**Lauro Silveira Neto**

# Uso de ferramentas livres para implantação de práticas e princípios ágeis em um projeto de desenvolvimento de software

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. Msc. Joaquim Quinteiro Uchôa

Lavras
Minas Gerais - Brasil
2009

**Lauro Silveira Neto**

**Uso de ferramentas livres para implantação de práticas e princípios ágeis em um projeto de desenvolvimento de software**

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *Abril de 2009*

Prof. Arlindo Follador Neto

Prof. Samuel Pereira Dias

Prof. Msc. Joaquim Quinteiro Uchôa
(Orientador)

Lavras
Minas Gerais - Brasil
2009

*Dedico esta monografia a minha família, em especial a minha esposa pela paciência e apoio em todos os momentos*

# Agradecimentos

Agradeço a todos que de alguma forma contribuiram para que eu realizasse este trabalho, em especial o Prof. Joaquim, por ter se prontificado em assumir a orientação em um momento delicado.Agradeço também os professores Arlindo e Samuel, por aceitarem participar da minha banca com tão pouco tempo para lerem e avaliarem o trabalho.

# Sumário

# Lista de Figuras

# Lista de Abreviaturas

| | |
|---|---|
| AJAX | *Asynchronous Javascript And XML.* |
| API | *Application Programming Interface*. |
| CASE | *Computer-Aided Software Engineering*. |
| CRUD | *Create, Retrieve, Update and Delete*. |
| CSS | *Cascade Style Sheet*. |
| DSDM | *Dynamic System Development Method.* |
| HTML | *HyperText Markup Language*. |
| JEE | *Java Enterprise Edition*. |
| LDAP | *Lightweight Directory Access Protocol*. |
| IDE | *Integrated Development Environment*. |
| MVC | *Model View Controller*. |
| UML | *Unified Modeling Language*. |
| XML | (*Extensible Markup Language*). |
| XP | Programação Extrema (*Extreme Programming*). |

# Resumo

O desenvolvimento de sistemas têm evoluído bastante nos últimos anos, e dentre as tecnologias que mais se destacaram na área, Java pode ser considerada uma das mais bem sucedidas. Na área de metodologias de desenvolvimento, as chamadas metologias ágeis têm obtido grande espaço em detrimento às chamadas metodologias tradicionais. Buscando adequar as tecnologias às metodologias ágeis, inúmeros *frameworks* e ferramentas de desenvolvimento surgiram para que se possa obter o máximo de produtividade sem perder a qualidade exigida. Este trabalho têm como objetivo apresentar a reengenharia de um sistema através da utilização de práticas e ferramentas ágeis (livres e voltadas para a tecnologia Java), analisando os benefícios que a adoção das mesmas podem trazer para o projeto de desenvolvimento de um *software*.

**Palavras-Chave:** Metodologias Ágeis; Desenvolvimento de *Software*; Java; *Groovy*; *Grails*; *Extreme Programming*.

# Capítulo 1

# Introdução

Com todo o avanço na área de tecnologia da informação ocorrido nos últimos anos, a demanda pela informatização de procedimentos e processos dentro das empresas cresceu exponencialmente. A informatização se deu através do desenvolvimento de *softwares*, que automatizaram os processos que até então, na sua maioria, eram feitos de forma manual. No entanto, o desenvolvimento de *sofware* acabou se tornando cada vez mais complexo. Assim, com o intuito de auxiliar o desenvolvimento, surgiram as metodologias ou processos de desenvolvimento de *software*, que são um conjunto de atividades necessárias para produzir um *software*. Estas atividades por sua vez, são divididas em fases, que definem o que será feito, quem irá fazer e como irá fazer.

Com a evolução das linguagens de programação e as necessidades por parte dos clientes cada vez mais urgentes, muitas das metodologias se mostraram ineficientes, seja pela documentação excessiva, seja pela falta de comunicação com os usuários durante o processo de desenvolvimento, o que leva a construção de um *software* que não executa o que o cliente deseja.

## 1.1 Motivação

Em 2001 um grupo de especialistas em processos de desenvolvimento de *software*, que pregavam o uso metodologias mais simplificadas do que as tradicionais, se reuniram estabeleceram um conjunto de princípios comuns compartilhados entre as metodologias utilizadas por eles.

Nesse encontro foi estabelecido o *Manifesto Ágil*[1], com os seguintes princípios:

- Indivíduos e interações ao invés de processos e ferramentas;
- *Software* executável ao invés de documentação;
- Colaboração do cliente ao invés de negociação de contratos;
- Respostas rápidas a mudanças ao invés de seguir planos.

A partir do *Manifesto Ágil*, metodologias ágeis como o *XP*[2] e o *Scrum*[3] começaram a se difundir mais, mostrando-se mais eficientes do que as chamadas metodologias tradicionais. Hoje, inúmeras empresas estão migrando os seus processos de desenvolvimento para metodologias ágeis, pelo fato destas trazerem mais eficiência, menos burocracia e um maior grau de satisfação para os clientes, principalmente pelo fato das mesmas estarem focadas nas pessoas e não nos processos.

## 1.2 Objetivo

O objetivo deste trabalho é apresentar algumas das metodologias ágeis e estabelecer o ganho obtido através da implementação de práticas sugeridas pelas metodologias ágeis e a reengenharia de um sistema em um ambiente que até então utilizava uma metodologia tradicional.

Além disso, este trabalho também pretende mostrar que é perfeitamente viável implantar práticas sugeridas pelas metodologia ágeis utilizando apenas *software* livre, tanto nas ferramentas para o desenvolvimento do sistema (*Groovy*[4] e *Grails*[5]) como nas ferramentas de apoio ao processo.

## 1.3 Estrutura

Este trabalhando está dividido da seguinte maneira:

[1] http://agilemanifesto.org/
[2] http://www.extremeprogramming.org/
[3] http://www.scrumalliance.org/
[4] http://groovy.codehaus.org/
[5] http://grails.org/

O capítulo 2 têm como objetivo apresentar um pouco sobre as metodologias ágeis existentes no mercado, como o *Extreme Programming, Scrum, Lean*, etc, discorrendo um pouco sobre os seus príncipios e objetivos.

O Capítulo 3 traz um pouco sobre as ferramentas livres utilizadas no desenvolvimento (*Groovy* e *Grails*) e apoio ao desenvolvimento (*DotProject e JSPWiki*) dos sistema. Dentre as ferramentas de desenvolvimento estão o *Groovy*, que é a linguagem ágil para desenvolvimento de aplicações utilizando a tecnologia *Java*, e o *Grails*, que é um *framework* para desenvolvimento de aplicações *Web* que integra o *Groovy* a outras tecnologias já consolidadas no mercado. Já na parte de ferramentas de apoio ao desenvolvimento serão brevemente apresentados o *dotProject*[6] e o *JSPWiki*[7].

O capítulo 4 traz a reengenharia de um sistema adotando práticas e ferramentas livres ágeis, além de um comparativo de algumas métricas coletadas no sistema legado e no sistema refatorado.

Finalmente, o capítulo 5 contêm as considerações finais.

[6] http://www.dotproject.net
[7] http://jspwiki.org/

# Capítulo 2

# Metodologias Ágeis

Nos últimos anos, com o grande crescimento da indústria de desenvolvimento de *software*, surgiram inúmeras metodologias ágeis, sendo que todas possuem o mesmo objetivo: criar *software* confiável rapidamente (LEFFINGWELL, 2007).

Este capítulo traz um pouco do histórico das metodologias ágeis e apresenta algumas das metodologias existentes no mercado (Figura 2.1), dentre elas:

- *XP*;
- *Scrum*;
- *Lean*;
- *Dynamic System Development Method*;
- *Feature Driven Development*;

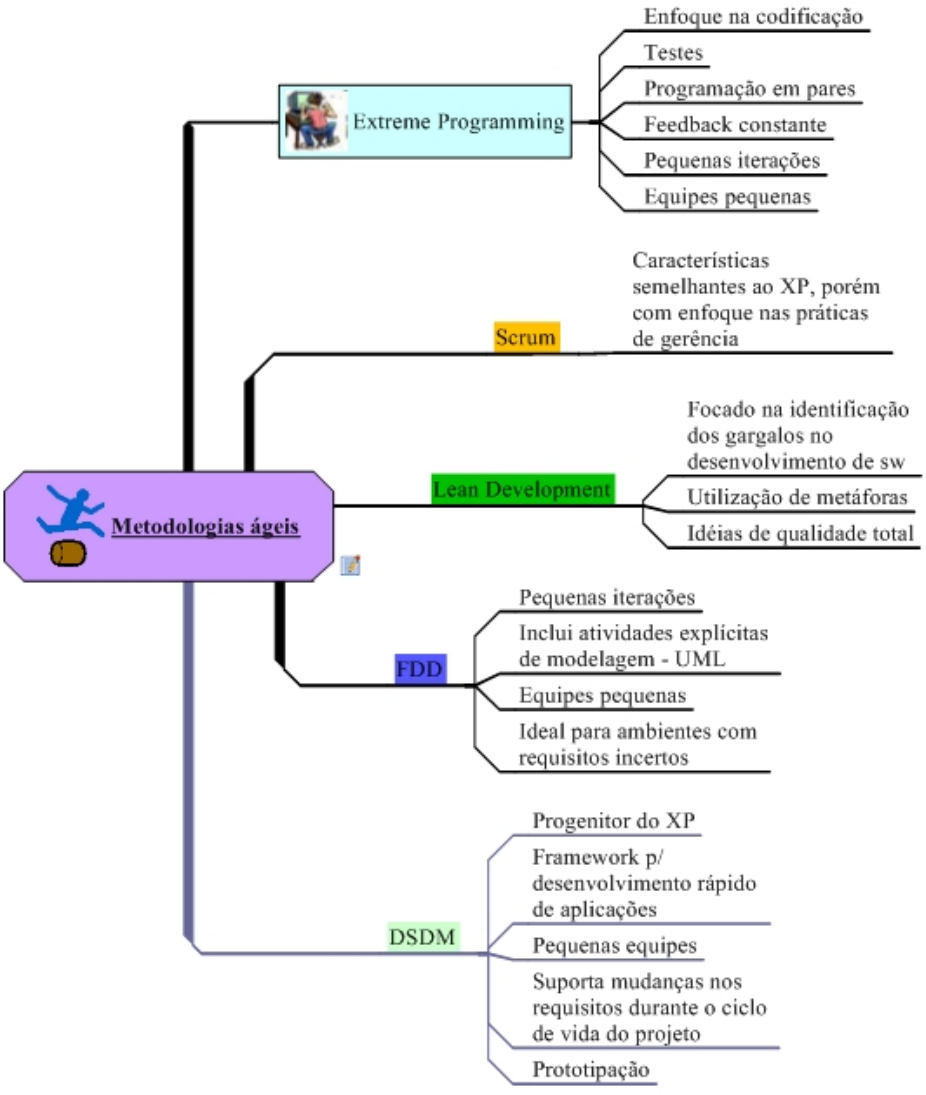


**Figura 2.1:** Metodologias ágeis

O capítulo terá maior ênfase no *XP* e no *Scrum*, pelo fato destas serem as metodologias ágeis mais utilizadas atualmente (LEFFINGWELL, 2007). As demais técnicas serão brevemente apresentadas e comparadas, levando em conta o seu propósito e a sua aplicabilidade.

## 2.1 Histórico

Em 2001, um grupo de interessados em metodologias ágeis e iterativas se reuniram para estabelecer um consenso e trocar experiências sobre as metodologias utilizadas , com o objetivo de propor alternativas ao desenvolvimento tradicional.

Assim, surgiu o Manifesto Ágil[1], que continha as seguintes premissas:

- Indivíduos e interações **são** mais importantes que processos e ferramentas;
- *Software* funcionando **é** mais importante do que documentação completa e detalhada;
- Colaboração com o cliente **é** mais importante do que a negociação de contratos;
- Adaptação a mudanças **é** mais importante do que seguir o plano inicial.

Em cima destas premissas chegou-se à conclusão que apesar dos itens à direita serem valorizados, os itens à esquerda seriam priorizados.

Além das quatro premissas, o manifesto ágil estabeleceu uma série de princípios que o desenvolvimento ágil deveria seguir (LARMAN, 2003):

1. A prioridade é satisfazer o cliente, entregando o mais rápido possível e de forma contínua *software* que tenha valor;
2. Requisitos mutantes são bem vindos, mesmo no final do desenvolvimento. Os processos ágeis podem ser usados a favor de mudanças que tragam vantagem competitiva para o cliente;
3. É importante entregar *software* funcionando freqüentemente, mensalmente, quinzenalmente ou, se possível, toda semana;

[1] http://agilemanifesto.org/

4. Clientes e desenvolvedores devem trabalhar juntos diariamente num projeto;
5. Projetos devem ser feitos por indivíduos motivados. Os indivíduos precisam da confiança de que seu trabalho será realizado. Eles devem ter suas necessidades atendidas e trabalhar num ambiente adequado;
6. Conversa pessoalmente é sempre a melhor forma de comunicação;
7. *Software* funcionando é a primeira medida de progresso;
8. O processo ágil torna o desenvolvimento sustentável. Patrocinadores, desenvolvedores e usuários devem manter a paz indefinidamente;
9. Atenção constante à excelência técnica e bom *design* aumenta a agilidade;
10. A chave é a simplicidade: a arte de minimizar a quantidade de trabalho desnecessário;
11. As melhores arquiteturas, requisitos e *design* surgem de equipes de desenvolvimento auto-organizadas;
12. Em intervalos regulares, a equipe de desenvolvimento reflete como se tornar mais eficiente. Então ajusta seu comportamento para atingir esse objetivo.

A partir do manifesto ágil, as metodologias ágeis ganharam força dentro da indústria do *software*. Algumas destas metodologias serão apresentadas a seguir.

## 2.2 Extreme Programming

O *Extreme Programming* , ou simplesmente *XP*, como é popularmente conhecido, é uma metodologia ágil para equipes médias e pequenas, que desenvolvem *software* com requisitos que são vagos ou mudam frequentemente (BECK, 2000). No *XP*, a principal tarefa é a programação, tendo uma ênfase menor em processos formais e maior em disciplina rigorosa. Além disso, baseia-se em atividades como:

- Revisão permanente do código;
- Testes frequentes;
- Participação do usuário final;

- Refatoramento contínuo;
- Refinação contínua da arquitetura;
- Integração contínua;
- Planejamento, *design* e *redesign* a qualquer hora.

O *XP* é composto de alguns valores e práticas, descritos nas seções a seguir.

### 2.2.1 Valores

O *XP* possui os seguintes valores:

1. **Comunicação:** a comunicação não é limitada por procedimentos formais, dando maior agilidade ao escolher a forma mais rápida. Assim, usa-se o melhor meio possível, que pode ser: uma conversa, uma reunião informal, um *email*, diagramas, o próprio código, etc.
2. **Simplicidade:** o *Extreme Programming* incentiva ao máximo práticas que reduzam a complexidade do sistema. A solução escolhida deve ser sempre a mais simples e que alcance os resultados esperados. Por exemplo, preferir sempre a tecnologia mais simples, bem como a implementação de algoritmo mais simples para atender os requisitos. Outro aspecto é simplificar qualquer fase (projeto, codificação, etc) sempre que possível e descartar tudo que seja relacionado às iterações seguintes.
3. **Feedback:** as práticas estabelecidas pelo *XP* permitem que o *feedback* seja dado de forma mais rápida. Isto possibilita obter de forma mais rápida informações como o estado do desenvolvimento, o número de erros detectados e corrigidos dentre outras informações. Com este *feedback*, fica mais fácil a tomada de decisões, através da obtenção de estimativas mais precisas, gerando maior segurança e menos riscos para o cliente. A Figura 2.2 representa a relação do *feedback* com o ciclo de vida do *software*. O cliente define o valor, ou seja, aquilo que ele deseja que seja feito (requisitos). O programador por sua vez realiza as estimativas de custo e tempo para implementar os requisitos definidos pelo usuário. O cliente então define quais os valores(requisitos) são prioritários e quais podem ser descartados. Uma vez definidas as prioridades do cliente, o programador finalmente constrói o que foi definido pelo cliente. Assim, através do *feedback* constante, o cliente aprende com o programador e vice-versa.

4. **Coragem:** ao realizar as práticas propostas pelo *XP*, o programador se sente mais confiante, o que lhe ajuda a ter mais coragem para: refatorar código que está funcionando para torná-lo mais simples; jogar fora código desnecessário; investir tempo no desenvolvimento de testes; pedir ajuda aos mais experientes; dizer a um cliente que um requisito não vai ser implementado no prazo estipulado; abandonar processos formais e fazer documentação na forma de código.

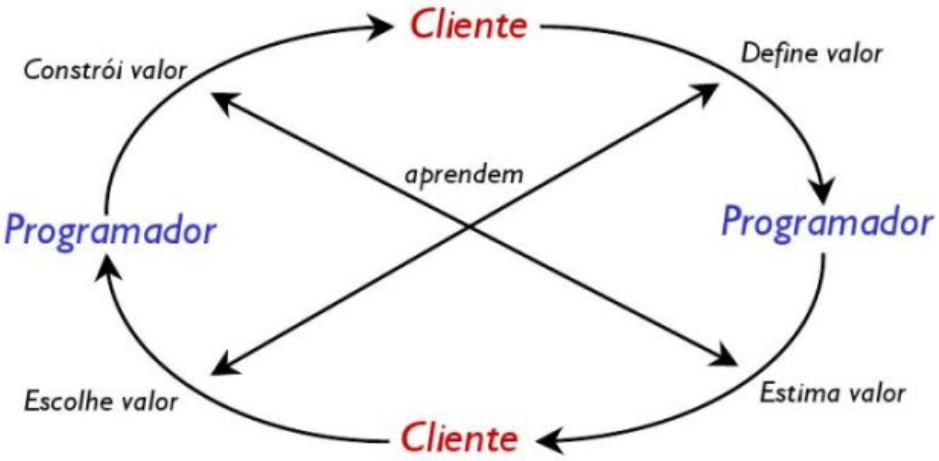


**Figura 2.2:** Feedback x Ciclo de vida - Fonte: (BECK, 2000)

### 2.2.2 Práticas

Conforme pode ser visto na Figura 2.3, o *XP* possui as seguintes práticas:


**Figura 2.3:** Práticas do XP - Fonte: (JEFFRIES, 2001)

1. **Equipe:** a equipe é formada por aqueles que irão construir o *software* (analistas, programadores, gerentes) e também pelo cliente, que participará ativamente estabelecendo os requisitos, definindo as prioridades e controlando o rumo do projeto.

2. **Jogo do Planejamento:** nesta prática o cliente define as funcionalidades desejadas para a iteração. Então os programadores avaliam a dificuldade de implementá-las. Com esta prática, cliente tem sempre o status do projeto e o que está sendo feito. Resumindo, nesta prática são definidas as estimativas de cada tarefa e quais são as tarefas prioritárias.

3. **Testes de aceitação:** testes elaborados pelo cliente através das estórias que descrevem as funcionalidades do sistema. Estes testes devem ser automáticos e quando rodarem com sucesso, a funcionalidade está totalmente implementada. Os testes de aceitação também oferecem *feedback*, pois através da execução dos mesmos é possível saber qual o percentual implementado de uma determinada funcionalidade

4. **Pequenos lançamentos:** liberar um novo lançamento a cada duas semanas com as funcionalidades prioritárias definidas pelo usuário para a iteração. Esta prática permite um *feedback* constante do cliente e que problemas sejam detectados cedo.

5. **Design Simples:** o sistema deve ser projetado tão simples quanto possível. A complexidade extra é removida assim que detectada. Implementar apenas a funcionalidade que foi solicitada;

6. **Programação em pares:** Dois programadores utilizam apenas um computador e se alternam na operação do mesmo. Com isso, a interação e a troca de conhecimento entre a equipe é intensa, a revisão de código é constante e consequentemente, o código gerado é de maior qualidade.

7. **Desenvolvimento guiado por testes:** nesta filosofia de codificação, os programadores escrevem os testes primeiro, escrevem o código e os testes automatizados são executados para validar o código escrito. Dessa forma, o código é testado continuamente. Os testes também servem como documentação do sistema, pois mostram como uma determinada funcionalidade deve ser utilizada.

8. **Refatoração:** processo que permite a melhoria contínua do código. Programadores reestruturam o código sem alterar a sua finalidade, removendo código duplicado, funções desnecessárias , deixando-o mais coeso e menos acoplado.

9. **Integração contínua:** testar a integração ao sistema de cada funcionalidade nova implementada. Esta prática deve ser executada diversas vezes durante o dia.

10. **Padrões de codificação:** o código escrito deve seguir um padrão definido pela equipe. Todo o código parece que foi escrito por um único indivíduo, competente e organizado

11. **Uso de metáforas:** O uso de analogias facilita a comunicação entre a equipe e os clientes.

12. **Ritmo saudável:** estabelecer cronogramas razoáveis, onde a equipe não tenha que trabalhar mais do que quarenta horas por semana. Horas-extra podem ser necessárias em um projeto, mas não devem se tornar rotina. Trabalhar com qualidade, buscando um ritmo saudável.

### 2.2.3 Conclusão

O *XP* é uma metodologia baseada na interação contínua entre cliente e equipe, para que as necessidades do cliente sejam visualizadas e compreendidas de forma correta. Para que haja o sucesso do projeto, os seus valores e as suas práticas devem ser seguidos durante todo o clico de vida do projeto.

## 2.3 Scrum

*Scrum* é uma metodologia ágil para gerenciamento de projetos criada por Ken Schwaber (SCHWABER; BEEDLE, 2002), Jeff Sutherland e Mike Beedle na década de 90, que fornece práticas que ajudam gerentes a tornar mais dinâmico e gerenciável o ambiente de desenvolvimento de *software*.

Ela é baseada em ciclos de 30 dias chamados *Sprints*, onde se trabalha para alcançar objetivos bem definidos. Estes objetivos são representados no *Product Backlog*, uma lista todas as funcionalidades para fazer que é constantemente atualizada e repriorizada (SANCHEZ, 2007).

### 2.3.1 Papéis

Dentro da metodologia *Scrum*, os seguintes papéis são assumidos (SOFTHOUSE, 2007):

**Equipe:** responsável por entregar soluções, geralmente é formada por um grupo pequeno (entre 5 e 9 pessoas) e que trabalha de forma auto-gerenciada;

**Product Owner:** responsável pela visão de negócios do projeto, é ele quem define e prioriza o *Product Backlog*. Geralmente é o papel desempenhado pelo cliente;

**Scrum Master:** é uma mistura de gerente, facilitador e mediador. Seu papel é remover obstáculos da equipe e assegurar que as práticas de *Scrum* estão sendo executadas com eficiência.

### 2.3.2 Dinâmica

De forma bastante resumida, o *Scrum* funciona da seguinte forma:

1. **Definição do *Backlog*:** todas as funcionalidades ou mudanças no produto são definidas pelo *Product Owner* no *Product Backlog*. Esta lista é priorizada para refletir a necessidade dos clientes ou demandas do mercado. Os itens do topo da lista são destacados para serem entregues no final do próximo *Sprint*.
2. **Andamento do *Sprint*:** durante o *Sprint*, os itens do *Product Backlog* que devem ser entregues são agora tratados no *Sprint Backlog*. As tarefas agora são responsabilidade da equipe, que tem autonomia para decidir como elas devem ser executadas.
3. **Reuniões Diárias:** o *Scrum Master* se reune diariamente com a equipe num mesmo horário, para que se reporte:

   O que foi feito ontem?

   O que se pretende fazer hoje?

   Quais são os impedimentos que estão atrapalhando a execução das tarefas?
4. **Revisões:** no final do *Sprint* a e demonstra os resultados para o *Product Owner* e demais interessados, de forma que os itens do *Backlog* sejam considerados prontos e então possa se iniciar um novo *Sprint*.

## 2.4 Lean Software Development

É um conjunto de princípios e práticas que teve a sua origem na indústria automobilística japonesa nos anos 90. O uso do *Lean* possibilitou que as indústrias automobilísticas japonesas desenvolvessem um novo modelo de carro em um terço do tempo utilizado pelas indústrias automobilísticas norte-americanas (POPPENDIECK; POPPENDIECK, 2003).

### 2.4.1 Princípios

O LSD (Lean Software Development) pode ser considerado como a aplicação dos princípios do *Lean* ao desenvolvimento de *software*, sendo baseado em oito princípios (POPPENDIECK; POPPENDIECK, 2003):

1. **Iniciar cedo:** desenvolvimento deve iniciar assim que houver informações suficientes
2. **Aprender constantemente:** o objetivo não é ser capaz de prever o futuro, mas sim estar pronto para responder ao futuro a medida que se desdobra.
3. **Adiar Compromissos:** as decisões irreversíveis devem ser adiadas em domínios que envolvam incerteza, porque a equipe de desenvolvimento permanece com um maior número de opções disponíveis por mais tempo e as decisões apresentam uma melhor qualidade quando tomadas com base em fatos do que em previsões.
4. **Entregar Rápido:** o foco do desenvolvimento de *software* tem que mudar do "não cometer erro" para "maior qualidade e rapidez" ao desenvolver. Entregar ao cliente o que ele precisa hoje e não o que era preciso ontem.
5. **Eliminar desperdício:** evitar esforços em atividades que não produzem valor adicinal;
6. **Equipe com poder de decisão:** ninguém melhor para tomar decisões técnicas do que a equipe de desenvolvimento, por combinar o conhecimento dos detalhes com o poder de várias mentes trabalhando juntas.
7. **Construindo com Integridade:** não há espaço para trabalho descuidado, sendo assim, uma completa estrutura de teste e controle dos requisitos dos clientes deve fazer parte do desenvolvimento.

8. **Evitar subotimização:** o sistema deve ser visto como um todo, evitando a divisão em pequenas partes o que pode acarretar uma super-otimização de um pedaço do sistema enquanto o sistema como um todo fica sub-otimizado.

## 2.5 Dynamic System Development Method

O *DSDM* é um *framework* que fornece uma maneira ágil de trabalhar com o desenvolvimento de sistemas, tendo como foco o fornecimento de uma solução de qualidade em um curto espaço de tempo . Considera que o desenvolvimento deve combinar o conhecimento sobre o domínio do negócio do cliente com o conhecimento técnico da equipe de TI (Tecnologia da Informação) (METHOD, 2006).

### 2.5.1 Princípios

O *DSDM* é baseado em nove princípios que estão de acordo com os princípios ágeis (METHOD, 2006):

1. É imprescindível o envolvimento dos usuários;
2. As equipes *DSDM* devem ter autorização para tomar decisões
3. O foco está na entrega freqüente de produtos;
4. A adequação ao propósito empresarial é o critério essencial para aceitação do produto
5. O desenvolvimento incremental e iterativo é necessário para convergir em uma solução empresarial precisa
6. Todas as mudanças durante desenvolvimento são reversíveis
7. Os requisitos são escritos em alto nível
8. O teste deve estar integrado ao ciclo de vida
9. É essencial a colaboração e a cooperação entre todos os interessados

### 2.5.2 Práticas

O *DSDM* fornece diversas práticas que podem ser utilizadas em diferentes projetos, sendo que algumas delas sempre devem ser utilizadas e outras apenas de acordo com a natureza do projeto (METHOD, 2006):

***Timeboxing*:** São intervalos de tempos , que são usados de forma equivalente às iterações no *XP*.

***Moscow*:** consiste em determinar os requisitos que o projeto precisa ter (*Must Have*), deve ter (*Should have*), pode ter (*Could have*) e não terá (*won't have*).

**Prototipação:** construir protótipos desde o início do projeto para facilitar a comunicação entre a equipe de desenvolvedores e os usuários do sistema.

**Modelagem:** O *DSDM* considera que a modelagem ajuda a equipe a adquirir um bom conhecimento sobre o domínio do negócio.

***Workshops*:** O *workshop* é a principal técnica do *DSDM*, fornecendo um mecanismo para a equipe tomar decisões de qualidade em um curto espaço de tempo, devendo ser usado durante todo o projeto tanto para criar produtos, quanto para tomar decisões.

**Testes:** devem ocorrer durante todo o desenvolvimento, sendo sistemáticos e com maior número de testes feitos por usuários não técnicos.

**Gerenciamento de Configurações:** imprescindível a gerência de configuração do *software* e da documentação, visto que um dos seus princípios diz que todas as mudanças durante o projeto podem ser revertidas, o que dá uma maior liberdade para o desenvolvimento.

## 2.6 Feature Driven Development

O FDD é uma metodologia ágil e adaptativo, com foco nas fases de desenho e construção, baseada no desenvolvimento iterativo,que enfatiza aspectos de qualidade e inclui entregas frequentes, além de interagir com outras metodologias.

### 2.6.1 Práticas

O FDD (MACHADO, 2005) estabelece um conjunto de práticas que devem se seguidas para o sucesso da adoção da metodologia:

**Modelagem dos Objetos de Domínio:** é onde ocorre a construção de diagrama de classes UML, os diagramas de seqüências;

**Desenvolvimento Através de Características:** é feita a identificação das características do sistema, que são as funcionalidades definidas pelos usuários;

**Propriedade Individual da Classe:** cada classe ou conjunto de classes é de responsabilidade de um indivíduo;

**Equipes de Características:** são definidas equipes onde os componentes possuem as propriedades das classes para a construção de determinada característica;

**Inspeções:** verificar se o código ou o modelo possuem inconsistências;

**Construções Regulares:** devem ocorrer durante a execução de um conjunto de característica para detectar erros de integração;

**Administração de Configuração:** deve ser usado um sistema de controle de versão;

**Relatórios de Resultados:** devem ser disseminados para todos os membros do projeto.

### 2.6.2 Etapas do projeto

O FDD possue definições claras em relação às etapas do projeto, sendo focado no projeto e construção e é composto pelas seguintes etapas (MACHADO, 2005):

**Desenvolvimento de um Modelo Global:** quando ocorre a definição do contexto e requisitos do *software*;

**Construir uma Lista de Características:** é construída uma lista completa de todas as características do produto a ser desenvolvido;

**Planejar a Construção por Características:** são planejadas as execuções dos conjuntos de características e as classes são distribuídas para seus proprietários;

**Projetar Cada Característica:** é gerado o diagrama de seqüência detalhado, atualizado e estudado o diagrama de classes;

**Construir Cada Característica:** é realizada a implementação e os testes.

### 2.6.3 Papéis

O FDD divide os papéis da equipe em três categorias: chave, suporte e adicional.

Os papéis chaves são:

**Gerente de Projeto:** gerencia o projeto como um todo e, juntamento com o Gerente de Desenvolvimento e os Programadores Chefe planeja a ordem que as funcionalidades serão implementadas;

**Arquiteto Principal:** alguém com experiência em modelagem de objetos que tem a responsabilidade de guiar os desenvolvedores;

**Gerente de Desenvolvimento:** Assim como o Gerente de Projeto e os Programadores Chefe,é incumbido de planejar a ordem em que as funcionalidades serão implementadas, além das tarefas comuns aos Gerentes de Desenvolvimento;

**Programador Chefe:** é o responsável por liderar e distribuir tarefas entre os desenvolvedores de determinada equipe, além de ser responsável por planejar a ordem que as funcionalidades serão implementadas, juntamente com o Gerente de Projeto e o Gerente de Desenvolvimento.

**Desenvolvedores:** são coordenados por um Programador chefe. São responsáveis por uma ou mais classes chave no desenvolvimento de cada funcionalidade. Lembrando que Programadores Chefe tambem são desenvolvedores;

**Especialista no Domínio:** responsável por prover todas informações necessárias a respeito da área de domínio, visão geral e modelagem destes.

Os papéis de suporte são: Gerente de Domínio, Gerente de Versão, Especialista na Linguagem, Coordenador de Configuração, Toolsmith, Administrador de Sistema. Os papéis adicionais são: Testadores, e Escritor Técnico (MACHADO, 2005).

## 2.7 Comentários finais

Este capítulo trouxe uma visão geral de algumas das principais metodologias ágeis existentes, seus respectivos princípios e práticas para que possam ser implementadas em um ambiente de desenvolvimento tradicional.

Grande parte das metodologias ágeis procura envolver mais o cliente no desenvolvimento do sistema, de forma que este direcione o desenvolvimento do sistema para que o mesmo atenda as suas expectativas. Outra premissa das metodologias vistas é garantir versões executáveis e de qualidade do sistema cheguem com frequência ao cliente.

Para que isto ocorra, é necessário que a equipe tenha disponível ferramentas de desenvolvimento que possibilitem a implementação e o controle de sistemas de forma mais rápida, sem perder qualidade. O próximo capítulo traz sugestões de ferramentas livres que viabilizem a produção de *software* adequada aos princípios ágeis vistos anteriormente.

# Capítulo 3

# Ferramentas de apoio às Metodologias Ágeis

Conforme visto no capítulo anterior, as metodologias ágeis surgiram com o intuito de criar *software* de qualidade rapidamente. Neste capítulo serão abordadas algumas sugestões de ferramentas que ajudam a construir um *software* adotando as práticas e princípios de uma metodologia ágil.

Elas podem ser classificadas em ferramentas de desenvolvimento (linguagens de programação, *IDE*[1], *frameworks*) e ferramentas de apoio ao desenvolvimento (*CASE*[2], gerência de projetos, documentação).

Apesar de atualmente existirem ótimas linguagens de programação e *frameworks* como *Python*[3], *Ruby On Rails*[4], a escolha tanto do *Groovy* como linguagem de programação quanto do *Grails* como *framework* de desenvolvimento, se deu pelo fato de ambos serem específicos para a plataforma *Java*, sendo que a equipe de desenvolvimento tinha grande experiência nesta plataforma.

Já na parte de ferramentas de apoio ao desenvolvimento, a opção foi pelo *DotProject* como ferramenta de controle e acompanhamento de tarefas, por já ser utilizado no ambiente de desenvolvimento e do *JSPWiki* como ferramenta de documentação, pela fácil instalação e por ser feito em cima da tecnologia *Java*, o que facilitaria no caso da necessidade de uma eventual manutenção no código-fonte e

[1] http://pt.wikipedia.org/wiki/IDE
[2] http://pt.wikipedia.org/wiki/FerramentaCASE
[3] http://www.python.org/
[4] http://rubyonrails.org/

resolução de problemas. As seções a seguir trazem uma breve introdução sobre as ferramentas escolhidas. Para facilitar o entendimento das seções seguintes, Um resumo destas tecnologias e respectivas utilizações é apresentado na Figura 3.1.

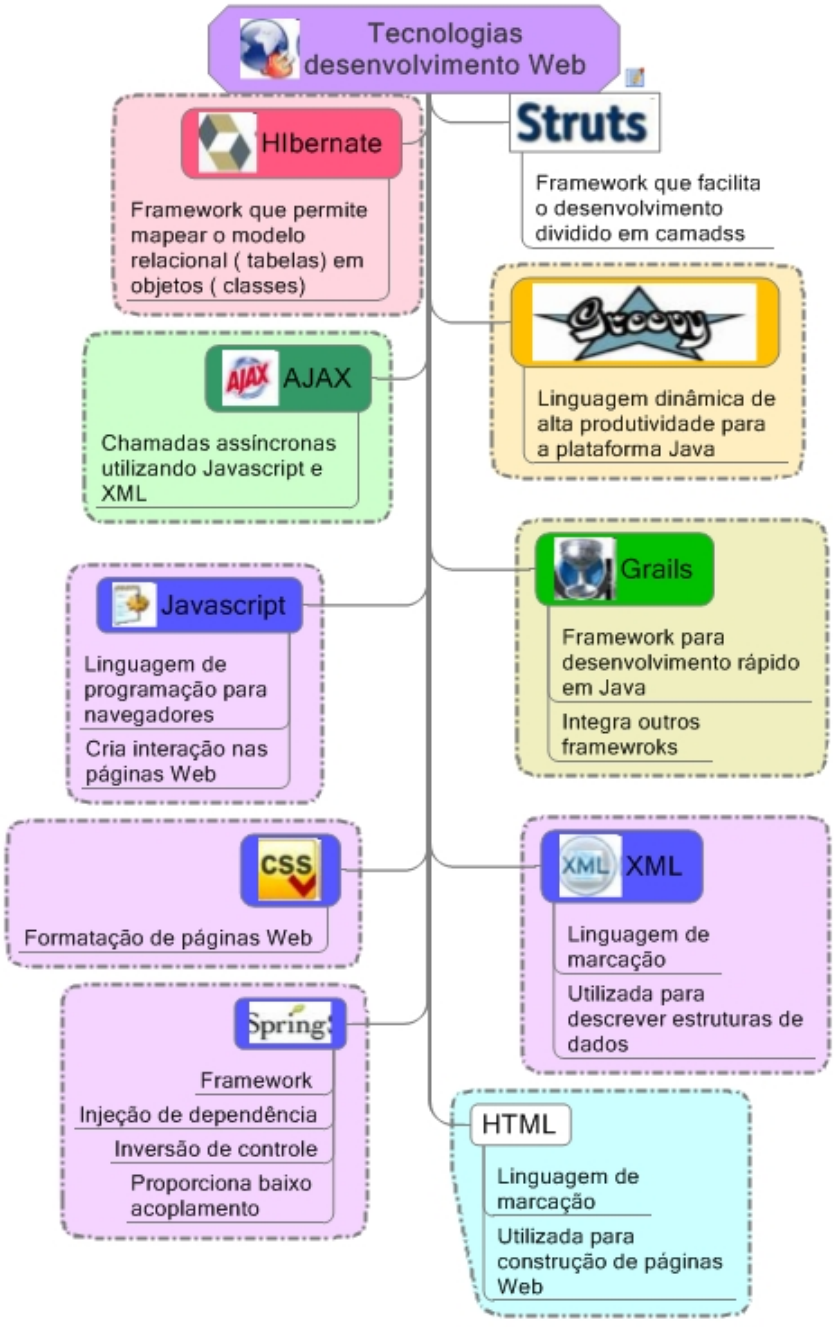


**Figura 3.1:** Tecnologias para desenvolvimento web

## 3.1 Groovy

Na última década, a plataforma *Java* conquistou grande parte do mercado de desenvolvimento de software. No entanto, apesar de todas as ferramentas de desenvolvimento que surgiram para a plataforma *Java*, os desenvolvedores nunca conseguiram o nível de produtividade exigido pelos princípios e práticas ágeis. Isto se deve em parte pela longa curva de aprendizado exigida pela tecnologia (JUDD; NUSAIRAT; SHINGLER, 2008).

Esta falta de agilidade fez com que linguagens dinâmicas como *Ruby* , *Python* crescessem e tivessem cada vez mais adeptos, dentre os quais se destacavam aqueles que seguiam as metodologias ágeis (JUDD; NUSAIRAT; SHINGLER, 2008).

Com o intuito de suprir estas deficiências da plataforma, um grupo dentro da comunidade *Java* percebeu a necessidade de trazer a produtividade e flexibilidade oferecidas pelas linguagens dinâmicas para a tecnologia *Java*, procurando manter a compatibilidade com a *API* existente. Assim surgiu o *Groovy*, uma linguagem dinâmica projetada para ser executada na *Java Virtual Machine*.

As subseções a seguir mostram algumas das características do *Groovy* traçando um paralelo com a linguagem *Java*.

### 3.1.1 Características

O *Groovy* é uma linguagem de programação dinâmica relativamente nova, que pode ser compilada ou interpretada e foi projetada especificamente para a plataforma *Java*. Como a linguagem já foi projetada tendo a máquina virtual *Java* em mente,a curva de aprendizado para desenvolvedores *Java* é mínima. Os desenvolvedores *Java* que optarem pelo *Groovy* têm acesso a toda *API Java*, sem a necessidade de aprender uma nova *API*. Além de poder acessar toda *API Java*, o *Groovy Development Kit* (GDK) estende a *API* existente com novas classes e métodos utilitários. Pelo fato de ter sido projetado para *JVM*, existe uma estreita integração em nível de *bytecode*, que facilita bastante integrar *Java* ao *Groovy* e vice-versa (JUDD; NUSAIRAT; SHINGLER, 2008).

### 3.1.2 Exemplo

Esta seção mostra uma simples classe escrita em *Java* e a seguir a mesma classe escrita em *Groovy*, para que possa ficar mais evidente os ganhos com a linguagem *Groovy*.

Conforme pode ser visto, o primeiro grande ganho é número de linhas de código necessário em cada uma das linguagens. Na linguagem *java* são necessárias 43 linhas (Figura 3.2), contra 12 (Figura 3.3) em *Groovy*. Isto se deve a algumas caracteristicas interessantes do *Groovy*: os métodos de acesso a atributos (leia-se getters e setters) são implícitos. Outra diferença é a forma de declarar um *Array* em *Java* (linhas 33 a 36 da Figura 3.2) e em *Groovy* (linhas 5 a 9 da Figura 3.3), onde esta se mostra muito mais simples e legível. E por fim, no código *Groovy* não existe a necessidade do método *main* (linhas 32 a 43 da Figura 3.2). Isto porque em *Groovy*, podemos transformar uma classe em um *script* executável. Para isto, basta colocar o código que deseja executar após a definição da classe (linhas 5 a 12 da Figura 3.3) e renomear o arquivo para um nome diferente da classe. Com este simples exemplo, já foi possível ver alguns dos ganhos utilizando *Groovy* ao invés da *Java*. Para aprender mais sobre *Groovy*, o leitor pode consultar (JUDD; NUSAIRAT; SHINGLER, 2008), (KOENIG *et al.*, 2007) e (COMMUNITY, 2008).

## 3.2 Grails

O desenvolvimento de aplicações *Web* é complexo. Com a enorme gama de tecnologias (*XML, Web Services, HTML, CSS, Java, Javascript,* AJAX) e praticamente uma nova tecnologia surgindo a cada semana, o desenvolvimento deste tipo de aplicação tem se tornado uma tarefa árdua (JUDD; NUSAIRAT; SHINGLER, 2008). Para facilitar um pouco esta tarefa, surgiram os *frameworks*, que são conjuntos de bibliotecas que ajudam a construir aplicações de forma mais rápida. No topo das escolhas de *frameworks*, estão os *frameworks* baseados no modelo *MVC* [5] e os *frameworks* AJAX[6] (JUDD; NUSAIRAT; SHINGLER, 2008) .

A comunidade *Java* trouxe a plataforma JEE[7] para o desenvolvimento de aplicações robustas e escaláveis. Apesar de se mostrar adequada nestes quesitos, a plataforma *Java* não possibilita o desenvolvimento de aplicações rápidas. Isto se deve a longa curva de aprendizado exigida pela plataforma. Assim, surgiram al-

[5] http://en.wikipedia.org/wiki/MVC
[6] http://pt.wikipedia.org/wiki/AJAX
[7] http://java.sun.com/javaee/technologies/javaee5.jsp

```
1  import java.util.List;
2  import java.util.ArrayList;
3  import java.util.Iterator;
4
5  public class Todo {
6    private String name;
7    private String note;
8
9    public Todo() {}
10
11   public Todo(String name, String note) {
12     this.name = name;
13     this.note = note;
14   }
15
16     public String getName() {
17     return name;
18   }
19
20   public void setName(String name) {
21     this.name = name;
22   }
23
24   public String getNote() {
25     return note;
26   }
27
28   public void setNote(String note) {
29     this.note = note;
30   }
31
32   public static void main(String[] args) {
33     List todos = new ArrayList();
34     todos.add(new Todo("1", "one"));
35     todos.add(new Todo("2", "two"));
36     todos.add(new Todo("3","three"));
37
38     for(Iterator iter = todos.iterator();iter.hasNext();) {
39       Todo todo = (Todo)iter.next();
40         System.out.println(todo.getName() + " " + todo.getNote());
41     }
42   }
43 }
```

**Figura 3.2:** Classe Todo em Java

```
1  public class Todo {
2    String name
3    String note
4  }
5  def todos = [
6    new Todo(name:"1", note:"one"),
7    new Todo(name:"2", note:"two"),
8    new Todo(name:"3", note:"three")
9  ]
10 todos.each {
11   println "${it.name} ${it.note}"
12 }
```

**Figura 3.3:** Classe Todo em Groovy

guns *frameworks* com um nível de abstração que a tecnologia *JEE* não oferecia. Dentre eles, os mais famosos e utilizados são o *Struts*[8], *Spring*[9] e o *Hibernate*[10].

Acontece que a integração destes *frameworks* também não é simples. E é aí que entra o *Grails*, um *framework* de código aberto que facilita a integração destes *frameworks* e empacota as melhores práticas de desenvolvimento, como testes frequentes e outras práticas sugeridas pelas metodologias ágeis.

Aliado a isto, o *Grails* se aproveita de todas as vantagens oferecidas pelo *Groovy*, que é a sua linguagem padrão (JUDD; NUSAIRAT; SHINGLER, 2008).

### 3.2.1 Características

Dentre as principais características e benefícios oferecidos pelo *Grails* , vale a pena destacar(JUDD; NUSAIRAT; SHINGLER, 2008) :

**Convenções ao invés de configurações:** a maioria dos *frameworks* utilizados tem como forma de configuração arquivos *XML*. No entanto, ao utilizar 3,4 *frameworks*, cada um com seus arquivos de configuração, esta tarefa se torna extremamente dispendiosa e difícil de administrar. Para contornar este problema, o *Grails* prefere a utilização de convenções, como estrutura de diretórios pré-definida, padrão no nome de artefatos, etc ;

[8] http://struts.apache.org/
[9] http://www.springframework.org
[10] http://www.hibernate.org/

**Testes unitários:** Sempre que uma classe de domínio é gerada, o *Grails* gera automaticamente os testes para a mesma. Além disso, o *Grails* fornece testes de integração (WIKIPEDIA, 2008a) e funcionais (WIKIPEDIA, 2008b);

***Scaffolding*:** a partir de uma classe de domínio, o *Grails* gera automaticamente funcionalidades *CRUD*[11] para a mesma;

**Mapeamento objeto-relacional:** possibilita mapear objetos para tabelas em um modelo relacional de banco de dados;

***Plugins*:** a comunidade oferece uma infinidade de *plugins* com funcionalidades não oferecidas previamente pelo *Grails*;

**Integração dos principais frameworks *Java* de código aberto:** a arquitetura do *Grails* integra alguns dos melhores e consolidados frameworks e linguagens de programação: *Groovy*, *Spring*, *Hibernate*, *Sitemash*[12], *script.aculo.us*[13], *Rico*[14], and *Prototype*[15].

## 3.3 dotProject

O *dotProject* é um sistema de gerência de projetos em *software* livre de fácil utilização, com um conjunto de funcionalidades e características que o tornam indicado para implementação em ambientes corporativos, pois atende a diversas necessidades de gerentes de projetos.

O *dotProject* é uma aplicação *web* , e seu acesso é feito através de um navegador, assim sua utilização independe de sistema operacional e instalação na máquina do usuário, pois é executado em um servidor. O *dotProject* é um sistema escrito em *PHP*[16], que utiliza banco de dados *MySQL*[17] (WIKIPEDIA, 2009).

[11] http://pt.wikipedia.org/wiki/CRUD
[12] http://www.opensymphony.com/sitemesh/
[13] http://script.aculo.us
[14] http://openrico.org
[15] http://www.prototypejs.org
[16] http://www.php.net
[17] http://www.mysql.com

### 3.3.1 Características

O *dotProject* possui várias funcionalidades interessantes que auxiliam no gerenciamento de um projeto(WIKIPEDIA, 2009):

**Cadastro de usuários:** há um cadastro para usuários de uma empresa com recursos de autorização e possibilidade de integração com bases de usuários já existentes (LDAP);

**Informações de projetos de cada empresa:** centraliza o controle de todos os projetos de uma empresa;

**cadastro das tarefas necessárias à execução de cada projeto:** todas as tarefas de um projeto podem ser cadastradas, podendo se atribuídas a usuários e colaboradores previamente cadastrados no projeto. Além disso, é possível acompanhar o andamento de cada tarefa, o que facilita o feedback;

**controle de cronograma;**

**Abertura de chamados:** pode-se cadastrar e acompanhar um chamado (possivelmente de um cliente) em um projeto. Todos os responsáveis pelo projeto são automaticamente notificados pelo sistema.

## 3.4 JspWiki

### 3.4.1 Wiki

O *wiki* é um ambiente *web* para edição de hipertextos e documentos de forma colaborativa. O seu acesso é feito através de um navegador, e a edição dos hipertextos é feita *online* diretamente no navegador. A idéia é qualquer usuário possa alterar os documentos, mas para que o ambiente não se torne um caos, é aconselhável que exista um administrador para organizar o *wiki* da maneira mais harmoniosa e gerenciar as modificações de conteúdo no mesmo.

### 3.4.2 JspWiki

O *JspWiki* é uma implementação do ambiente wiki através da tecnologia JEE e possui as seguintes características (JSPWIKI, 2008):

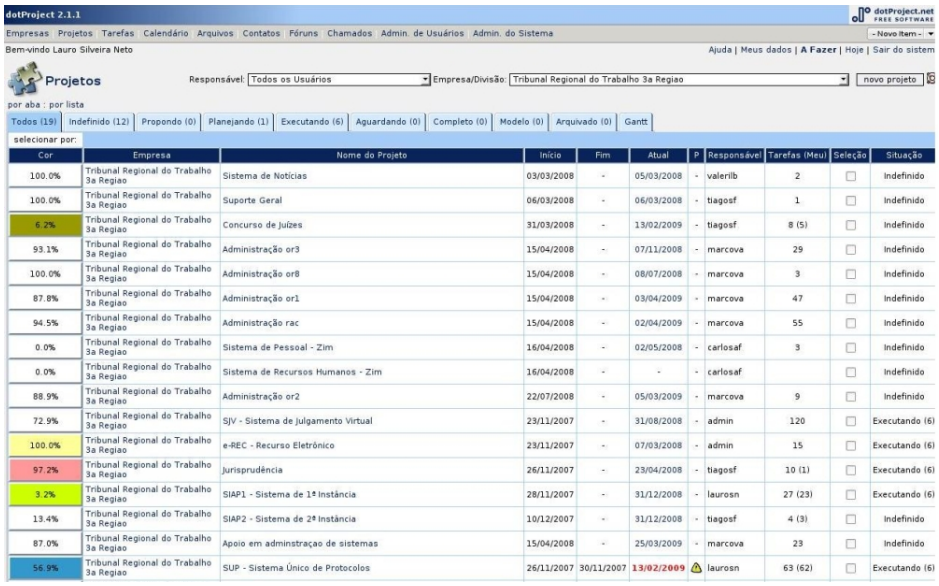


**Figura 3.4:** Ambiente do dotProject

**Linguagem de marcação:** muito simples e permite a criação de documentos rapidamente;

**Anexar arquivos:** permite que o usuário anexe arquivos (imagens, textos, etc) que sejam pertinentes ou complementares ao conteúdo do documento sendo criado;

**Segurança :** é possível definir privilégios em nível de *wiki* ou em nível de página. Estes privilégios podem ser atribuídos a grupos ou usuários;

**Plugins:** possui vários *plugins* de fácil instalação que permitem adicionar funcionalidades extra;

**Fácil instalação:** para distribuições baseadas em Debian[18], basta utilizar o apt-get[19];

**Armazenamento:** pode ser feito tanto em sistema de arquivos como em sistemas de banco de dados;

**Resolução de conflito de edição através do *lock* de página:** caso dois usuários estejam editando um arquivo simultaneamente, o aplicativo utiliza um

[18] http://www.debian.org

[19] www.debian.org/doc/manuals/apt-howto/index.pt-br.html

mecanismo baseado na versão do documento para evitar que um usuário sobrescreva as alterações do outro;

**Temas:** possibilidade de modificar o visual do *wiki* alterando a configuração para um dos temas pré-existentes ou elaborando um tema próprio.


**Figura 3.5:** Ambiente do JspWiki

## 3.5 Considerações finais

Com o crescimento das metodologias ágeis, inúmeras ferramentas têm surgido com o intuito de por em prática os príncipios e as premissas sugeridas por estas metodologias. Dentre estas ferramentas, a linguagem de programação *Groovy*, o *framework* de desenvolvimento *Grails*, o *dotProject* e o *JspWiki* demonstram ter as características necessárias para auxiliar na implementação de uma metodologia ágil. No próximo capítulo será visto como utilizar estas ferramentas em conformidade com alguns princípios e práticas propostas pelas metodologias ágeis.

# Capítulo 4

# Refatoração do sistema com técnicas ágeis

Este capítulo apresenta a refatoração de um sistema desenvolvido com metodologia tradicional com a utilização de técnicas e ferramentas ágeis apresentadas anteriormente. Primeiramente o sistema em questão será brevemente descrito e a seguir a refatoração, explanada através da comparação de alguns aspectos como codificação, documentação, testes e gerenciamento e controle de tarefas.

## 4.1 Sistema de concurso de Juízes

O sistema de concurso de juízes é um sistema relativamente simples, e por este motivo foi escolhido como piloto na adoção e utilização de técnicas e ferramentas ágeis. A Figura 4.1) apresenta a estrutura até então utilizada para a construção do sistema de concurso de juízes. Este sistema foi desenvolvido tendo como base

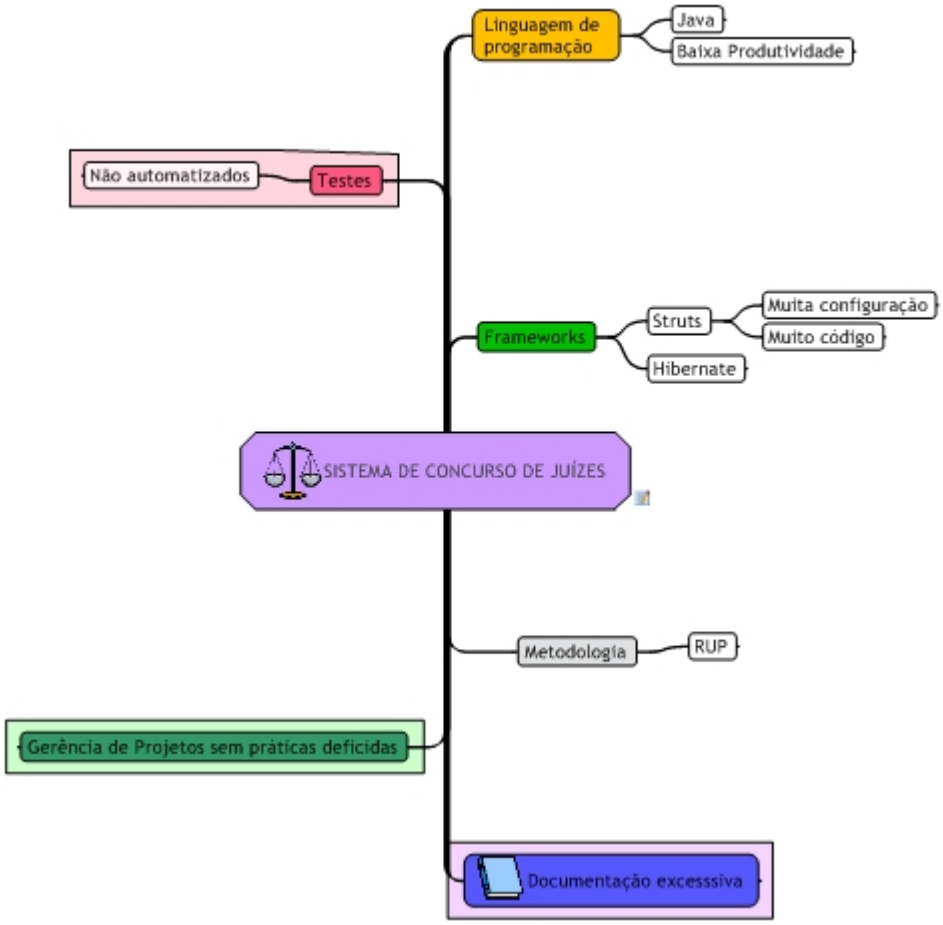


**Figura 4.1:** Estrutura de desenvolvimento do sistema de concurso de juízes

a metodologia *RUP*[1]. Na parte de tecnológica, utilizou-se a tecnologia *Java* e o modelo *MVC*[2]. Para a implementação do modelo *MVC*, foram adotados os *frameworks Struts*[3], para a parte de controle, o *Hibernate*[4] para mapeamento objeto-relacional e persistência de dados e o *framework Tiles*[5] e *JSPs*[6] para a visualização dos dados. O sistema de concurso de juízes possui as seguintes funcionalidades:

- Cadastro/manutenção de candidatos (Figura 4.2), concursos, provas e locais de prova;
- Distribuição de candidatos nos locais de prova;
- Controle de notas;
- Relatórios;
- Controle de acesso aos módulos baseado em papéis.

## 4.2 Comparativo

Esta seção mostra a forma como uma determinada característica foi realizada no sistema legado, como foi refatorada no sistema atual e os eventuais benefícios trazidos pela aplicação de técnicas e ferramentas ágeis (Figura 4.3)).

[1] http://www.ibm.com/software/br/rational/rup.shtml
[2] http://java.sun.com/blueprints/patterns/MVC-detailed.html
[3] http://struts.apache.org/
[4] http://www.hibernate.org/
[5] http://tiles.apache.org/
[6] http://java.sun.com/products/jsp/

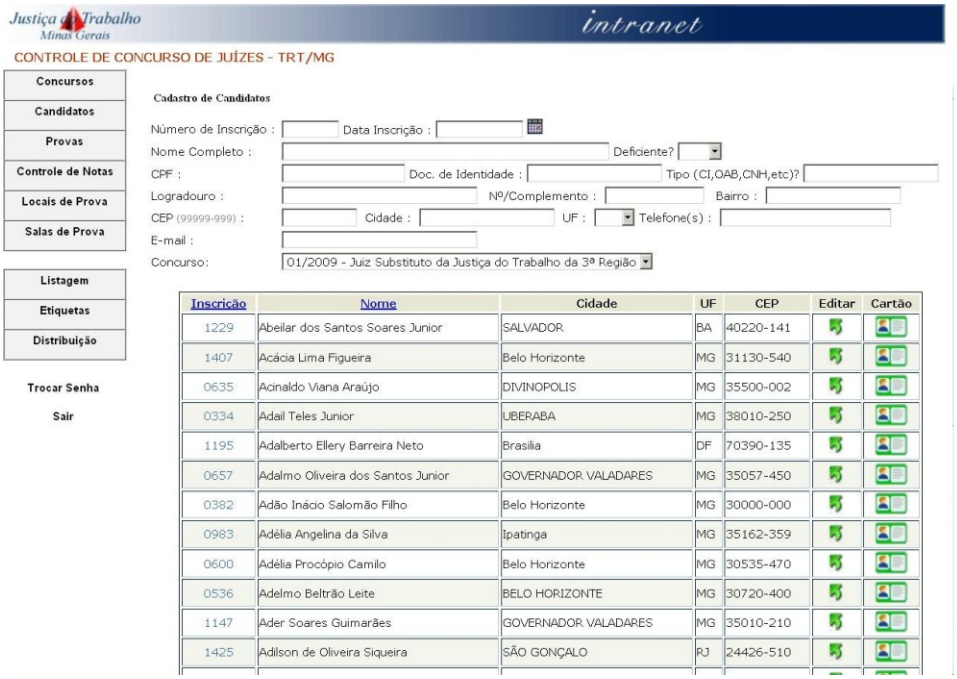


**Figura 4.2:** Sistema de concurso de juízes

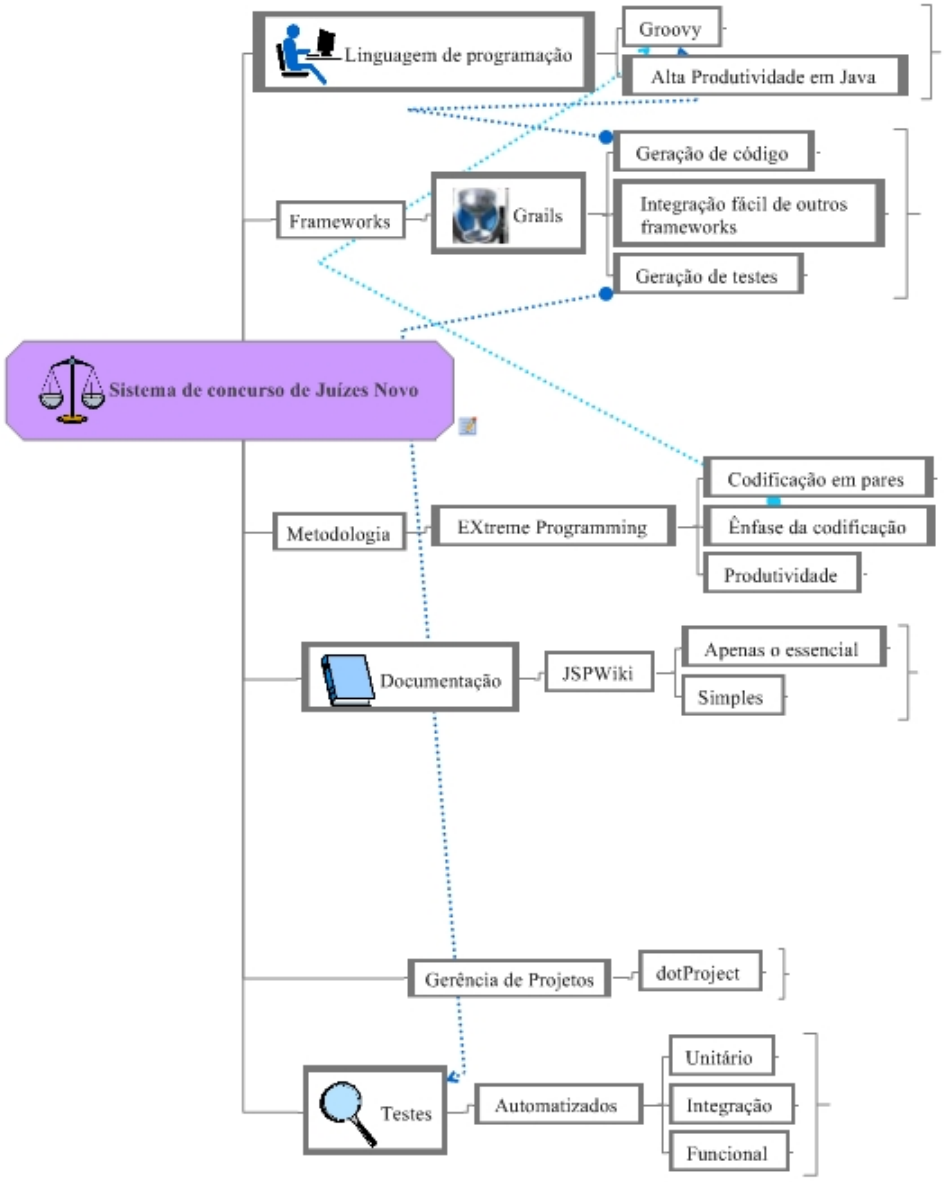


**Figura 4.3:** Estrutura nova de desenvolvimento do sistema de concurso de juízes

### 4.2.1 Codificação

Para exemplificar os benefícios obtidos com a refatoração, será utilizada a funcionalidade de cadastrar e manter candidatos. Na estrutura utilizada até então, para se gerar um *CRUD*[7], os seguintes passos são necessários, levando-se em conta que o modelo já esteja criado no banco de dados:

1. Criar página *JSP* para criação e manutenção de um candidato;
2. Alterar configuração do *Tiles*, adicionando a entrada relativa à *JSP* criada;
3. Criar classe de domínio e mapeamento objeto-relacional, necessários para o *Hibernate*;
4. Criar classe para implementação das regras de negócio e interação com banco de dados (*DAO*);
5. Alterar arquivo de configuração do *Hibernate*, adicionando o mapeamento criado;
6. Criar classe controladora do *Struts (Action)*, que farão a interação entre a camada de visão e de negócio, validação de dados;
7. Criar classe que fará a conversão de dados da *JSP* para uma classe de negócio e vice-versa;
8. Criar a entrada no arquivo de configuração do *Struts*, relativa à classe controladora criada.

O código-fonte relativo aos passos descritos anteriormente pode ser conferido no Apêndice A.

Para a criação da mesma funcionalidade utilizando o *framework Grails*, são necessários os seguintes passos:

1. Criação da classe de domínio, fazendo o mapeamento objeto-relacional na própria definição da classe;
2. Gerar o controlador e a parte de visão através da execução do comando *grails generate-all candidato*.

[7] http://pt.wikipedia.org/wiki/CRUD

Após a execução do segundo comando, o *Grails* gera uma tela completamente funcional para a execução de operações *CRUD* na classe de domínio. Isto é possível graças ao mecanismo conhecido como *scaffolding* no *Grails*, que nada mais é do que a capacidade de gerar artefatos para satisfazer um determinado conjunto de requisitos (ADBDUL-JAWAD, 2009). Para visualizar o código implementado e gerado pelo *Grails*, consulte o Apêndice B. Fazendo um breve comparativo para o mesmo requisito implementado com as duas tecnologias, duas métricas valem a pena serem destacadas:

**Número de artefatos:** no desenvolvimento com a tecnologia Java tradicional, foi necessário criar e modificar nove artefatos manualmente, enquanto na funcionalidade feita com o *Grails* apenas um artefato precisou ser criado manualmente;

**Linhas de código:** no legado, foram necessárias 931 linhas, sendo que nenhuma delas foi gerada automaticamente. Na implementação com *Grails*, foram necessárias 721 linhas de código, sendo que destas, apenas 83 precisaram ser feitas manualmente e o restante do código foi inteiramente gerado pelo *Grails*.

Conforme pode ser verificado, analisando estas duas métricas, o *Grails* se mostrou muito mais produtivo e eficiente para implementar um mesmo requisito utilizando o ambiente de desenvolvimento *Java* tradicional. Vale a pena destacar que além do *scaffolding*, o *Grails* possui muitas outras funcionalidades (segurança, relatórios, mecanismos de busca) que podem ser obtidas a partir da utilização de *plugins*, além de todo dinamismo e flexibilidade oferecidos pela linguagem *Groovy*. Outro ponto interessante é que o *Grails* se encaixa perfeitamente em ambientes que já utilizam a tecnologia *Java* para desenvolvimento *Web*, pois além de rodar na mesma máquina virtual, sua arquitetura utiliza grande parte dos *frameworks* consagrados na comunidade ( *Spring, Hibernate, Sitemesh, etc*), o que torna a adoção do mesmo muito menos traumática.

Também foram realizadas algumas experiências como a realização de sessões de programação em pares, conforme sugere o *XP*. Pelo fato da equipe já ter um bom entrosamento e possuirem a mesma estrutura de desenvolvimento, os resultados foram satisatórios. Em algumas situações, um detectava erros de programação do outro, dava alternativas de implementação para um problema ou mesmo indicava a solução para um problema que já havia passado.

### 4.2.2 Documentação

O sistema legado foi desenvolvido seguindo boa parte do processo de desenvolvimento proposto pelo *RUP*. Sendo assim, boa parte dos artefatos gerados são documentos. Dentre os documentos gerados durantes as fases de desenvolvimento do sistema de concurso de juízes, podem ser citados:

**Documento de visão:** Os requisitos principais, as características-chave e as principais restrições do projeto foram documentados;

**Glossário:** Termos importantes definidos;

**Modelo de caso de uso:** modelo das funções pretendidas do sistema e seu ambiente;

**Documento de arquitetura de software:** fornece uma visão geral de arquitetura abrangente do sistema, usando diversas visões de arquitetura para descrever diferentes aspectos do sistema;

**Modelo de dados:** descreve a representação lógica e física dos dados persistentes no sistema. Também abrange qualquer comportamento definido no banco de dados;

**Diagramas UML (classe, alguns de sequência, caso de uso):** representam de forma visual os modelos ou regras de negócio, facilitando o entendimento de todos os envolvidos;

**Plano de testes:** contém informações sobre a finalidade e as metas dos testes no projeto. Além disso, ele identifica as estratégias a serem usadas para implementar e executar os testes, além de identificar os recursos necessários.

Apesar de alguns documentos serem interessantes para a comunicação entre a equipe e com o cliente, muitos deles eram elaborados apenas porque o processo de desenvolvimento sugeria a elaboração dos mesmos durante determinada fase do desenvolvimento. O que se percebeu no dia a dia, é que os desenvolvedores pouco consultavam os documentos criados e no entanto tinham que atuálizá-los constantemente. Segundo (AMBLER, 2004), qualquer tempo gasto produzindo documentação é tempo gasto não desenvolvendo novas funcionalidades para o usuário.Isto vai de encontro a uma das premissas do Manifesto Ágil, que diz que *software funcionando é mais importante do que documentação completa e detalhada* . Um outro problema que vale pena destacar, é que os documentos eram criados em cima de

templates do *Microsoft Word* [8]. A manutenção e atualização de cada um dos documentos a cada modificação em um requisito ou em um modelo de dados, tornou-se uma tarefa maçante e que tomou boa parte do tempo de desenvolvimento. Além disso, não existe a possibilidade de navegar entre os documentos. Assim, muita vezes era necessário ter vários documentos abertos para encontrar uma determinada informação.

Para resolver estas deficiências, buscou-se uma solução que satisfizesse alguns dos critérios propostos por (AMBLER, 2004) para a criação de documentos ágeis:

**Documentos ágeis são magros e econômicos:** Um documento ágil é tão simples quanto possível. O conteúdo deve ser mais importante do que a forma;

**Documentos ágeis satisfazem a um propósito:** se não souber porque está fazendo o documento, o analista deve parar e repensar o que está fazendo;

**Documentos ágeis descrevem coisas boas de saber:** documentos ágeis não devem conter informações óbvias;

**Documento ágeis são suficientemente indexados:** uma documentação não será eficaz se você não puder encontrar a informação contida nela;

**O benefício de ter documentação deve ser maior do que o custo de criá-la e mantê-la:** se a documentação se mostrar necessária, fazer o uso de ferramentas que demandem o menor esforço possível para gerá-la.

Dentre as soluções possíveis para implementar estas modificações, a ferramenta que se demonstrou mais adequada foi o *wiki*. Conforme falado anteriormente, o *wiki* é uma ferramenta que permite a criação e edição de páginas de forma colaborativa. Essas páginas podem ser disponibilizadas em um *website*, permitindo o compartilhamento de documentos e divulgação de informações relevantes ao projeto. Para a documentação do sistema de concurso de juízes a implementação de *wiki* escolhida foi o *JSPWiki*, principalmente pela sua facilidade de instalação e por ser desenvolvido em Java. Assim, como a equipe possui prévia experiência nesta tecnologia, uma eventual manutenção ou ajuste na ferramenta seriam facilitados. Dentre as suas características e funcionalidades oferecidas, podem ser citadas algumas que favorecem a criação de documentos ágeis:

- Fácil criação e edição de documentos (páginas);

[8] http://office.microsoft.com/pt-br/word/default.aspx

- Acesso centralizado e facilitado, necessitando apenas de um navegador;
- Mecanismo de busca;
- Controle versão, com histórico de todas as alterações feitas em um documento.

Com a utilização de um *wiki*, foi possível disponibilizar uma documentação simples, dinâmica e contendo apenas o estritamente necessário, conforme pode ser visto nas figuras Figura 4.4, Figura 4.5 e Figura 4.6.


**Figura 4.4:** Wiki do sistema de concurso de juízes

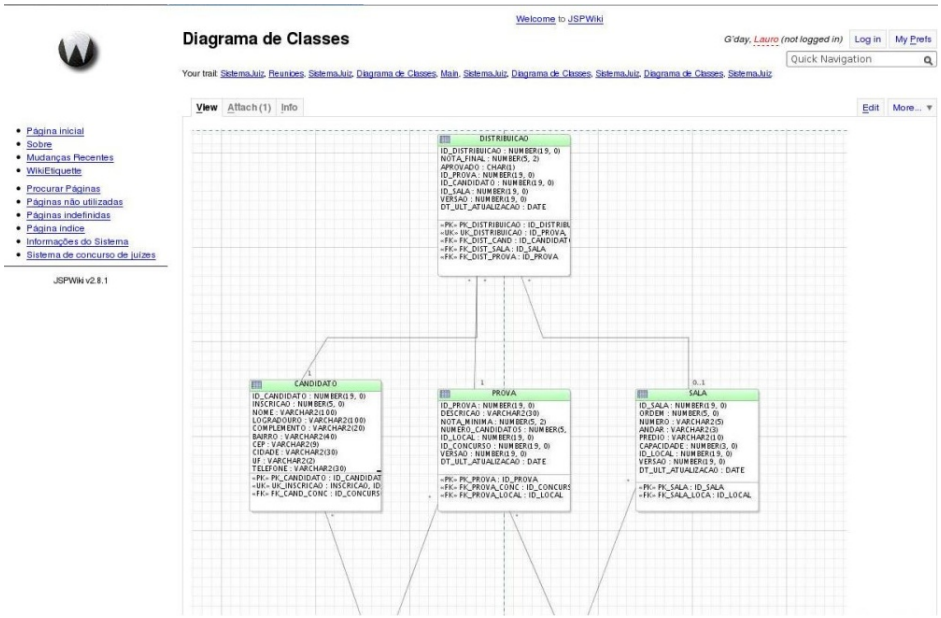


**Figura 4.5:** Visualização do diagrama de classes no wiki

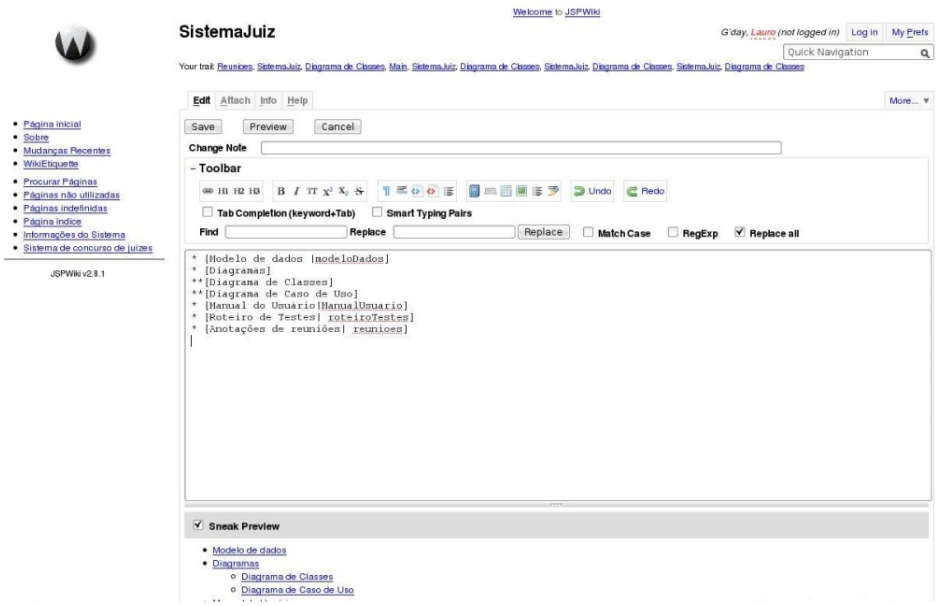


**Figura 4.6:** Edição de uma página no wiki

### 4.2.3 Testes

Segundo (KRUCHTEN, 2003), a disciplina de testes dentro do *RUP* têm os seguintes objetivos:

- Descobrir defeitos de problemas no sofwtare;
- avaliar a qualidade do software;
- validar se o software construído funciona conforme foi projetado;
- validar se os requisitos foram implementados de forma apropriada.

Dentre os tipos de teste, valem ser citados alguns comumente utilizados (KRUCHTEN, 2003):

**Teste de unidade:** A menor unidade executável do sistema á testada;

**Teste de integração:** As unidades são testadas de forma integrada;

**Teste de sistema:** O sistema como um todo é testado;

**Teste de aceitação:** A aplicação é testada por terceiros para validar se o que foi proposto nos requisitos está sendo realizado de forma apropriada no sistema.

Além dos testes outra prática utilizada é a depuração, que é a técnica de encontrar e reparar um erro com o auxílio de ferramentas após o erro ter sido detectado pelos testes. Dentro do sistema de concurso de juízes, o procedimento de testes utilizado é rudimentar: apenas o teste de aceitação e a depuração são realizados. O teste de aceitação consiste apenas em utilizar o plano de testes e verificar se cada funcionalidade proposta em um caso de uso gerava a saída esperada. Outro aspecto negativo, é que nenhuma das tarefas relacionada a testes é realizada de forma automatizada.

Conforme visto no Capítulo 2, quase todas as metodologias ágeis dão bastante importância aos testes. Com o intuito de melhorar e automatizar, foram utilizados os testes oferecidos pelos *plugins* do *Grails*: teste de unidade, teste de integração e teste funcional. Os dois primeiros são automatizados através da integração com o framework de testes *JUnit*[9] e o teste funcional através do *framework Canoo WebTest*[10] .

A Figura 4.7 mostra um exemplo teste de unidade sobre a classe de negócio candidato, onde as restrições de integridade são testadas. A Figura 4.8 é um exemplo de teste de integração, pois faz o teste sobre a operação *create*. Esta operação faz a interação do controlador com a classe de domínio candidato. Finalmente, a Figura 4.9 apresenta um teste funcional sobre a classe de domínio candidato[11]. O teste funcional simula a interação de um usuário com a interface gráfica de cadastro de usuário. Assim, é possível testar preenchimento de campos, cliques de botões e o comportamento do aplicativo a estes eventos: mensagens de confirmação, mensagens de erro, redirecionamento para outra tela, etc.

[9] http://www.junit.org/

[10] http://webtest.canoo.com/

[11] Trechos do código foram cortados por questão de espaço

```
1
2  class CandidatoTests extends GroovyTestCase {
3
4
5     /* testa as constraints da classe de dominio Candidato */
6     void testBlankFields(){
7     def c = new Candidato(nome:"")
8
9     assertFalse "Existem erros", c.validate()
10
11
12    println "\nErros:"
13    println c.errors ?: "Nao foram encontrados erros"
14
15    def badField = c.errors.getFieldError('nome')
16    println "\nCampo invalido:"
17    println badField ?: "nome nao e um campo invalido"
18  //  assertNotNull "I'm expecting to find an error on the nome
        field", badField
19    def code = badField?.codes.find {it == 'candidato.nome.blank'}
20    println code ?: "a chave com a mensagem de nome invalido nao
          foi encontrada"
21
22    def co = new Concurso(id:1, numero:"001", ano:"2000", cargo:"
          Juiz substituto", dtUltAtualizacao:new Date(),
          dataUltAlteracao:new Date(), usuarioUltAlteracao:"laurosn")
23    c = new Candidato(id:1, nome:"Lauro", inscricao:100, logradouro
          :"Rua Padre Rolim", complemento:"1003", bairro:"Sta
          Efigenia", cep:"30130090", cidade:"BH", uf:"MG", telefone:"
          3333333", email:"laurosn@gmail.com", idConcurso:co,
          deficiente:"N", deferido:"S", dtUltAtualizacao:new Date())
24    assertTrue "Nao existem erros", c.validate()
25    println "\nErros:"
26    println c.errors ?: "Nao foram encontrados erros"
27
28   }
29 }
```

**Figura 4.7:** Teste unitário para a classe de domínio candidato

```
1
2  class CandidatoControllerTests extends GroovyTestCase {
3      void setUp() {
4     new Candidato(id:1, nome:"Lauro", inscricao:100, logradouro:"
          Rua Padre Rolim", complemento:"1003", bairro:"Sta Efigenia"
          , cep:"30130090", cidade:"BH", uf:"MG", telefone:"3333333",
           email:"laurosn@gmail.com", idConcurso:co, deficiente:"N",
          deferido:"S", dtUltAtualizacao:new Date()).save()
5      }
6
7      void testCreate() {
8          def c = Candidato.findByNome('Lauro')
9          def cc = new CandidatoController()
10         cc.request.parameters = [nome:"Lauro", inscricao:100,
               logradouro:"Rua Padre Rolim", complemento:"1003",
               bairro:"Sta Efigenia", cep:"30130090", cidade:"BH", uf
               :"MG", telefone:"3333333", email:"laurosn@gmail.com",
               idConcurso:co, deficiente:"N", deferido:"S",
               dtUltAtualizacao:new Date()]
11         def candidato = cc.create().candidato
12         assertToString candidato, "$c"
13     }
14
15     void tearDown() {
16         Candidato.findByNome('Lauro').delete()
17     }
18 }
```

**Figura 4.8:** Teste de integração para a classe de domínio candidato

Conforme destacado , o *Grails* possui fácil integração com os *frameworks* de teste. Ao executar o comando *grails test-app*, os seguintes passos são realizados:

1. Executa todos os testes de unidade;
2. Executa os testes de integração;
3. Executa os testes funcionais;
4. Gera o relatório com os resultados dos testes.

Como pode ser visto, a utilização das ferramentas de teste possibilita a realização de testes de forma automatizada, poupando tempo na realização dos mesmos, redução no número de *bugs* (por realizar testes mais completos), proporcionando um código de melhor qualidade. Vale a pena destacar que a tarefa de construção

```
1  class CandidatoTest extends grails.util.WebTest {
2      void suite() {
3          testCandidatoListNewDelete()
4      }
5      def testCandidatoListNewDelete() {
6          webtest('Candidato basic operations: view list, create new
                entry, view, edit, delete, view') {
7              invoke      'candidato'
8              verifyText  'Home'
9              verifyListSize 0
10             clickLink   'New Candidato'
11             verifyText  'Create Candidato'
12             clickButton 'Create'
13             verifyText  'Show Candidato', description:'Detail page
                   '
14             clickLink   'List', description:'Back to list view'
15             verifyListSize 1
16             group(description:'edit the one element') {
17                 ant.clickLink   '1', description:'go to detail
                       view'
18                 clickButton 'Edit'
19                 verifyText  'Edit Candidato'
20                 clickButton 'Update'
21                 verifyText  'Show Candidato'
22                 clickLink 'List', description:'Back to list view'
23             }
24             verifyListSize 1
25             group(description:'delete the only element') {
26                 ant.clickLink   '1', description:'go to detail
                       view'
27                 clickButton 'Delete'
28                 verifyXPath xpath:  "//div[@class='message']",
29                             text:   /.*Candidato.*deleted.*/,
30                             regex:  true
31             }
32             verifyListSize 0
33         }
34     }
35     String ROW_COUNT_XPATH = "count(//div[@class='list']//tbody/tr
           )"
36 }
```

**Figura 4.9:** Teste funcional para a classe de domínio candidato

de testes automatizados não é simples, pois requer tempo para que os desenvolvedores ganhem a maturidade suficiente para se especializarem na construção dos mesmos. Apesar do tempo despendido com esta tarefa (muito em função do aprendizado na construção de testes automatizados), a diferença de qualidade nos testes realizados no sistema legado e nos testes automatizados foi gritante e trouxe muito mais benefícios do que ônus.

### 4.2.4 Gerenciamento e controle de tarefas

O gerenciamento de projetos tradicional está relacionado ao processo de apoio ao desenvolvimento que permita o controle dos problemas durante o ciclo de vida do projeto. Dentre as principais características do gerenciamento tradicional, podem ser citadas (RIBEIRO; ARAKAKI, 2006):

- dificuldades em responder com rapidez a mudanças impostas pelo cliente, podendo ocasionar conflitos e comprometimento de prazos;
- o gerente de projeto tem atuação forte e centralizadora, sendo o responsável pelo sucesso do projeto;
- equipe é limitada na colaboração e influência durante a execução do projeto;
- a participação do cliente é intensa apenas no início e depois se limita a validações de produtos;
- a comunicação ocorre de maneira formal;
- maior efetividade em projetos de longa duração e com equipes grandes;
- planejamento é detalhado e os envolvidos apenas o papel de validação.

O gerenciamento ágil surgiu a partir dos conceitos de desenvolvimento rápido como uma alternativa para tratar com o ambiente competitivo do mercado atual que exige resultados imediatos, sob condições de altas incertezas e constantes mudanças. Algumas características do gerenciamento ágil (RIBEIRO; ARAKAKI, 2006):

- adapta-se a projetos com mudanças constantes e que necessitam de respostas rápidas;

- mais efetivos em projetos pequenos;
- gerente de projetos tem papel de facilitador e coordenador;
- a equipe tem atuação colaborativa em todas as atividades do projeto;
- cliente é parte da equipe e atua durante o projeto ;
- planejamento é curto e com o envolvimento de todos na sua elaboração;
- comunicação informal.

Dentro da equipe que desenvolveu o sistema de concurso de juízes, apesar da equipe ser pequena, os projetos serem pequenos, a gerência de projetos se aproximava muito mais da gerência tradicional do que uma gerência ágil. De maneira resumida, o projeto foi realizado da seguinte forma:

1. Elicitação de requisitos com o cliente e delimitação do escopo do projeto (Documento de visão e casos de uso);
2. Planejamento do projeto inteiro, todo feito pelo gerente;
3. Distribuição das tarefas ;
4. Execução das tarefas;
5. Entrega do produto para o cliente;
6. Validação por parte do cliente;
7. Implantação.

Esta forma de trabalho trouxe alguns problemas para o projeto:

**Inconsistências devido à falta de comunicação:** como o usuário participava apenas no início e no final do projeto, muitos requisitos tiveram que ser reimplementados, algumas vezes pelo fato do requisito estar mal especificado e outras simplesmente porque após tanto tempo sem contato com o usuário, os desejos deste haviam mudado;

**Produtividade baixa e desperdício de tempo:** Isto se deve principalmente pela falta de comunicação entre a equipe. Em diversas situações, um desenvolvedor perdia várias horas de trabalho na resolução de um problema que outros já haviam passado. Outro problema é que em algumas ocasiões, desenvolvedores ficavam ociosos, pois já haviam acabado as suas tarefas porém o seu prazo de término estava longe de acabar;

**Perda de tempo na resolução de problemas (*bugs*) no sistema:** a comunicação necessariamente tinha que ser feita de maneira formal para solicitar uma manutenção no sistema. O setor do usuário encaminhava um documento impresso e assinado pelo superior para o setor de desenvolvimento. O documento chegava ao gerente do projeto, que então passava o problema para o responsável. Muitas vezes esta solicitação demorava alguns dias para chegar na mão do responsável. A comunicação de resolução do problema seguia o caminho inverso;

**Redundâncias dentro do sistema:** Como os membros da equipe tinham pouca comunicação, o mesmo problema às vezes era solucionado mais de uma vez.

Para solucionar e amenizar os problemas citados anteriormente, algumas práticas foram adotadas:

**Participação do usuário durante o projeto inteiro:** Na medida do possível, o usuário tinha contato diretamente com equipe semanalmente, o que permitia o *feedback* constante do mesmo para ver se o que estava sendo feito atendia as sua expectativas;

**Planejamento curto:** Planejar para no máximo uma iteração de duas semanas, com as prioridades do que seria feito definidas pelo cliente. Com isso, mudanças nos requisitos não acarretariam em mudar o planejamento do projeto todo. Para o cadastro dos projeto e controle das tarefas foi utilizado o *dotProject*. Com o mesmo é possível cadastrar as tarefas , atribuí-las ao responsável e acompanhar o andamento de cada uma delas (a Figura 4.10 mostra o exemplo de uma iteração e suas respectivas tarefas) ;

**Reuniões diárias:** Esta foi uma prática absorvida do *Scrum*. Uma reunião diária de quinze minutos onde os participantes da equipe respondiam as seguintes questões: O que foi feito ontem? O que se pretende fazer hoje? Quais são os impedimentos que estão atrapalhando a execução das tarefas?. Respondendo estas questões foi possível corrigir problemas como produtividade

baixa e desperdício de tempo, já que se algum se algum membro da equipe estivesse com algum problema impedindo a execução das sua tarefas, todo a equipe se concentraria na solução do problema. Outro problema amenizado por estas reuniões diárias foi o desperdício de tempo, pois caso alguém estivesse adiantado e houvesse terminado as suas tarefas, isto seria comunicado na reunião e poderia se dedicar a outras tarefas. Um resumo destas reuniões diárias é disponibilizado no *wiki* do projeto, para que todos possam ter acesso aos assuntos discutidos nas mesmas;

**Controle de bugs:** Para otimizar o tempo gasto com a resolução de problemas, resolveu-se utilizar funcionalidade de Chamados existente no *dotProject*. Através desta funcionalidade, o usuário cadastra o problema encontrado no sistema (Figura 4.11). A partir daí, os membros da equipe responsáveis pelo sistema recebem uma notificação do *bug* cadastrado. Então, a cada atualização na situação do problema, todos os interessados são automaticamente comunicados (via email) até que o problema seja corrigido. Além disso, todas alterações e correções de *bugs* efetuadas são disponibilizadas no *wiki* do sistema. Com a adoção destas práticas o tempo para resolução de problemas reduziu drasticamente, já que grande parte da burocracia foi eliminada.

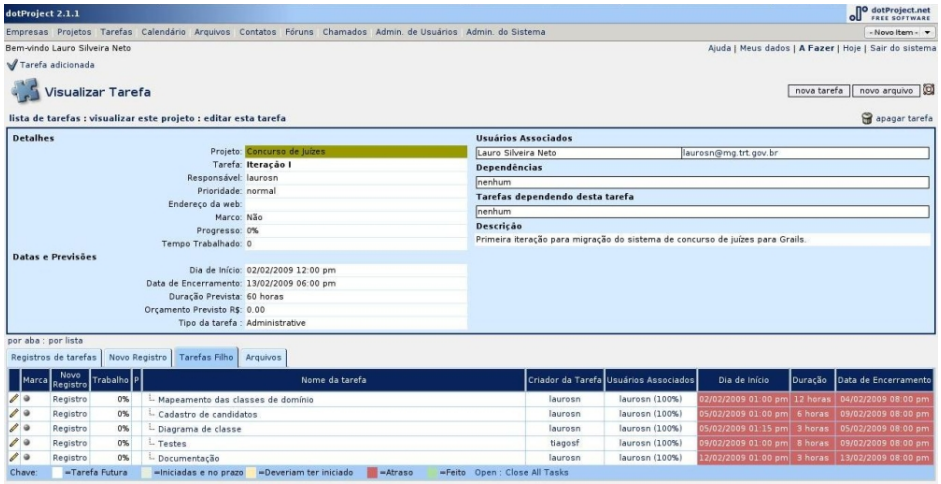


**Figura 4.10:** Controle de iterações e tarefas

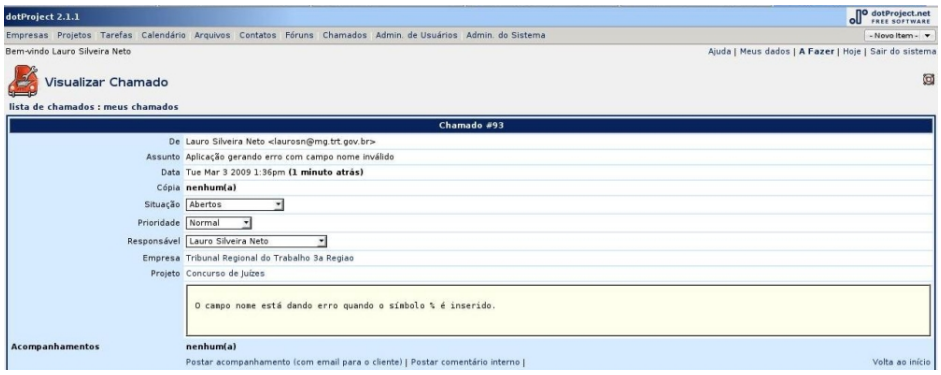


**Figura 4.11:** Cadastro de chamados no dotProject

## 4.3 Considerações finais

A aplicação de ferramentas livres como *Groovy, Grails, JspWiki* e o *dotProject* guiadas pelas práticas e princípios das metodologias ágeis pode trazer inúmeros benefícios para uma equipe de desenvolvimento de software, conforme pode ser verificado neste capítulo.

Para que isso ocorra a equipe deve ter a maturidade e experiência suficientes para identificar quais os principais problemas e situações que não contribuem para o bom andamento de um projeto. Somente assim será possível ter a base necessária para escolher as ferramentas e a metodologia que se encaixam no contexto atual.

Como os resultados obtidos foram altamente positivos, a expectativa é que em breve todos os módulos do sistema de concurso de juízes e futuros projetos adotem a estrutura de trabalho proposta neste capítulo.

# Capítulo 5

# Considerações finais

## 5.1 Conclusões

O trabalho proposto não têm a intenção de desqualificar as metodologias tradicionais de desenvolvimento, e, ao mesmo tempo, não possui a pretensão de sugerir a utilização de metodologias ágeis como a solução de todos os problemas dentro do desenvolvimento de *software*.

A idéia principal gira em torno de como aumentar a produtividade e qualidade dentro de um ambiente acostumado com uma metodologia tradicional através da implementação de práticas propostas pelas metologias ágeis e da adoção de ferramentas livres que auxiliem na implantação das mesmas.

## 5.2 Contribuições

Esta experiência permitiu verificar que apesar de não implantar por completo uma metodologia ágil, a adoção de algumas práticas sugeridas por estas com o auxílio das ferramentas livres corretas pode trazer inúmeros benefícios, tais como:

**Redução na tarefa de codificação e aumento da produtividade:** a utilização do *Groovy* trouxe todos os benefícios e facilidades de uma linguagem de *script* para o mundo Java, eliminando grande parte da verborragia que a linguagem Java exige. Já com o *Grails* foi possível gerar boa parte do código

que até então era gerado manualmente, além de integrar de forma facilitada os diversos *frameworks* existentes para a plataforma JEE;

**Uma forma de comunicação mais simples, menos burocrática e mais eficaz:** isso graças a ferramenta *JspWiki*, que através das suas características permite criação e edição de documentos de forma simples, centralizada e facilmente acessível ;

**Aumento na qualidade do código:** obtido através da automatização proporcionada pelo *Grails* de testes de unidade, de integração e funcionais;

**Maior integração da equipe:** através de práticas como codificação em pares e pequenas reuniões diárias;

**Gerência de projetos ágil:** distribuição, planejamento, acompanhamento de tarefas através do *dotProject*, que além destas funcionalidades permite o controle de *bugs* existentes e chamados abertos pelos usuários;

**Usuário mais satisfeito:** proporcionado pelo *feedback* constante, participação ativa dentro da equipe, ajudando a obter um desenvolvimento de qualidade e que traga resultados dentro das suas expectativas.

## 5.3 Trabalhos Futuros

A realização deste trabalho permitiu verificar que a utilização de metodologias ágeis com o auxílio de ferramentas livres trouxe muito mais benefícios do que problemas para uma equipe de desenvolvimento que até então adotava ferramentas mais consolidadas e uma metodologia tradicional.

No entanto, apesar dos resultados satisfatórios, apenas algumas práticas sugeridas pelas metodologias ágeis foram adotadas. Vários recursos disponíveis nas ferramentas examinadas também não foram exploradas. Acredita-se que a adoção de práticas como desenvolvimento guiado por testes (XP), *planning poker* (*Scrum*) além da exploração de recursos avançados do *Groovy* e do *Grails* possam ser objeto de um trabalho complementar a este.

# Referências Bibliográficas


ADBDUL-JAWAD, B. *Groovy and Grails Recipes: Practical anwers to your Groovy and Grails questions*. 1. ed. [S.l.]: Apress, 2009.

AMBLER, S. W. *Modelagem Ágil: Práticas eficazes para programação extrema e o processo unificado*. 1. ed. [S.l.]: Bookman, 2004.

BECK, K. *Extreme Programming Explained*. 1. ed. [S.l.]: Addison Wesley, 2000.

COMMUNITY, C. G. *Groovy Home*. 2008. [Online; acessado em 22 de janeiro de 2009]. Disponível em: <http://groovy.codehaus.org/>.

JEFFRIES, R. *What is Extreme Programming?* 2001. [Online; acessado em 13 de janeiro de 2009]. Disponível em: <http://www.xprogramming.com/xpmag-/whatisxp.htm>.

JSPWIKI. *JSPWiki: JSP Wiki Features*. 2008. [Online; acessado em 26 de janeiro de 2009]. Disponível em: <http://www.jspwiki.org/wiki/JSPWikiFeatures>.

JUDD, C. M.; NUSAIRAT, J. F.; SHINGLER, J. *Beginning Groovy and Grails: From Novice to Professional*. 1. ed. [S.l.]: Apress, 2008.

KOENIG, D.; GLOVER, A.; KING, P.; LAFORGE, G.; SKEET, J. *Groovy in Action*. 1. ed. [S.l.]: Manning Publications, 2007.

KRUCHTEN, P. *The Rational Unified Process: An introduction*. 3. ed. [S.l.]: Addison-Wesley, 2003.

LARMAN, C. *Agile and Iterative Development: A Manager's Guide*. 1. ed. [S.l.]: Addison Wesley, 2003.

LEFFINGWELL, D. *Scaling Software Agility: Best Practices for Large Enterprises*. 1. ed. [S.l.]: Addison Wesley, 2007.

MACHADO, T. L. Uma ferramenta de suporte ao framework para comparação e análise de métodos ágeis. *Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do grau de bacharel em Sistemas de Informação da Universidade Federal de Santa Catarina*, I, n. 1, 2005.

METHOD, D. S. D. *Dynamic Systems Development Method*. 2006. [Online; acessado em 14 de janeiro de 2009]. Disponível em: <http://www.dsdm.org>.

POPPENDIECK, M.; POPPENDIECK, T. *Lean Software Development: An Agile Toolkit for Software Development Managers*. 1. ed. [S.l.]: Addison-Wesley, 2003.

RIBEIRO, A. L. D.; ARAKAKI. Gerenciamento de projetos tradicional x gerenciamento de projetos Ágil: Uma análise comparativa. I, n. 1, 2006.

SANCHEZ, I. *Scrum em 2 minutos*. 2007. [Online; acessado em 13 de janeiro de 2009]. Disponível em: <http://dojofloripa.wordpress.com/2007/02/07/scrum-em-2-minutos/>.

SCHWABER, K.; BEEDLE, M. *Agile Software Development with Scrum*. 1. ed. [S.l.]: Prentice-Hall, 2002.

SOFTHOUSE. *Scrum in five minutes*. 2007. [Online; acessado em 13 de janeiro de 2009]. Disponível em: <http://www.softhouse.se/Uploades/>.

WIKIPEDIA. *Teste de caixa-preta - Wikipédia, a enciclopédia livre*. 2008. [Online; acessado em 26 de janeiro de 2009]. Disponível em: <http://pt-.wikipedia.org/wiki/Teste%5Fde%5Fintegracao>.

WIKIPEDIA. *Teste de caixa-preta - Wikipédia, a enciclopédia livre*. 2008. [Online; acessado em 26 de janeiro de 2009]. Disponível em: <http://pt-.wikipedia.org/wiki/Teste%5Fde%5Fcaixa%2Dpreta>.

WIKIPEDIA. *DotProject - Wikipédia, a enciclopédia livre*. 2009. [Online; acessado em 26 de janeiro de 2009]. Disponível em: <http://pt.wikipedia.org-/wiki/DotProject>.

# Apêndice A

# Código-fonte legado

```
1  <%@ taglib uri="/WEB-INF/c.tld" prefix="c"%>
2  <%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic"%>
3  <%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>
4  <%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean"%>
5  <%@ taglib uri="/WEB-INF/displaytag-12.tld" prefix="display"%>
6
7  <html:hidden property="identificador"/>
8
9    <table align="center" width="100%" >  <!--width="800px"-->
10
11   <!-- Primeira linha de campos -->
12   <tr>
13     <td> <bean:message key="label.cadastro.candidato.inscricao"/>
           : </td><td>
14     <html:text property="inscricao" size="7" maxlength="5"/>
15     <bean:message key="label.cadastro.candidato.nome"/> :
16     <html:text property="nome" size="52" maxlength="100"/>
17     <bean:message key="label.cadastro.candidato.deficiente"/>?
18
19       <html:select property="deficiente">
20         <html:option value=""> </html:option>
21         <html:option value="N">Nao</html:option>
22         <html:option value="S">Sim</html:option>
23       </html:select>
24
25     </td>
26   </tr>
27
28   <!-- Segunda linha de campos -->
29   <tr>
```

```
30      <td> <bean:message key="label.cadastro.candidato.logradouro"
            /> : </td><td>
31      <html:text property="logradouro" size="30" maxlength="100"/> &
            nbsp;
32      <bean:message key="label.cadastro.candidato.complemento"/>&
            nbsp;: 
33      <html:text property="complemento" size="14" maxlength="20"/> &
            nbsp;
34      <bean:message key="label.cadastro.candidato.bairro"/> :&
            nbsp;
35      <html:text property="bairro" size="20" maxlength="40"/> </td>
36    </tr>
37
38    <!-- Terceira linha de campos -->
39    <tr>
40      <td> <bean:message key="label.cadastro.candidato.cep"/> 
            ;: </td><td>
41      <html:text property="cep" size="10" maxlength="9"/> 
42      <bean:message key="label.cadastro.candidato.cidade"/> :&
            nbsp;
43      <html:text property="cidade" size="20" maxlength="30"/> 
44
45         <!-- Lista de Estados Brasileiros -->
46         <logic:present name="listaEstados">
47              <bean:message key="label.cadastro.candidato.uf"/> 
                    ;: 
48              <html:select property="uf">
49                <html:options collection="listaEstados" property="
                      sigla" labelProperty="sigla"/>   
50              </html:select>
51        </logic:present>
52
53      <bean:message key="label.cadastro.candidato.telefone"/> :&
            nbsp;
54      <html:text property="telefone" size="26" maxlength="30"/></td>
55    </tr>
56    <tr>
57      <td><bean:message key="label.cadastro.candidato.email"/> 
            ;:  </td>
58      <td><html:text property="email" size="30" maxlength="40"/></td
            >
59    </tr>
60
61    <!-- Select com os concursos para selecao -->
62    <tr>
63         <logic:present name="listaConcursos">
64            <td> Concurso: </td><td>
65              <html:select property="concurso">
```

```
66              <html:options collection="listaConcursos" property="
                    id" labelProperty="descricao"/>
67            </html:select>
68          </td>
69      </logic:present>
70  </tr>
71
72  <tr><td> </td></tr>
73
74  <!-- Pesquisa de Candidatos -->
75  <tr><td align="center" colspan="6">
76    <logic:present name="listaConteudoPesquisa">
77      <display:table name="sessionScope.listaConteudoPesquisa"
            class="classTabela" pagesize="20"
78      requestURI="/cadastroCandidato.do?evento=X" export="true">
79        <display:column class="linhaTabela" title="Inscricaoo"
              property="inscricaoFormatada" sortable="true" style="
              text-align:center; color:#336699;"/>
80        <display:column class="linhaTabela" title="Nome" property=
              "nome" sortable="true"/>
81        <display:column class="linhaTabela" title="Cidade"
              property="cidade"/>
82        <display:column class="linhaTabela" title="UF" property="
              UF"/>
83        <display:column class="linhaTabela" title="CEP" property="
              CEP"/>
84        <display:column class="linhaTabela" title="Editar" href="
              cadastroCandidato.do?evento=Editar" paramId="
              identificador" paramProperty="id" style="text-align:
              center">
85          <img src="<%=request.getContextPath()%>/resources/
                images/alterar.gif" border='0' />
86        </display:column>
87      </display:table>
88    </logic:present>
89
90  </td></tr>
91  </table>
```

**Listing A.1:** Página JSP para criação e manutenção de um candidato

```
1    <!-- Definition para Cadastro Candidato -->
2     <definition name="def.cadastro.candidato" extends="def.pagina.
          ancestral.intranet">
3         <put name="titulo" value="label.cadastro.candidato.titulo"
              />
4         <put name="corpo" value="def.cadastro.candidato.corpo" />
5     </definition>
```

```
6       <definition name="def.cadastro.candidato.corpo" extends="def.
            corpo.ancestral.intranet">
7           <put name="tituloCorpo" value="label.cadastro.candidato.
                titulo"/>
8           <putList name="lista" >
9             <add value="/pages/Candidato.jsp"/>
10          </putList>
11      </definition>
```

**Listing A.2:** Configuração do Tiles para a página de manutenção criada

```
1    package gov.trt.mg.cj.pojo;
2  import gov.trt.mg.comum.pojo.Ancestral;
3  import java.io.Serializable;
4  import java.text.DecimalFormat;
5
6
7  /**
8   * Classe para a entidade Candidato
9   *
10  *
11  * @hibernate.class table="CANDIDATO"
12  */
13 public class Candidato extends Ancestral  {
14
15   /* Atributos */
16   Long inscricao;
17   String nome;
18   String logradouro;
19   String complemento;
20   String bairro;
21   String CEP;
22   String cidade;
23   String UF;
24   String telefone;
25   String email;
26   String deficiente;
27   String deferido;
28   Concurso concurso;
29
30   public Candidato() {
31   }
32
33   public Serializable getId() {
34       return super.getId();
35   }
36
37   public void setInscricao(Long inscricao) {
38     this.inscricao = inscricao;
```

```
39    }
40
41
42    public Long getInscricao() {
43      return inscricao;
44    }
45
46    public String getInscricaoFormatada () {
47      DecimalFormat formato = new DecimalFormat("0000");
48      return formato.format(inscricao);
49    }
50
51
52    public void setNome(String nome) {
53      this.nome = nome;
54    }
55
56
57    public String getNome() {
58      if (nome != null)
59        return nome.trim();
60      else
61        return nome;
62    }
63
64
65    public void setLogradouro(String logradouro) {
66      this.logradouro = logradouro;
67    }
68
69
70    public String getLogradouro() {
71      return logradouro;
72    }
73
74
75    public void setCEP(String CEP) {
76      this.CEP = CEP;
77    }
78
79
80    public String getCEP() {
81      return CEP;
82    }
83
84
85    public void setCidade(String cidade) {
86      this.cidade = cidade;
```

```
87    }
88
89
90    public String getCidade() {
91      return cidade;
92    }
93
94
95    public void setUF(String UF) {
96      this.UF = UF;
97    }
98
99
100   public String getUF() {
101     return UF;
102   }
103
104
105   public void setTelefone(String telefone) {
106     this.telefone = telefone;
107   }
108
109
110   public String getTelefone() {
111     return telefone;
112   }
113
114
115   public void setConcurso(Concurso concurso) {
116     this.concurso = concurso;
117   }
118
119
120   public Concurso getConcurso() {
121     return concurso;
122   }
123
124
125   public void setComplemento(String complemento) {
126     this.complemento = complemento;
127   }
128
129
130   public String getComplemento() {
131     return complemento;
132   }
133
134
```

```
135    public void setBairro(String bairro) {
136      this.bairro = bairro;
137    }
138
139
140    public String getBairro() {
141      return bairro;
142    }
143
144
145    public void setEmail(String email) {
146      this.email = email;
147    }
148
149
150    public String getEmail() {
151      return email;
152    }
153
154
155    public void setDeficiente(String deficiente) {
156      this.deficiente = deficiente;
157    }
158
159
160    public String getDeficiente() {
161      return deficiente;
162    }
163
164    public boolean isCandidatoDeficiente() {
165
166      if (deficiente != null && deficiente.equals("S"))
167        return true;
168      else
169        return false;
170    }
171
172
173      public String getDeferido() {
174          return deferido;
175      }
176
177      public void setDeferido(String deferido) {
178          this.deferido = deferido;
179      }
180  }
```

**Listing A.3:** Classe de domínio Candidato

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >

<hibernate-mapping>
    <class
        name="gov.trt.mg.cj.pojo.Candidato"
        table="CANDIDATO"
        dynamic-update="false"
        dynamic-insert="false"
    >

        <id
            name="id"
            column="ID_CANDIDATO"
            type="java.lang.Long"
            unsaved-value="null"
        >
            <generator class="sequence">
                <param name="sequence">sq_CANDIDATO</param>
            </generator>
         </id>

        <version
            name="versao"
            type="long"
            column="VERSAO"
            access="property"
            unsaved-value="null"
        />

        <property
            name="inscricao"
            type="java.lang.Long"
            update="true"
            insert="true"
            access="property"
            column="INSCRICAO"
            not-null="true"
            unique="true"
        />

        <property
            name="nome"
            type="string"
            update="true"
            insert="true"
```

```
49              access="property"
50              column="NOME"
51              length="100"
52              not-null="true"
53              unique="false"
54          />
55
56          <property
57              name="logradouro"
58              type="string"
59              update="true"
60              insert="true"
61              access="property"
62              column="LOGRADOURO"
63              length="100"
64              not-null="false"
65              unique="false"
66          />
67
68          <property
69              name="complemento"
70              type="string"
71              update="true"
72              insert="true"
73              access="property"
74              column="COMPLEMENTO"
75              length="20"
76              not-null="false"
77              unique="false"
78          />
79
80          <property
81              name="bairro"
82              type="string"
83              update="true"
84              insert="true"
85              access="property"
86              column="BAIRRO"
87              length="40"
88              not-null="false"
89              unique="false"
90          />
91
92          <property
93              name="CEP"
94              type="string"
95              update="true"
96              insert="true"
```

```
97              access="property"
98              column="CEP"
99              length="9"
100             not-null="true"
101             unique="false"
102         />
103
104         <property
105             name="cidade"
106             type="string"
107             update="true"
108             insert="true"
109             access="property"
110             column="CIDADE"
111             length="30"
112             not-null="false"
113             unique="false"
114         />
115
116         <property
117             name="UF"
118             type="string"
119             update="true"
120             insert="true"
121             access="property"
122             column="UF"
123             length="2"
124             not-null="true"
125             unique="false"
126         />
127
128         <property
129             name="telefone"
130             type="string"
131             update="true"
132             insert="true"
133             access="property"
134             column="TELEFONE"
135             length="30"
136             not-null="false"
137             unique="false"
138         />
139
140         <property
141             name="email"
142             type="string"
143             update="true"
144             insert="true"
```

```
145             access="property"
146             column="EMAIL"
147             length="40"
148             not-null="false"
149             unique="false"
150         />
151
152         <property
153             name="deficiente"
154             type="string"
155             update="true"
156             insert="true"
157             access="property"
158             column="DEFICIENTE"
159             length="1"
160             not-null="false"
161             unique="false"
162         />
163
164         <property
165             name="deferido"
166             type="string"
167             update="true"
168             insert="true"
169             access="property"
170             column="DEFERIDO"
171             length="1"
172             not-null="false"
173             unique="false"
174         />
175
176         <many-to-one
177             name="concurso"
178             class="gov.trt.mg.cj.pojo.Concurso"
179             cascade="none"
180             outer-join="false"
181             update="true"
182             insert="true"
183             access="property"
184             column="ID_CONCURSO"
185             not-null="true"
186         />
187
188         <property
189             name="dataUltimaAtualizacao"
190             type="timestamp"
191             update="true"
192             insert="true"
```

```
193             access="property"
194             column="DT_ULT_ATUALIZACAO"
195         />
196
197         <!--
198             To add non XDoclet property mappings, create a file
                named
199                 hibernate-properties-Processo.xml
200             containing the additional properties and place it in
                your merge dir.
201         -->
202
203     </class>
204
205 </hibernate-mapping>
```

**Listing A.4:** Mapeamento objeto-relacional da classe de domínio

```
1  package gov.trt.mg.cj.dao;
2
3  import gov.trt.mg.cj.pojo.Candidato;
4  import gov.trt.mg.cj.pojo.Concurso;
5  import gov.trt.mg.comum.model.dao.DataAccessException;
6  import gov.trt.mg.comum.model.dao.HibernateImplDAO;
7  import gov.trt.mg.comum.pojo.Persistent;
8
9  import java.sql.Connection;
10 import java.sql.ResultSet;
11 import java.sql.Statement;
12
13 import java.util.List;
14
15 import org.hibernate.Criteria;
16 import org.hibernate.HibernateException;
17 import org.hibernate.Session;
18 import org.hibernate.criterion.Example;
19 import org.hibernate.criterion.Expression;
20 import org.hibernate.criterion.MatchMode;
21 import org.hibernate.criterion.Order;
22
23
24 public final class HibernateCandidatoDAO extends HibernateImplDAO
25   implements ICandidatoDAO {
26
27   public HibernateCandidatoDAO() {
28   }
29
30     public List findByExample(Persistent obj, String orderProperty
         , boolean asc) throws DataAccessException {
```

```
31          log.info("<HibernateCandidatoDAO> - entrou em
                findByExample");
32          Session sess = null;
33          try {
34              obj.setDataUltimaAtualizacao(null);
35              obj.setVersao(null);
36              sess = openSession();
37
38              Candidato candidato = (Candidato) obj;
39
40              Example example = Example.create(obj).ignoreCase()
41                .enableLike(MatchMode.ANYWHERE)
42                .excludeProperty("complemento")
43                .excludeProperty("telefone");
44
45              if (candidato.getEmail() == null || candidato.getEmail
                    ().equals("")) {
46                example.excludeProperty("email");
47              }
48
49              Class clazz = obj.getClass();
50              Criteria criteria = null;
51              if (asc)
52               criteria = sess.createCriteria(clazz).add(example).
                    addOrder(Order.asc(orderProperty));
53              else
54                criteria = sess.createCriteria(clazz).add(example).
                     addOrder(Order.desc(orderProperty));
55
56              criteria.add(Expression.eq("concurso", candidato.
                   getConcurso()));
57
58              criteria = antesPesquisarMontaCriteria(criteria, obj);
59
60              List result = criteria.list();
61
62              return result;
63
64          } catch(HibernateException e) {
65              log.error("<HibernateCandidatoDAO> - erro ao pesquisar
                     objeto no banco",e);
66              throw new DataAccessException("Erro ao pesuisar objeto
                     no banco",e);
67          } catch (Exception e) {
68              log.error("<HibernateCandidatoDAO> - erro ao pesquisar
                     objeto no banco",e);
69              e.printStackTrace();
```

```
70              throw new DataAccessException("Erro ao pesuisar objeto
                    no banco",e);
71          }
72
73          finally {
74              closeSession(sess);
75          }
76      }
77
78      public int obterCandidatosConcurso(String idConcurso) throws
            DataAccessException {
79          log.info("<HibernateCandidatoDAO> - entrou em
                obterCandidatosConcurso");
80          Connection con = null;
81          Statement vStatement = null;
82          ResultSet vResultSet = null;
83          Session sess = null;
84
85          try {
86
87              sess = openSession();
88              con = sess.connection();
89              vStatement = (Statement)con.createStatement();
90
91              String vSQL = "select count(*) from CANDIDATO where
                    ID_CONCURSO = " + idConcurso;
92              vResultSet = vStatement.executeQuery(vSQL);
93
94              String vContagem = null;
95              if (vResultSet.next()) {
96                vContagem = vResultSet.getString(1);
97                return Integer.parseInt(vContagem);
98              }
99
100             return 0;
101
102         } catch (Exception e) {
103             log.error("<HibernateCandidatoDAO> - erro ao contar
                    candidatos de um concurso",  e);
104             throw new DataAccessException("Erro ao contar
                    candidatos de um concurso", e);
105         } finally {
106             closeSession(sess);
107         }
108     }
109
110     /**
```

```
111       * Retorna a lista com candidatos para um determinado concurso
              . Utiliza o metodo
112       * findbyExample ja existente.
113       *
114       * @throws gov.trt.mg.comum.model.dao.DataAccessException
115       * @return
116       * @param idConcurso
117       */
118      public List findCandidatosConcurso(String idConcurso) throws
              DataAccessException {
119        log.info("<HibernateCandidatoDAO> - entrou em
               findCandidatosConcurso");
120
121        Candidato candidato = new Candidato();
122        Concurso concurso = (Concurso)findByPrimaryKey(Concurso.
               class, new Long(idConcurso));
123        candidato.setConcurso(concurso);
124        return findByExample(candidato, "id", true);
125      }
126
127  }
```

**Listing A.5:** Classe de interação com o banco de dados

```
1   <mapping resource="./gov/trt/mg/cj/pojo/Concurso.hbm.xml" />
```

**Listing A.6:** Entrada no arquivo de configuração do Hibernate

```
1  package gov.trt.mg.cj.controller.action;
2
3  import gov.trt.mg.cj.facade.CadastroCandidatoFacade;
4  import gov.trt.mg.cj.facade.CadastroConcursoFacade;
5  import gov.trt.mg.cj.util.Util;
6  import gov.trt.mg.comum.controller.ControllerLayerException;
7  import gov.trt.mg.comum.controller.action.CRUDAction;
8  import gov.trt.mg.comum.facade.FacadeException;
9  import gov.trt.mg.comum.pojo.Persistent;
10 import gov.trt.mg.comum.util.Constantes;
11
12 import java.util.ArrayList;
13 import java.util.HashMap;
14 import java.util.List;
15 import java.util.Map;
16
17 import javax.servlet.http.HttpServletRequest;
18 import javax.servlet.http.HttpServletResponse;
19
20 import org.apache.struts.action.ActionForm;
21 import org.apache.struts.action.ActionForward;
22 import org.apache.struts.action.ActionMapping;
```

```
23 import org.apache.struts.action.DynaActionForm;
24
25
26 /**
27  * DOCUMENT ME!
28  *
29  * @author Tiago Falqueto
30  */
31 public final class CandidatoAction extends CRUDAction {
32     //~ Methods
          ------------------------------------------------------------------------


33
34     /**
35      * DOCUMENT ME!
36      *
37      * @return
38      */
39     protected Map mapeiaEventosEspecificos() {
40         log.info("<CRUDAction> - entrou em
               mapeiaEventosEspecificos");
41
42         Map eventos = new HashMap();
43         eventos.put("evento.gravar", "gravar");
44         eventos.put("evento.editar", "editar");
45         eventos.put("evento.pesquisar", "pesquisar");
46         eventos.put("evento.excluir", "excluir");
47         eventos.put("evento.limpar", "limpar");
48         eventos.put("evento.pesquisar.todos", "pesquisarTodos");
49         eventos.put("evento.pesqBack", "pesqBack");
50         eventos.put("evento.pesqNext", "pesqNext");
51
52         return eventos;
53     }
54
55     /**
56      * DOCUMENT ME!
57      *
58      * @param mapping
59      * @param form
60      * @param request
61      * @param response
62      * @param obj
63      *
64      * @throws ControllerLayerException
65      */
66     protected void montaGridPesquisa(ActionMapping mapping,
           ActionForm form,
```

```
67          HttpServletRequest request, HttpServletResponse response,
                Persistent obj)
68          throws ControllerLayerException {
69          List list = new ArrayList();
70
71          /*Recupera lista completa de todos os TituloEmenta */
72          try {
73              CadastroCandidatoFacade facade = new
                    CadastroCandidatoFacade();
74
75              // Passa o objeto e a coluna do banco que vai ser
                    usada para ordenacao
76              list = facade.findByExample(obj, "nome", true);
77
78              // Se a lista vazia, retira o Grid.
79              if ((list == null) || list.isEmpty()) {
80                  request.getSession().setAttribute("
                        listaConteudoPesquisa", null);
81                  throw mensagemErro(request, "erro.pesquisa.vazia",
                         null, null);
82              }
83
84              request.getSession().setAttribute("
                    listaConteudoPesquisa", list);
85
86          } catch (FacadeException e) {
87              log.error("<ManterTituloEmentaAction> - erro ao criar
                    lista de TituloEmenta",
88                  e);
89          }
90      }
91
92      /**
93       * Ajusta Botoes de navegacao
94       *
95       * @param request
96       * @param modo
97       */
98      protected void ajustaBotoes(HttpServletRequest request, String
             modo) {
99          log.info("<CRUDAction> - entrou em ajusta botoes");
100
101         request.getSession().setAttribute("modoImpressao", new
                Boolean(false));
102
103         if (modo.equals(Constantes.MODO_INCLUSAO)) {
104             request.setAttribute(Constantes.BOTAO_GRAVAR,
                    Constantes.EXIBIR);
```

```
105             request.setAttribute(Constantes.BOTAO_PESQUISAR,
                    Constantes.EXIBIR);
106             request.setAttribute(Constantes.BOTAO_EXCLUIR,
                    Constantes.NAO_EXIBIR);
107             request.setAttribute(Constantes.BOTAO_LIMPAR,
                    Constantes.EXIBIR);
108         } else if (modo.equals(Constantes.MODO_EDICAO)) {
109             request.setAttribute(Constantes.BOTAO_GRAVAR,
                    Constantes.EXIBIR);
110             request.setAttribute(Constantes.BOTAO_PESQUISAR,
                    Constantes.NAO_EXIBIR);
111             request.setAttribute(Constantes.BOTAO_EXCLUIR,
                    Constantes.EXIBIR);
112             request.setAttribute(Constantes.BOTAO_LIMPAR,
                    Constantes.NAO_EXIBIR);
113         } else if (modo.equals(Constantes.MODO_CONSULTA)) {
114             request.setAttribute(Constantes.BOTAO_GRAVAR,
                    Constantes.NAO_EXIBIR);
115             request.setAttribute(Constantes.BOTAO_PESQUISAR,
                    Constantes.EXIBIR);
116             request.setAttribute(Constantes.BOTAO_EXCLUIR,
                    Constantes.NAO_EXIBIR);
117             request.setAttribute(Constantes.BOTAO_LIMPAR,
                    Constantes.EXIBIR);
118         } else if (modo.equals(Constantes.MODO_VISUALIZACAO)) {
119             request.setAttribute(Constantes.BOTAO_GRAVAR,
                    Constantes.EXIBIR);
120             request.setAttribute(Constantes.BOTAO_PESQUISAR,
                    Constantes.NAO_EXIBIR);
121             request.setAttribute(Constantes.BOTAO_EXCLUIR,
                    Constantes.NAO_EXIBIR);
122             request.setAttribute(Constantes.BOTAO_LIMPAR,
                    Constantes.EXIBIR);
123         }
124         log.info("<CRUDAction> - saiu de ajusta botoes");
125     }
126
127     public ActionForward limpar(ActionMapping mapping, ActionForm
            form,
128         HttpServletRequest request, HttpServletResponse response)
129         throws ControllerLayerException {
130         log.info("<CRUDAction> - entrou em limpar");
131
132         CadastroConcursoFacade facade = new CadastroConcursoFacade
                ();
133
134         try {
135           /*Limpa o atributo de grid da sessao*/
```

```
136             request.getSession().removeAttribute("
                    listaCamposPesquisa");
137             request.getSession().removeAttribute("
                    listaConteudoPesquisa");
138
139             List listaConcursos = facade.findAllConcursos();
140             List listaEstados = Util.getListaEstados();
141             request.getSession().setAttribute("listaConcursos",
                    listaConcursos);
142             request.getSession().setAttribute("listaEstados",
                    listaEstados);
143
144             DynaActionForm df = (DynaActionForm) form;
145             df.initialize(mapping);
146             disponibilizaClassesLookup(mapping, form, request,
                    response);
147             montaRequest(mapping, df, request, response, Constantes.
                    MODO_INCLUSAO);
148             request.getSession().removeAttribute("
                    listaConteudoPesquisa");
149
150         }
151         catch (FacadeException e) {
152             log.error("<CandidatoAction> - erro ao recuperar as
                        listas de cep e candidato",
153                     e);
154         }
155
156         return mapping.findForward(Constantes.FW_MESMA_PAGINA);
157     }
158
159
160 }
```

**Listing A.7:** Classe controladora

```
1  package gov.trt.mg.comum.helper;
2
3  import gov.trt.mg.cj.pojo.Candidato;
4  import gov.trt.mg.cj.pojo.Concurso;
5  import gov.trt.mg.cj.util.Constantes;
6  import gov.trt.mg.comum.facade.IFacade;
7  import gov.trt.mg.comum.helper.AncestralHelper;
8  import gov.trt.mg.comum.helper.HelperException;
9  import gov.trt.mg.comum.pojo.Persistent;
10 import org.apache.struts.action.DynaActionForm;
11
12 import java.util.Date;
13
```

```
14
15 /**
16  * DOCUMENT ME!
17  *
18  * @author Tiago Falqueto
19  */
20 public final class CandidatoHelper extends AncestralHelper {
21     //~ Methods
          ------------------------------------------------------------------------


22
23     /**
24      * DOCUMENT ME!
25      *
26      * @param from Formulario com os dados
27      * @param toPersistentClassName Nome da classe de persistencia
28      * @param facade Fachada do caso de uso
29      *
30      * @return tema  Objeto com os dados
31      *
32      * @throws HelperException
33      */
34     public Persistent formToPersistent(DynaActionForm from, String
            toPersistentClassName,
35         IFacade facade) throws HelperException {
36         try {
37             //Class persistentClass = Class.forName(
                   toPersistentClassName);
38             Class persistentClass = Class.forName(
                   toPersistentClassName, true,
39                     this.getClass().getClassLoader());
40             Candidato candidato = (Candidato) persistentClass.
                   newInstance();
41
42             String id = (String) from.get(Constantes.IDENTIFICADOR
                   );
43
44             if ((id != null) && !(id.trim().length() == 0)) {
45                 candidato.setId(new Long(id));
46             }
47
48             // campos especificos
49             String vInscricao = (String)from.get("inscricao");
50             if (vInscricao!=null && !vInscricao.equals("")) {
51               candidato.setInscricao(new Long(vInscricao));
52             }
53             candidato.setNome((String) from.get("nome"));
```

```
54              candidato.setLogradouro((String) from.get("logradouro"
                    ));
55              candidato.setComplemento((String) from.get("
                    complemento"));
56              candidato.setBairro((String) from.get("bairro"));
57              candidato.setCEP((String) from.get("cep"));
58              candidato.setCidade((String) from.get("cidade"));
59              candidato.setUF((String) from.get("uf"));
60              candidato.setTelefone((String) from.get("telefone"));
61              candidato.setEmail((String) from.get("email"));
62              candidato.setDeficiente((String) from.get("deficiente"
                    ));
63
64              String vIdConcurso = (String)from.get("concurso");
65              if (vIdConcurso!=null && !vIdConcurso.equals("")) {
66                Concurso vConcurso = (Concurso)facade.
                      findByPrimaryKey(
67                 Concurso.class, new Long(vIdConcurso));
68                candidato.setConcurso(vConcurso);
69              }
70
71              // campos fixos
72              candidato.setVersao((Long) from.get("versao"));
73              candidato.setDataUltimaAtualizacao((Date) from.get("
                    dataUltimaAtualizacao"));
74
75              return candidato;
76
77          } catch (InstantiationException e) {
78              throw new HelperException("Erro ao instanciar fase do
                    projeto", e);
79          } catch (IllegalAccessException e) {
80              throw new HelperException("Erro ao acessar objeto
                    concurso", e);
81          } catch (ClassNotFoundException e) {
82              throw new HelperException("Classe concurso nao
                    encontrada", e);
83          } catch (Exception e) {
84              throw new HelperException("Erro ao acessar fachada", e
                    );
85          }
86
87      }
88
89      /**
90       * DOCUMENT ME!
91       *
92       * @param from Objeto com os dados
```

```
93        * @param to Form de destino que sera retornado pronto
94        * @param facade Fachada do caso de uso
95        *
96        * @return to formulario pronto
97        *
98        * @throws HelperException
99        */
100       public DynaActionForm persistentToForm(Persistent from,
              DynaActionForm to,
101               IFacade facade) throws HelperException {
102               Candidato candidato = (Candidato) from;
103
104               // campos especificos
105
106               to.set("inscricao", candidato.getInscricao().toString());
107               to.set("nome", candidato.getNome());
108               to.set("deficiente", candidato.getDeficiente());
109               to.set("logradouro", candidato.getLogradouro());
110               to.set("complemento", candidato.getComplemento());
111               to.set("bairro", candidato.getBairro());
112               to.set("cep", candidato.getCEP());
113               to.set("cidade", candidato.getCidade());
114               to.set("uf", candidato.getUF());
115               to.set("telefone", candidato.getTelefone());
116               to.set("email", candidato.getEmail());
117               to.set("concurso", candidato.getConcurso().getId().
                      toString());
118
119               //Padrao
120               to.set("versao", candidato.getVersao());
121               to.set("dataUltimaAtualizacao", candidato.
                      getDataUltimaAtualizacao());
122
123               return to;
124       }
125
126       public void clearForm(DynaActionForm form) throws
              HelperException {
127               form.set("inscricao", null);
128               form.set("nome", null);
129               form.set("deficiente", null);
130               form.set("logradouro", null);
131               form.set("complemento", null);
132               form.set("bairro", null);
133               form.set("cep", null);
134               form.set("cidade", null);
135               form.set("uf", null);
136               form.set("telefone", null);
```

```
137         form.set("email", null);
138         form.set("concurso", null);
139     }
140 }
```

**Listing A.8:** Classe que fará a conversão de dados da JSP para uma classe de negócio e vice-versa. label

```
1     <!-- Form-Bean para pagina de Candidato -->
2         <form-bean name="cadastroCandidatoForm" type="org.apache.
              struts.validator.DynaValidatorActionForm">
3             <form-property name="inscricao" type="java.lang.String
                  "/>
4             <form-property name="nome" type="java.lang.String"/>
5             <form-property name="deficiente" type="java.lang.
                  String"/>
6             <form-property name="logradouro" type="java.lang.
                  String"/>
7             <form-property name="complemento" type="java.lang.
                  String"/>
8             <form-property name="bairro" type="java.lang.String"/>
9             <form-property name="cep" type="java.lang.String"/>
10            <form-property name="cidade" type="java.lang.String"/>
11            <form-property name="uf" type="java.lang.String"/>
12            <form-property name="telefone" type="java.lang.String"
                  />
13            <form-property name="email" type="java.lang.String"/>
14            <form-property name="concurso" type="java.lang.String"
                  />
15            <form-property name="prova" type="java.lang.String"/>
16            <form-property name="tipoListagem" type="java.lang.
                  String"/>
17            <form-property name="identificador" type="java.lang.
                  String"/>
18            <form-property name="modoOperacao" type="java.lang.
                  String" />
19            <form-property name="versao"  type="java.lang.Long"/>
20            <form-property name="dataUltimaAtualizacao"  type="
                  java.util.Date"/>
21        </form-bean>
22
23    <!-- Mapping para Cadastro Candidato -->
24        <action path="/cadastroCandidato"
25            name="cadastroCandidatoForm"
26            input="def.cadastro.candidato"
27            parameter="evento"
28            validate="false"
29            className="gov.trt.mg.comum.controller.
                  JPDActionMapping"
```

```
30              type="gov.trt.mg.cj.controller.action.CandidatoAction"
                    >
31              <set-property property="useCaseId" value="
                    csuCadastroCandidato" />
32              <set-property property="persistentClassName" value="
                    gov.trt.mg.cj.pojo.Candidato" />
33              <forward name="mesmaPagina"    path="def.cadastro.
                    candidato"/>
34          </action>
```

**Listing A.9:** Adição do controlador criado no arquivo de configuração do Struts

# Apêndice B

# Código-fonte Grails

```
1
2  class Candidato {
3      static mapping = {
4           table 'CANDIDATO'
5           id column: 'ID_CANDIDATO', generator: 'sequence' ,params
                :[sequence: 'SQ_CANDIDATO']
6           version false
7           inscricao column:'INSCRICAO'
8           nome column:'NOME'
9           logradouro column:'LOGRADOURO'
10          complemento column:'COMPLEMENTO'
11          bairro column:'BAIRRO'
12          cep column:'CEP'
13          cidade column:'CIDADE'
14          uf column:'UF'
15          telefone column:'TELEFONE'
16          dtUltAtualizacao column:'DT_ULT_ATUALIZACAO'
17          email column:'EMAIL'
18          deficiente column:'DEFICIENTE'
19          deferido column:'DEFERIDO'
20          idConcurso column:'ID_CONCURSO'
21     }
22     java.lang.Integer inscricao
23     java.lang.String nome
24     java.lang.String logradouro
25     java.lang.String complemento
26     java.lang.String bairro
27     java.lang.String cep
28     java.lang.String cidade
29     java.lang.String uf
30     java.lang.String telefone
```

```
31      java.util.Date dtUltAtualizacao
32      java.lang.String email
33      java.lang.String deficiente
34      java.lang.String deferido
35      // Relation
36      Concurso idConcurso
37      SortedSet distribuicoes
38      static constraints = {
39          inscricao(nullable: false)
40          nome(size: 1..100, blank: false)
41          logradouro(size: 0..100)
42          complemento(size: 0..20)
43          bairro(size: 0..40)
44          cep(size: 0..9)
45          cidade(size: 0..30)
46          uf(size: 0..2)
47          telefone(size: 0..30)
48          dtUltAtualizacao(nullable: true)
49          email(size: 0..40)
50          deficiente(size: 0..1)
51          deferido(size: 0..1)
52          idConcurso()
53      }
54      static transients = [ 'concurso' ]
55
56      static belongsTo = Concurso
57
58      static hasMany = [ distribuicoes : Distribuicao]
59
60      def beforeInsert = {
61         dtUltAtualizacao = new Date()
62      }
63      def beforeUpdate = {
64         dtUltAtualizacao = new Date()
65      }
66      String toString() {
67          return "${id}"
68      }
69
70      String getConcurso() {
71          return "${idConcurso.numero}/${idConcurso.ano} - ${
                idConcurso.cargo}"
72      }
73
74  }
```

**Listing B.1:** Classe de domínio gerada manualmente

```
1   import org.codehaus.groovy.grails.plugins.filter.FilterType as
        Type
2   class CandidatoController {
3
4    def index = { redirect(action:list,params:params) }
5
6    // the delete, save and update actions only accept POST
        requests
7    def allowedMethods = [delete:'POST', save:'POST', update:'POST
        ']
8
9    def list = {
10       if(!params.max) params.max = 10
11       [ candidatoList: Candidato.list( params ) ]
12   }
13
14   def show = {
15       def candidato = Candidato.get( params.id )
16
17       if(!candidato) {
18           flash.message = "Candidato not found with id ${params.
                id}"
19           redirect(action:list)
20       }
21       else { return [ candidato : candidato ] }
22   }
23
24   def delete = {
25       def candidato = Candidato.get( params.id )
26       if(candidato) {
27           candidato.delete()
28           flash.message = "Candidato ${params.id} deleted"
29           redirect(action:list)
30       }
31       else {
32           flash.message = "Candidato not found with id ${params.
                id}"
33           redirect(action:list)
34       }
35   }
36
37   def edit = {
38       def candidato = Candidato.get( params.id )
39
40       if(!candidato) {
41           flash.message = "Candidato not found with id ${params.
                id}"
42           redirect(action:list)
```

```
43          }
44          else {
45              return [ candidato : candidato ]
46          }
47      }
48
49      def update = {
50          def candidato = Candidato.get( params.id )
51          if(candidato) {
52              candidato.properties = params
53              if(!candidato.hasErrors() && candidato.save()) {
54                  flash.message = "Candidato ${params.id} updated"
55                  redirect(action:show,id:candidato.id)
56              }
57              else {
58                  render(view:'edit',model:[candidato:candidato])
59              }
60          }
61          else {
62              flash.message = "Candidato not found with id ${params.
                    id}"
63              redirect(action:edit,id:params.id)
64          }
65      }
66
67      def create = {
68          def candidato = new Candidato()
69          candidato.properties = params
70          return ['candidato':candidato]
71      }
72
73      def save = {
74          def candidato = new Candidato(params)
75          if(!candidato.hasErrors() && candidato.save()) {
76              flash.message = "Candidato ${candidato.id} created"
77              redirect(action:show,id:candidato.id)
78          }
79          else {
80              render(view:'create',model:[candidato:candidato])
81          }
82      }
83  }
```

**Listing B.2:** Classe controladora gerada automaticamente

```
1
2
3   <html>
4       <head>
```

```
5           <meta http-equiv="Content-Type" content="text/html;
                charset=UTF-8"/>
6           <meta name="layout" content="main" />
7           <title>Create Candidato</title>
8       </head>
9       <body>
10          <div class="nav">
11              <span class="menuButton"><a class="home" href="${
                    createLinkTo(dir:'')}">Home</a></span>
12              <span class="menuButton"><g:link class="list" action="
                    list">Candidato List</g:link></span>
13          </div>
14          <div class="body">
15              <h1>Create Candidato</h1>
16              <g:if test="${flash.message}">
17              <div class="message">${flash.message}</div>
18              </g:if>
19              <g:hasErrors bean="${candidato}">
20              <div class="errors">
21                  <g:renderErrors bean="${candidato}" as="list" />
22              </div>
23              </g:hasErrors>
24              <g:form action="save" method="post" >
25                  <div class="dialog">
26                      <table>
27                          <tbody>
28
29                              <tr class="prop">
30                                  <td valign="top" class="name">
31                                      <label for="inscricao">
                                            Inscricao:</label>
32                                  </td>
33                                  <td valign="top" class="value ${
                                        hasErrors(bean:candidato,field
                                        :'inscricao','errors')}">
34                                      <input type="text" id="
                                            inscricao" name="inscricao
                                            " value="${fieldValue(bean
                                            :candidato,field:'
                                            inscricao')}" />
35                                  </td>
36                              </tr>
37
38                              <tr class="prop">
39                                  <td valign="top" class="name">
40                                      <label for="nome">Nome:</label
                                            >
41                                  </td>
```

```
42                          <td valign="top" class="value ${
                                hasErrors(bean:candidato,field
                                :'nome','errors')}">
43                              <input type="text" maxlength="
                                    100" id="nome" name="nome"
                                     value="${fieldValue(bean:
                                    candidato,field:'nome')}"
                                    />
44                          </td>
45                      </tr>
46
47                      <tr class="prop">
48                          <td valign="top" class="name">
49                              <label for="logradouro">
                                    Logradouro:</label>
50                          </td>
51                          <td valign="top" class="value ${
                                hasErrors(bean:candidato,field
                                :'logradouro','errors')}">
52                              <input type="text" maxlength="
                                    100" id="logradouro" name=
                                    "logradouro" value="${
                                    fieldValue(bean:candidato,
                                    field:'logradouro')}"/>
53                          </td>
54                      </tr>
55
56                      <tr class="prop">
57                          <td valign="top" class="name">
58                              <label for="complemento">
                                    Complemento:</label>
59                          </td>
60                          <td valign="top" class="value ${
                                hasErrors(bean:candidato,field
                                :'complemento','errors')}">
61                              <input type="text" maxlength="
                                    20" id="complemento" name=
                                    "complemento" value="${
                                    fieldValue(bean:candidato,
                                    field:'complemento')}"/>
62                          </td>
63                      </tr>
64
65                      <tr class="prop">
66                          <td valign="top" class="name">
67                              <label for="bairro">Bairro:</
                                    label>
68                          </td>
```

```
69                          <td valign="top" class="value ${
                                hasErrors(bean:candidato,field
                                :'bairro','errors')}">
70                              <input type="text" maxlength="
                                    40" id="bairro" name="
                                    bairro" value="${
                                    fieldValue(bean:candidato,
                                    field:'bairro')}"/>
71                          </td>
72                      </tr>
73
74                      <tr class="prop">
75                          <td valign="top" class="name">
76                              <label for="cep">Cep:</label>
77                          </td>
78                          <td valign="top" class="value ${
                                hasErrors(bean:candidato,field
                                :'cep','errors')}">
79                              <input type="text" maxlength="
                                    9" id="cep" name="cep"
                                    value="${fieldValue(bean:
                                    candidato,field:'cep')}"/>
80                          </td>
81                      </tr>
82
83                      <tr class="prop">
84                          <td valign="top" class="name">
85                              <label for="cidade">Cidade:</
                                    label>
86                          </td>
87                          <td valign="top" class="value ${
                                hasErrors(bean:candidato,field
                                :'cidade','errors')}">
88                              <input type="text" maxlength="
                                    30" id="cidade" name="
                                    cidade" value="${
                                    fieldValue(bean:candidato,
                                    field:'cidade')}"/>
89                          </td>
90                      </tr>
91
92                      <tr class="prop">
93                          <td valign="top" class="name">
94                              <label for="uf">Uf:</label>
95                          </td>
96                          <td valign="top" class="value ${
                                hasErrors(bean:candidato,field
                                :'uf','errors')}">
```

```
97                          <input type="text" maxlength="
                                2" id="uf" name="uf" value
                                ="${fieldValue(bean:
                                candidato,field:'uf')}"/>
98                      </td>
99                  </tr>
100
101                 <tr class="prop">
102                     <td valign="top" class="name">
103                         <label for="telefone">Telefone
                                :</label>
104                     </td>
105                     <td valign="top" class="value ${
                            hasErrors(bean:candidato,field
                            :'telefone','errors')}">
106                         <input type="text" maxlength="
                                30" id="telefone" name="
                                telefone" value="${
                                fieldValue(bean:candidato,
                                field:'telefone')}"/>
107                     </td>
108                 </tr>
109
110                 <tr class="prop">
111                     <td valign="top" class="name">
112                         <label for="dtUltAtualizacao">
                                Dt Ult Atualizacao:</label
                                >
113                     </td>
114                     <td valign="top" class="value ${
                            hasErrors(bean:candidato,field
                            :'dtUltAtualizacao','errors')}
                            ">
115                         <g:datePicker name="
                                dtUltAtualizacao" value="$
                                {candidato?.
                                dtUltAtualizacao}"
                                noSelection="['':'']"></g:
                                datePicker>
116                     </td>
117                 </tr>
118
119                 <tr class="prop">
120                     <td valign="top" class="name">
121                         <label for="email">Email:</
                                label>
122                     </td>
```

```
123                     <td valign="top" class="value ${
                            hasErrors(bean:candidato,field
                            :'email','errors')}">
124                         <input type="text" maxlength="
                                40" id="email" name="email
                                " value="${fieldValue(bean
                                :candidato,field:'email')}
                                "/>
125                     </td>
126                 </tr>
127
128                 <tr class="prop">
129                     <td valign="top" class="name">
130                         <label for="deficiente">
                                Deficiente:</label>
131                     </td>
132                     <td valign="top" class="value ${
                            hasErrors(bean:candidato,field
                            :'deficiente','errors')}">
133                         <input type="text" maxlength="
                                1" id="deficiente" name="
                                deficiente" value="${
                                fieldValue(bean:candidato,
                                field:'deficiente')}"/>
134                     </td>
135                 </tr>
136
137                 <tr class="prop">
138                     <td valign="top" class="name">
139                         <label for="deferido">Deferido
                                :</label>
140                     </td>
141                     <td valign="top" class="value ${
                            hasErrors(bean:candidato,field
                            :'deferido','errors')}">
142                         <input type="text" maxlength="
                                1" id="deferido" name="
                                deferido" value="${
                                fieldValue(bean:candidato,
                                field:'deferido')}"/>
143                     </td>
144                 </tr>
145
146                 <tr class="prop">
147                     <td valign="top" class="name">
148                         <label for="idConcurso">Id
                                Concurso:</label>
149                     </td>
```

```
150                                     <td valign="top" class="value ${
                                            hasErrors(bean:candidato,field
                                            :'idConcurso','errors')}">
151                                         <g:select optionKey="id" from=
                                                "${Concurso.list()}" name=
                                                "idConcurso.id" value="${
                                                candidato?.idConcurso?.id}
                                                " ></g:select>
152                                     </td>
153                                 </tr>
154
155                                 <tr class="prop">
156                                     <td valign="top" class="name">
157                                         <label for="distribuicoes">
                                                Distribuicoes:</label>
158                                     </td>
159                                     <td valign="top" class="value ${
                                            hasErrors(bean:candidato,field
                                            :'distribuicoes','errors')}">
160
161 <ul>
162 <g:each var="d" in="${candidato?.distribuicoes?}">
163     <li><g:link controller="distribuicao" action="show" id="${d.id
          }">${d?.encodeAsHTML()}</g:link></li>
164 </g:each>
165 </ul>
166 <g:link controller="distribuicao" params="['candidato.id':
        candidato?.id]" action="create">Add Distribuicao</g:link>
167
168                                     </td>
169                                 </tr>
170
171                             </tbody>
172                         </table>
173                     </div>
174                     <div class="buttons">
175                         <span class="button"><input class="save" type=
                                "submit" value="Create" /></span>
176                     </div>
177                 </g:form>
178             </div>
179         </body>
180 </html>
```

**Listing B.3:** Página para criação de um candidato gerada automticamente

```
1
2
3 <html>
```

```
4       <head>
5           <meta http-equiv="Content-Type" content="text/html;
                charset=UTF-8"/>
6           <meta name="layout" content="main" />
7           <title>Edit Candidato</title>
8       </head>
9       <body>
10          <div class="nav">
11              <span class="menuButton"><a class="home" href="${
                    createLinkTo(dir:'')}">Home</a></span>
12              <span class="menuButton"><g:link class="list" action="
                    list">Candidato List</g:link></span>
13              <span class="menuButton"><g:link class="create" action
                    ="create">New Candidato</g:link></span>
14          </div>
15          <div class="body">
16              <h1>Edit Candidato</h1>
17              <g:if test="${flash.message}">
18              <div class="message">${flash.message}</div>
19              </g:if>
20              <g:hasErrors bean="${candidato}">
21              <div class="errors">
22                  <g:renderErrors bean="${candidato}" as="list" />
23              </div>
24              </g:hasErrors>
25              <g:form method="post" >
26                  <input type="hidden" name="id" value="${candidato
                        ?.id}" />
27                  <div class="dialog">
28                      <table>
29                          <tbody>
30
31                              <tr class="prop">
32                                  <td valign="top" class="name">
33                                      <label for="inscricao">
                                            Inscricao:</label>
34                                  </td>
35                                  <td valign="top" class="value ${
                                        hasErrors(bean:candidato,field
                                        :'inscricao','errors')}">
36                                      <input type="text" id="
                                            inscricao" name="inscricao
                                            " value="${fieldValue(bean
                                            :candidato,field:'
                                            inscricao')}" />
37                                  </td>
38                              </tr>
39
```

```
40                          <tr class="prop">
41                              <td valign="top" class="name">
42                                  <label for="nome">Nome:</label
                                        >
43                              </td>
44                              <td valign="top" class="value ${
                                    hasErrors(bean:candidato,field
                                    :'nome','errors')}">
45                                  <input type="text" maxlength="
                                        100" id="nome" name="nome"
                                         value="${fieldValue(bean:
                                        candidato,field:'nome')}"
                                        />
46                              </td>
47                          </tr>
48
49                          <tr class="prop">
50                              <td valign="top" class="name">
51                                  <label for="logradouro">
                                        Logradouro:</label>
52                              </td>
53                              <td valign="top" class="value ${
                                    hasErrors(bean:candidato,field
                                    :'logradouro','errors')}">
54                                  <input type="text" maxlength="
                                        100" id="logradouro" name=
                                        "logradouro" value="${
                                        fieldValue(bean:candidato,
                                        field:'logradouro')}"/>
55                              </td>
56                          </tr>
57
58                          <tr class="prop">
59                              <td valign="top" class="name">
60                                  <label for="complemento">
                                        Complemento:</label>
61                              </td>
62                              <td valign="top" class="value ${
                                    hasErrors(bean:candidato,field
                                    :'complemento','errors')}">
63                                  <input type="text" maxlength="
                                        20" id="complemento" name=
                                        "complemento" value="${
                                        fieldValue(bean:candidato,
                                        field:'complemento')}"/>
64                              </td>
65                          </tr>
66
```

```
67                          <tr class="prop">
68                              <td valign="top" class="name">
69                                  <label for="bairro">Bairro:</
                                        label>
70                              </td>
71                              <td valign="top" class="value ${
                                    hasErrors(bean:candidato,field
                                    :'bairro','errors')}">
72                                  <input type="text" maxlength="
                                        40" id="bairro" name="
                                        bairro" value="${
                                        fieldValue(bean:candidato,
                                        field:'bairro')}"/>
73                              </td>
74                          </tr>
75
76                          <tr class="prop">
77                              <td valign="top" class="name">
78                                  <label for="cep">Cep:</label>
79                              </td>
80                              <td valign="top" class="value ${
                                    hasErrors(bean:candidato,field
                                    :'cep','errors')}">
81                                  <input type="text" maxlength="
                                        9" id="cep" name="cep"
                                        value="${fieldValue(bean:
                                        candidato,field:'cep')}"/>
82                              </td>
83                          </tr>
84
85                          <tr class="prop">
86                              <td valign="top" class="name">
87                                  <label for="cidade">Cidade:</
                                        label>
88                              </td>
89                              <td valign="top" class="value ${
                                    hasErrors(bean:candidato,field
                                    :'cidade','errors')}">
90                                  <input type="text" maxlength="
                                        30" id="cidade" name="
                                        cidade" value="${
                                        fieldValue(bean:candidato,
                                        field:'cidade')}"/>
91                              </td>
92                          </tr>
93
94                          <tr class="prop">
95                              <td valign="top" class="name">
```

```
96                          <label for="uf">Uf:</label>
97                      </td>
98                      <td valign="top" class="value ${
                            hasErrors(bean:candidato,field
                            :'uf','errors')}">
99                          <input type="text" maxlength="
                                2" id="uf" name="uf" value
                                ="${fieldValue(bean:
                                candidato,field:'uf')}"/>
100                     </td>
101                 </tr>
102
103                 <tr class="prop">
104                     <td valign="top" class="name">
105                         <label for="telefone">Telefone
                                :</label>
106                     </td>
107                     <td valign="top" class="value ${
                            hasErrors(bean:candidato,field
                            :'telefone','errors')}">
108                         <input type="text" maxlength="
                                30" id="telefone" name="
                                telefone" value="${
                                fieldValue(bean:candidato,
                                field:'telefone')}"/>
109                     </td>
110                 </tr>
111
112                 <tr class="prop">
113                     <td valign="top" class="name">
114                         <label for="dtUltAtualizacao">
                                Dt Ult Atualizacao:</label
                                >
115                     </td>
116                     <td valign="top" class="value ${
                            hasErrors(bean:candidato,field
                            :'dtUltAtualizacao','errors')}
                            ">
117                         <g:datePicker name="
                                dtUltAtualizacao" value="$
                                {candidato?.
                                dtUltAtualizacao}"
                                noSelection="['':'']"></g:
                                datePicker>
118                     </td>
119                 </tr>
120
121                 <tr class="prop">
```

```
122                         <td valign="top" class="name">
123                             <label for="email">Email:</
                                    label>
124                         </td>
125                         <td valign="top" class="value ${
                                hasErrors(bean:candidato,field
                                :'email','errors')}">
126                             <input type="text" maxlength="
                                    40" id="email" name="email
                                    " value="${fieldValue(bean
                                    :candidato,field:'email')}
                                    "/>
127                         </td>
128                     </tr>
129
130                     <tr class="prop">
131                         <td valign="top" class="name">
132                             <label for="deficiente">
                                    Deficiente:</label>
133                         </td>
134                         <td valign="top" class="value ${
                                hasErrors(bean:candidato,field
                                :'deficiente','errors')}">
135                             <input type="text" maxlength="
                                    1" id="deficiente" name="
                                    deficiente" value="${
                                    fieldValue(bean:candidato,
                                    field:'deficiente')}"/>
136                         </td>
137                     </tr>
138
139                     <tr class="prop">
140                         <td valign="top" class="name">
141                             <label for="deferido">Deferido
                                    :</label>
142                         </td>
143                         <td valign="top" class="value ${
                                hasErrors(bean:candidato,field
                                :'deferido','errors')}">
144                             <input type="text" maxlength="
                                    1" id="deferido" name="
                                    deferido" value="${
                                    fieldValue(bean:candidato,
                                    field:'deferido')}"/>
145                         </td>
146                     </tr>
147
148                     <tr class="prop">
```

```
149                                 <td valign="top" class="name">
150                                     <label for="idConcurso">Id
                                            Concurso:</label>
151                                 </td>
152                                 <td valign="top" class="value ${
                                        hasErrors(bean:candidato,field
                                        :'idConcurso','errors')}">
153                                     <g:select optionKey="id" from=
                                            "${Concurso.list()}" name=
                                            "idConcurso.id" value="${
                                            candidato?.idConcurso?.id}
                                            " ></g:select>
154                                 </td>
155                             </tr>
156
157                             <tr class="prop">
158                                 <td valign="top" class="name">
159                                     <label for="distribuicoes">
                                            Distribuicoes:</label>
160                                 </td>
161                                 <td valign="top" class="value ${
                                        hasErrors(bean:candidato,field
                                        :'distribuicoes','errors')}">
162
163 <ul>
164 <g:each var="d" in="${candidato?.distribuicoes?}">
165     <li><g:link controller="distribuicao" action="show" id="${d.id
            }">${d?.encodeAsHTML()}</g:link></li>
166 </g:each>
167 </ul>
168 <g:link controller="distribuicao" params="['candidato.id':
        candidato?.id]" action="create">Add Distribuicao</g:link>
169
170                                 </td>
171                             </tr>
172
173                         </tbody>
174                     </table>
175                 </div>
176                 <div class="buttons">
177                     <span class="button"><g:actionSubmit class="
                            save" value="Update" /></span>
178                     <span class="button"><g:actionSubmit class="
                            delete" onclick="return confirm('Are you
                            sure?');" value="Delete" /></span>
179                 </div>
180             </g:form>
181         </div>
```

```
182     </body>
183 </html>
```

**Listing B.4:** Página para edição/exclusão de candidato gerada automaticamente

```
1  <html>
2      <head>
3          <meta http-equiv="Content-Type" content="text/html;
               charset=UTF-8"/>
4          <meta name="layout" content="main" />
5          <title>Candidato List</title>
6      </head>
7      <body>
8          <div class="nav">
9              <span class="menuButton"><a class="home" href="${
                   createLinkTo(dir:'')}">Home</a></span>
10             <span class="menuButton"><g:link class="create" action
                   ="create">New Candidato</g:link></span>
11         </div>
12         <div class="body">
13             <h1>Candidato List</h1>
14             <g:if test="${flash.message}">
15             <div class="message">${flash.message}</div>
16             </g:if>
17             <div id="candidatoList">
18       <div class="list">
19       <table>
20       <thead>
21       <tr>
22          <filter:sortableColumn property="id" title="Id" bean="
               Candidato" update="candidatoList"/>
23          <filter:sortableColumn property="inscricao" title="
               Inscricao" bean="Candidato" update="candidatoList"/>
24          <filter:sortableColumn property="nome" title="Nome" bean=
               "Candidato" update="candidatoList"/>
25          <filter:sortableColumn property="logradouro" title="
               Logradouro" bean="Candidato" update="candidatoList"/>
26          <filter:sortableColumn property="complemento" title="
               Complemento" bean="Candidato" update="candidatoList"
               />
27          <filter:sortableColumn property="bairro" title="Bairro"
               bean="Candidato" update="candidatoList"/>
28          <filter:sortableColumn property="idConcurso" title="
               Concurso" bean="Candidato" update="candidatoList"/>
29       </tr>
30       </thead>
31       <tbody>
32       <g:each in="${candidatoList}" status="i" var="candidato">
33          <tr class="${(i % 2) == 0 ? 'odd' : 'even'}">
```

```
34          <td><g:link action="show" controller="candidato" id="${
                candidato.id}">${fieldValue(bean:candidato,field:'id'
                )}</g:link></td>
35          <td>${fieldValue(bean:candidato, field:'inscricao')}</td>
36          <td>${fieldValue(bean:candidato, field:'nome')}</td>
37          <td>${fieldValue(bean:candidato, field:'logradouro')}</td
                >
38          <td>${fieldValue(bean:candidato, field:'complemento')}</
                td>
39          <td>${fieldValue(bean:candidato, field:'bairro')}</td>
40          <td>${fieldValue(bean:candidato, field:'concurso')}</td>
41          </tr>
42        </g:each>
43        </tbody>
44        </table>
45        </div>
46        <div class="paginateButtons">
47        <filter:paginate total="${Candidato.count()}" bean="
              Candidato" update="candidatoList"/>
48        </div>
49        </div>
50          </div>
51      </body>
52  </html>
```

**Listing B.5:** Página para pesquisa de candidato gerada automticamente

```
1
2
3   <html>
4       <head>
5           <meta http-equiv="Content-Type" content="text/html;
                charset=UTF-8"/>
6           <meta name="layout" content="main" />
7           <title>Show Candidato</title>
8       </head>
9       <body>
10          <div class="nav">
11              <span class="menuButton"><a class="home" href="${
                    createLinkTo(dir:'')}">Home</a></span>
12              <span class="menuButton"><g:link class="list" action="
                    list">Candidato List</g:link></span>
13              <span class="menuButton"><g:link class="create" action
                    ="create">New Candidato</g:link></span>
14          </div>
15          <div class="body">
16              <h1>Show Candidato</h1>
17              <g:if test="${flash.message}">
18              <div class="message">${flash.message}</div>
```

```
19              </g:if>
20              <div class="dialog">
21                  <table>
22                      <tbody>
23
24
25                          <tr class="prop">
26                              <td valign="top" class="name">Id:</td>
27
28                              <td valign="top" class="value">${
                                    fieldValue(bean:candidato, field:'
                                    id')}</td>
29
30                          </tr>
31
32                          <tr class="prop">
33                              <td valign="top" class="name">
                                    Inscricao:</td>
34
35                              <td valign="top" class="value">${
                                    fieldValue(bean:candidato, field:'
                                    inscricao')}</td>
36
37                          </tr>
38
39                          <tr class="prop">
40                              <td valign="top" class="name">Nome:</
                                    td>
41
42                              <td valign="top" class="value">${
                                    fieldValue(bean:candidato, field:'
                                    nome')}</td>
43
44                          </tr>
45
46                          <tr class="prop">
47                              <td valign="top" class="name">
                                    Logradouro:</td>
48
49                              <td valign="top" class="value">${
                                    fieldValue(bean:candidato, field:'
                                    logradouro')}</td>
50
51                          </tr>
52
53                          <tr class="prop">
54                              <td valign="top" class="name">
                                    Complemento:</td>
```

```
55
56                              <td valign="top" class="value">${
                                    fieldValue(bean:candidato, field:'
                                    complemento')}</td>
57
58                          </tr>
59
60                          <tr class="prop">
61                              <td valign="top" class="name">Bairro
                                    :</td>
62
63                              <td valign="top" class="value">${
                                    fieldValue(bean:candidato, field:'
                                    bairro')}</td>
64
65                          </tr>
66
67                          <tr class="prop">
68                              <td valign="top" class="name">Cep:</td
                                    >
69
70                              <td valign="top" class="value">${
                                    fieldValue(bean:candidato, field:'
                                    cep')}</td>
71
72                          </tr>
73
74                          <tr class="prop">
75                              <td valign="top" class="name">Cidade
                                    :</td>
76
77                              <td valign="top" class="value">${
                                    fieldValue(bean:candidato, field:'
                                    cidade')}</td>
78
79                          </tr>
80
81                          <tr class="prop">
82                              <td valign="top" class="name">Uf:</td>
83
84                              <td valign="top" class="value">${
                                    fieldValue(bean:candidato, field:'
                                    uf')}</td>
85
86                          </tr>
87
88                          <tr class="prop">
```

```
89                          <td valign="top" class="name">Telefone
                                :</td>
90
91                          <td valign="top" class="value">${
                                fieldValue(bean:candidato, field:'
                                telefone')}</td>
92
93                      </tr>
94
95                      <tr class="prop">
96                          <td valign="top" class="name">Dt Ult
                                Atualizacao:</td>
97
98                          <td valign="top" class="value">${
                                fieldValue(bean:candidato, field:'
                                dtUltAtualizacao')}</td>
99
100                     </tr>
101
102                     <tr class="prop">
103                         <td valign="top" class="name">Email:</
                                td>
104
105                         <td valign="top" class="value">${
                                fieldValue(bean:candidato, field:'
                                email')}</td>
106
107                     </tr>
108
109                     <tr class="prop">
110                         <td valign="top" class="name">
                                Deficiente:</td>
111
112                         <td valign="top" class="value">${
                                fieldValue(bean:candidato, field:'
                                deficiente')}</td>
113
114                     </tr>
115
116                     <tr class="prop">
117                         <td valign="top" class="name">Deferido
                                :</td>
118
119                         <td valign="top" class="value">${
                                fieldValue(bean:candidato, field:'
                                deferido')}</td>
120
121                     </tr>
```

```
122
123                     <tr class="prop">
124                         <td valign="top" class="name">Id
                                Concurso:</td>
125
126                         <td valign="top" class="value"><g:link
                                 controller="concurso" action="
                                show" id="${candidato?.idConcurso
                                ?.id}">${candidato?.idConcurso?.
                                numero?.encodeAsHTML()}</g:link></
                                td>
127
128                     </tr>
129
130                     <tr class="prop">
131                         <td valign="top" class="name">
                                Distribuicoes:</td>
132
133                         <td  valign="top" style="text-align:
                                left;" class="value">
134                             <ul>
135                             <g:each var="d" in="${candidato.
                                    distribuicoes}">
136                                 <li><g:link controller="
                                        distribuicao" action="show
                                        " id="${d.id}">${d?.
                                        encodeAsHTML()}</g:link></
                                        li>
137                             </g:each>
138                             </ul>
139                         </td>
140
141                     </tr>
142
143                 </tbody>
144             </table>
145         </div>
146         <div class="buttons">
147             <g:form>
148                 <input type="hidden" name="id" value="${
                        candidato?.id}" />
149                 <span class="button"><g:actionSubmit class="
                        edit" value="Edit" /></span>
150                 <span class="button"><g:actionSubmit class="
                        delete" onclick="return confirm('Are you
                        sure?');" value="Delete" /></span>
151             </g:form>
152         </div>
```

```
153             </div>
154         </body>
155     </html>
```

**Listing B.6:** Página para visualização de candidato gerada automaticamente