaliação serviu como *feedback* sobre melhorias no sistema, além de haver um retorno positivo onde foram sugeridas funcionalidades para implementações futuras.

No questionário (Apêndice D) foram abordados pontos relativos à funcionalidade, confiabilidade, usabilidade, eficiência e portabilidade sendo pontuados de 1 (Pouco) a 5 (Bastante). Para facilitar a leitura e compreensão dos resultados foi elaborado um gráfico (Gráfico 2) onde estão apresentadas as questões mais relevantes ao trabalho.

No quesito funcionalidade conforme Gráfico 2, está ilustrada a contabilização das respostas da questão 1 (Apêndice D) avaliando os requisitos propostos, se mostrando 80% satisfatório.

Quanto a confiabilidade, foi demonstrada (Gráfico 2) a questão 4 (Apêndice D) referente a falhas do sistema, mostrando que existem pontos que precisam ser melhor avaliados mas que não impedem a realização dos objetivos.

Outro ponto avaliado pelos usuários foi a usabilidade, para demonstrar a aceitação dos usuários (Gráfico 2) foi utilizada a questão 5 (Apêndice D) quanto a facilidade de entender ao que a aplicação se propõe, sendo 60% satisfatório.

A questão 8 (Apêndice D) foi escolhida para demonstrar (Gráfico 2) os aspectos da eficiência do sistema, se mostrando 80% satisfatório neste quesito.

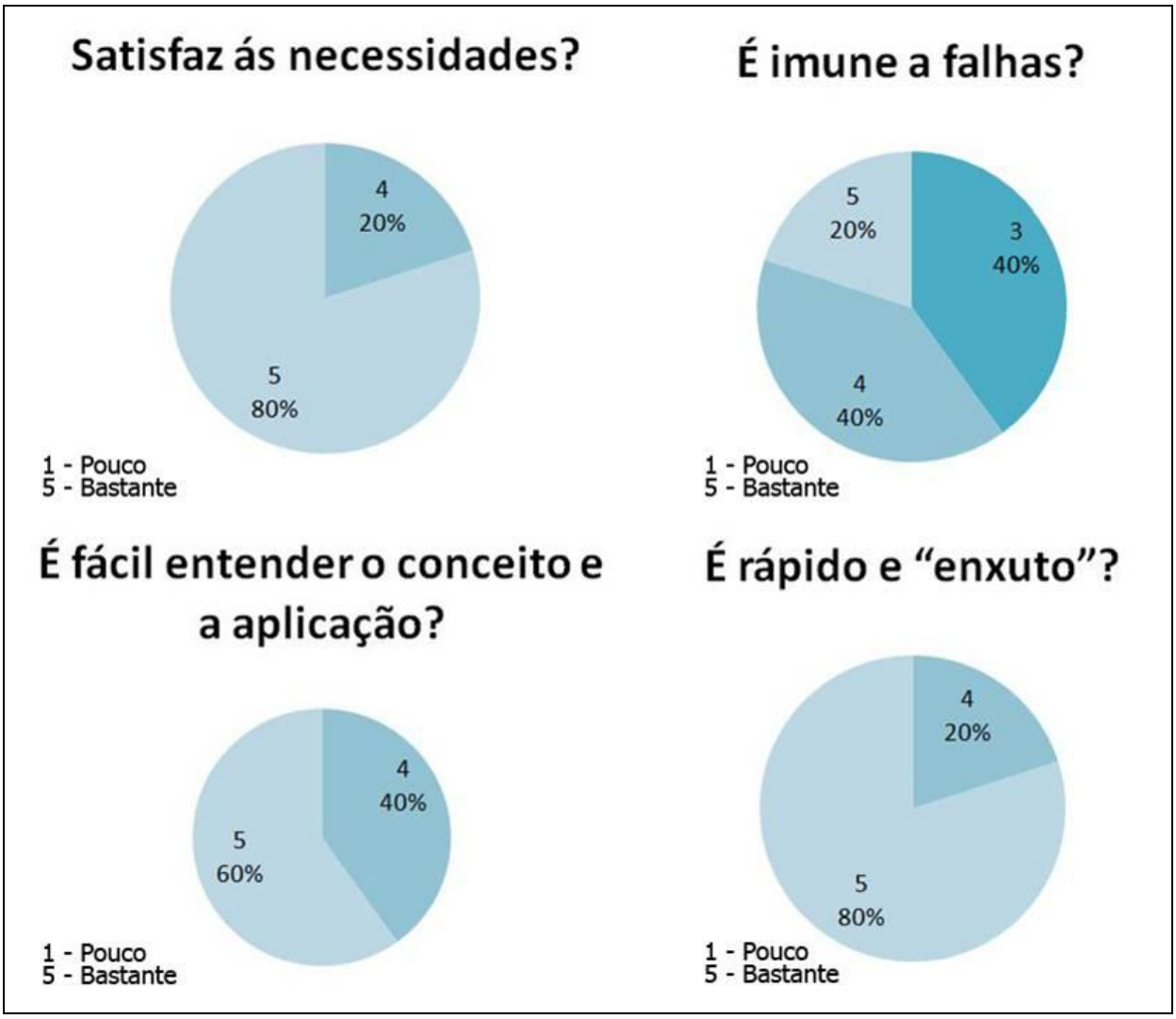


Gráfico 2: Gráficos das respostas do questionário
Fonte: Os autores (2012).

Com base nos dados visualizados pode-se identificar que os objetivos pretendidos foram alcançados, porém alguns aspectos precisam ser melhor avaliados.

Além dos resultados obtidos com a aplicação também foi constatado que a utilização de um *framework* pode auxiliar no desenvolvimento de tarefas que se tornariam repetitivas, tendo maior agilidade no desenvolvimento.

Assim, não foram encontradas dificuldades na utilização do *framework* devido ao 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.

Esta facilidade proporcionou a um dos integrantes acesso a publicação no blog oficial do *Demoiselle*, devido à divulgação de um tutorial através da lista de e-mails oficial do *framework* (Apêndice A).

Outro resultado importante foi o convite para explanar nossos conhecimentos com o *framework Demoiselle* e apresentar o presente projeto como resultado do mesmo no I BootCamp OeSC-Livre em Xanxerê – SC, realizado no dia 10 de março de 2012 (Apêndice B).

4 CONCLUSÃO

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

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

A escolha de tecnologias foi fator determinante para o desenvolvimento eficaz, pois as mesmas garantiram a separação, padronização, centralização de funções e operações ganhando tempo com desenvolvimento e facilitando a manutenção do sistema.

Um ponto fundamental para o sucesso do trabalho em equipe foi o gerenciamento de projetos, onde todo o processo de desenvolvimento foi planejado e dividido em tarefas, e cada componente do grupo realizou suas tarefas podendo adicionar anotações, tendo um maior controle do que foi feito e a quantidade de tempo para realizar tarefa.

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

4.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Em trabalhos futuros, devem ser acompanhadas as constantes atualizações do Framework juntamente com os demais acoplados ao mesmo, pois estão ocorrendo mudanças significativas no que diz respeito à implementação de padrões das especificações *Java*.

Notando certa tendência das aplicações migrarem para a nuvem (*cloud computing*) e a capacidade do *framework* neste aspecto, seria possível disponibilizar o software com um serviço (*SaaS*) tendo apenas uma aplicação em execução para vários usuários.

Um recurso significativo é a criação de uma versão para dispositivos móveis do site gerado pela aplicação, além de um aplicativo para controle de entrada no evento, não necessitando ter uma estação fixa neste local. Desta forma, aumentando a mobilidade e acompanhando o crescente aumento de utilização deste tipo de dispositivo.

REFERÊNCIAS

ALCARÁ, Adriana Rosecler; CHIARA, Ivone Guerreiro di; TOMAÉL, Maria Inês. **Das redes sociais à inovação.** Brasilia; Ci. Inf. 2005. Disponivel em: <http://www.scielo.br/pdf/ci/v34n2/28559.pdf> Acesso em: 24 maio 2011.

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

BECK, Kent et al. **Principles behind the Agile Manifesto**. Disponível em: <http://agilemanifesto.org/principles.html>. Acesso em: 23 maio 2011.

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

BURNS, Ed; SCHALK, Chris. **JavaServer Faces:** The Complete Reference. Estados Unidos da América: Mcgraw-hill, 2010. 722 p.

CENTRO DE DIFUSÃO DE TECNOLOGIA E CONHECIMENTO.**DotProject.** Disponível em: <http://www.cti.ufu.br/sites/cti.ufu.br/files/dotproject-user-manual-cdtc.pdf>. Acesso em: 30 abr. 2012.

CESCA, Cleuza G. Gimenes. **Organização de Eventos:** manual para planejamento e execução. São Paulo: Summus, 1997.

COLLINS-SUSSMAN, Ben; FITZPATRICK, Brian W.; PILATO, C. Michael. **Version Control with Subversion:** For Subversion 1.4 (Compiled from r2866). [s. L.]: Tba, [20--]. 370 p. Disponível em: <http://svnbook.red-bean.com/en/1.4/svn-book.pdf>. Acesso em: 30 maio 2011.

COSTA, Williams de Oliveira; GUEDES, Marcos. **GERENCIAMENTO DE PROJETOS COM DOTPROJECT:** UM CASO PRÁTICO. Disponível em: <http://www.dotproject.com.br/documentos?download=20%3Atcc-william>. Acesso em: 30 abr. 2012.

CONSTANTINO FILHO, Ademir. **Minha primeira aplicação com JSF + GlassFish**. Disponível em: <http://www.guj.com.br/articles/192>. Acesso em: 19 maio 2011.

CRAWFORD, William; KAPLAN, Jonathan. **J2EE™ Design Patterns**. Estados Unidos da América: O'Reilly, 2003. 350 p.

DALL'OGLIO, Pablo. **PHP:** Programando com Orientação a Objetos. 2. ed. São Paulo: Novated Editora, 2009. 574 p.

DEMOISELLE Framework. **Arquitetura de Referência para Aplicações Demoiselle**. 13 abr. 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.

FLANAGAN, David. **Java:** O Guia Essencial. 5. ed. São Paulo: O'reilly, 2006. 1099 p.

FURGERI, Sérgio. **Java 7:** Ensino Didático. São Paulo: Érica, 2010. 319 p.

GAMMA, Erich et al. **Design Patterns:** Elements of Reusable Object-Oriented Software.[S.I]: Addision-Wesley Publishing Co., 1998. 381 p.

GEARY, David; HORSTMANN, Cay. **Core JavaServer Faces**. Rio de Janeiro: Alta Books, 2005. 355 p.

GIÁCOMO, Cristina**. Tudo acaba em festa.** São Paulo: Página Aberta, 1993.

GOOGLE CODE. **Como usar a JPA com o Google App Engine**. Disponível em: <http://code.google.com/intl/pt-BR/appengine/docs/java/datastore/usingjpa.html>. Acesso em: 28 jun. 2011.

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

GONÇALVES, Edson. **Desenvolvendo aplicações web com NetBeans IDE 6**. Rio de Janeiro: Editora Ciência Moderna, 2008.

INSTITUTO Evaldo Lodi, Redes Sociais: Web 2.0 amplia possibilidades de negócios e revoluciona relações com o mercado. IEL Interação Revista, Brasília, p 12, Set 2009.

KEITH, Mike; SCHNICARIOL, Merrick. **Pro JPA 2**: Mastering the Java™ Persistence API. Estados Unidos da América: Apress, 2009. 481 p.

KOSCIANSKI, André; SOARES, Michel Dos Santos. **Qualidade de Software**: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. São Paulo: Novatec Editora, 2006. 395 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.

MATIAS, Marlene. **Organização de Eventos:** Procedimentos e técnicas. 3. ed. Barueru: Manole, 2004. 157 p.

MEIRELLES, Gilda Fleury. **Eventos:** seu negócio, seu sucesso.[S.I][s.n][200-].

MELO NETO, Francisco Paulo de. **Marketing de Eventos.** 4.ed. Rio de Janeiro:Spring, 2003. 236 p.

PAULA NETO, Aristides Vicente de. **Criando testes com JUnit.** Disponível em: <http://javafree.uol.com.br/dependencias/tutoriais/testes_junit.pdf>. Acesso em: 30 maio 2011.

PITANGA, Talita. **JavaServer Faces**. Disponível em: <http://www.guj.com.br/articles/158>. Acesso em: 19 maio 2011.

PRESSMAN, Roger S.. **Engenharia de software.** 6. ed. São Paulo: Mcgraw-hill, 2006. 719 p.

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.

STERENE, Jim. **Marketing na Web:** Integrando a Web à sua estratégia de marketing. Rio de Janeiro: Campus, 2000. 398 p.

SONATYPE. **Maven:** The Definitive Guide. [s. L.]: O'reilly, 2008. 455 p. Disponível em: <http://www.sonatype.com/books/mvnref-book/reference/>. Acesso em: 30 maio 2011.

VALENTE, Fernando. **[Rapidinhas] – O que é MVC?** 11 Jan. 2011. Disponível em: <http://www.fernandovalente.com.br/wordpress/2011/01/11/mvc-model-view-controller/>. Acesso em: 30 mai. 2011.

WEAVER, James L.; MUKHAR, Kevin; CRUME, Jim. **Beginning J2EE 1.4**: From novice to professional. Estados Unidos da América: Apress, 2004. 600 p.

WELLS, Don. **Extreme Programming**: A gentle introduction. Set 2009. Disponível em <http://www.extremeprogramming.org/index.html>. Acesso em: 29 maio 2011.

APÊNDICES

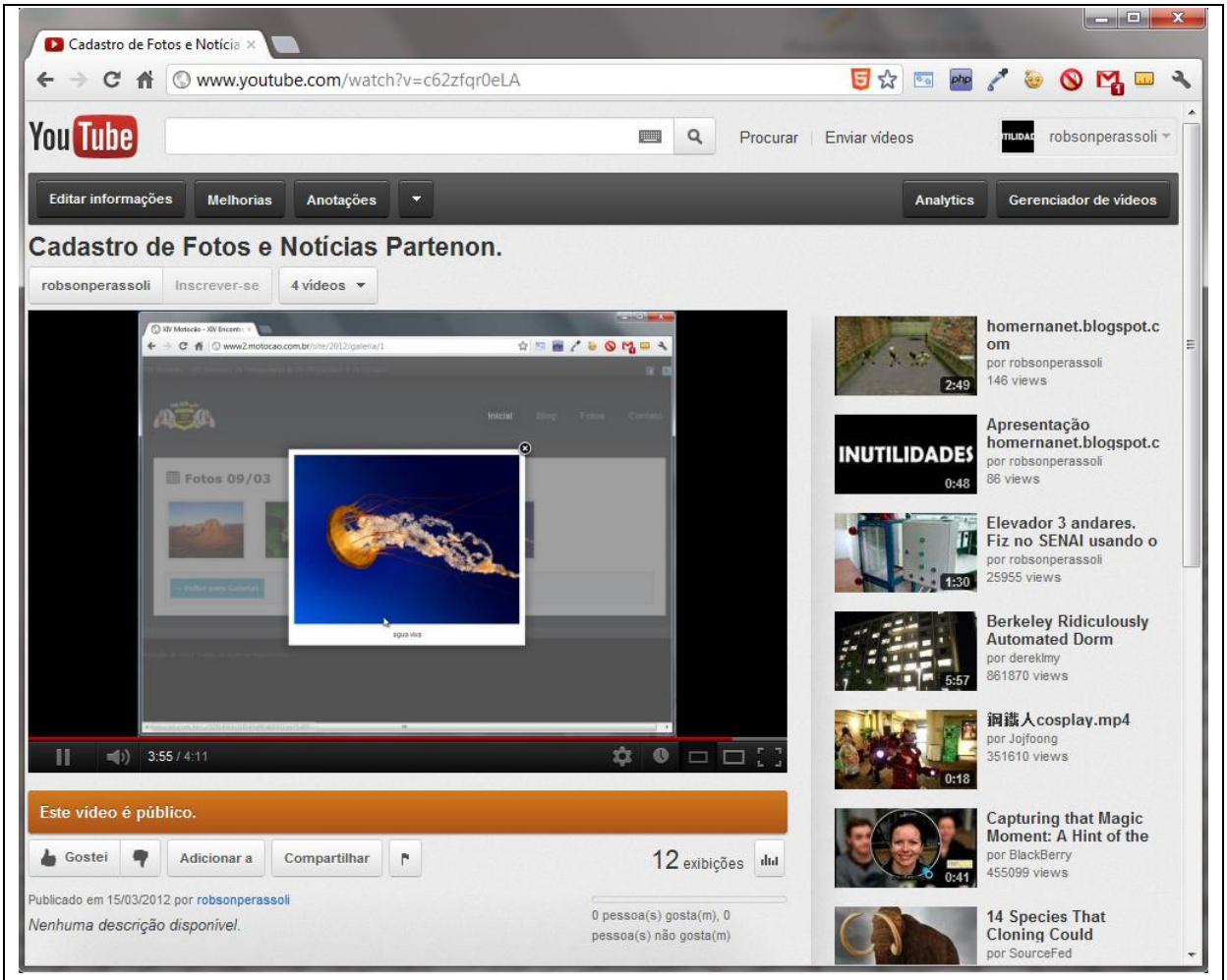
APÊNDICE A – Publicação no Blog Oficial do *Demoiselle Framework*

Publicação no Blog Oficial do *Framework.*

APÊNDICE B – Palestra *Demoiselle*/Partenon no I BootCamp OeSC-Livre em Xanxerê-SC

Luiz Felipe Bartz, Robson Gian Peressoli e Professor Roberson J. F. Alves (Orientador) no momento da apresentação.

APÊNDICE C – Tutorial para cadastrar fotos e notícias



Vídeo elaborado para auxiliar usuário a cadastrar fotos e notícias no gerenciamento de site através do sistema Partenon.
http://www.youtube.com/watch?v=c62zfqr0eLA

APÊNDICE D – Questionário aplicado para avaliação do Software

PARTENON	**Partenon** *Gestão Integrada de Eventos*

Questionário para avalição do sistema

Principais Requisitos:

- Criação de portal onde será possível visualizar notícias e páginas que serão cadastradas no sistema e atualizadas dinamicamente.
- No portal possibilitar aos participantes a visualização de imóveis para hospedagem, estes imóveis devem ser cadastrados no sistema.
- Criação de módulo para controle das movimentações financeiras.
- Módulo para controle da locação de stands

Nome: _____

Marque com um "x" as repostas de acordo com os requisitos descritos acima e a experiência que você obteve ao utilizar o sistema.

Funcionalidade

1-Satisfaz ás necessidades?
Pouco ( ) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 Bastante

2-Propõe-se a fazer o que é apropriado?
Pouco ( ) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 Bastante

3-Faz o que foi proposto de forma correta?
Pouco ( ) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 Bastante

Confiabilidade

4-É imune a falhas?
Pouco ( ) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 Bastante

Usabilidade

5-É fácil entender o conceito e a aplicação?
Pouco ( ) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 Bastante

6-É fácil aprender a usar?
Pouco ( ) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 Bastante

7-É fácil de operar e controlar?
Pouco ( ) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 Bastante

Eficiência

8-É rápido e "enxuto"?

Pouco ( ) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 Bastante

9-O tempo de resposta é satisfatório?

Pouco ( ) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 Bastante

Portabilidade

10-É fácil de usar em outro ambiente?

Pouco ( ) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 Bastante

Geral

11-Pontos positivos?

_____
_____
_____
_____
_____

12-Pontos negativos?

_____
_____
_____
_____
_____

Assinatura: _____