UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM WEB E SISTEMAS DE INFORMAÇÃO

CHEFERSON WASHINGTON LOVATTO

COMPARANDO AMBIENTES DE DESENVOLVIMENTO JAVA E RUBY ON RAILS

Trabalho de Conclusão apresentado como requisito parcial para a obtenção do grau de Especialista

Prof. M.E. Henrique J. Brodbeck
Orientador

Prof. Dr. Carlos Alberto Heuser
Coordenador do Curso

Porto Alegre, dezembro de 2007.

AGRADECIMENTOS

Em nossas vidas estamos sempre rodeados de pessoas que de alguma forma, boa ou não, influenciam os caminhos que tomamos, tanto profissional quanto afetivamente. Então quero, como sempre, enfatizar apenas a importância daquelas que me conduzem para bons caminhos. Entre elas estão meus pais, alguns parentes e alguns amigos, mas, principalmente, a minha esposa Laurilei a quem chamo carinhosamente de Paixãozinha, a qual tem sido meu alicerce, apoio e incentivadora nos meus caminhos e foi quem mais me apoiou a iniciar e finalizar essa Pós-graduação. A minha Amada dedico não só esse trabalho de conclusão, mas toda a minha vida e todo meu amor, o qual ela sabe que é do tamanho de “dois dedinhos”.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

MVC	Models-Views-Controllers
BD	Banco de Dados
IIS	Internet Information Services
ORM	Object Relational Mapping
RSS	Rich Site Summary ou Really Simple Syndication
ER	Entidade-Relacionamento
CSS	Cascading Style Sheets
CGI	Common Gateway Interface
MB	Mega Byte
GB	Giga Byte
TB	Tera Byte
SQL	Structured Query Language
RAM	Random Access Memory

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Com a rápida evolução e disponibilização de novas tecnologias, cada vez mais se observa uma dificuldade em decidir até que ponto uma nova ferramenta poderá ou não auxiliar nos processos de desenvolvimentos de aplicações para *web*. Esse é o caso do *framework* Ruby on Rails, que, embora muito recente, vem recebendo boas críticas, porém, na maioria dos casos, são elogios exaltados considerando algumas facilidades observadas num primeiro contato.

Para que se possa oferecer uma avaliação mais precisa, este trabalho traçará um comparativo entre resultados obtidos no desenvolvimento de uma aplicação em ambiente Java e em ambiente Ruby on Rails. A opção de se comparar com Java se deve ao fato de se tratar de uma tecnologia muito bem consolidada e ser muito difundida e utilizada.

Como escopo de comparação será avaliado, durante o desenvolvimento de uma aplicação *web*, o tempo de desenvolvimento, o tamanho do código gerado, a facilidade na obtenção de informações sobre a linguagem, desempenho de execução e facilidade na manutenção do aplicativo.

Palavras-Chave: Ruby on Rails, *Framework*, Ruby, Desenvolvimento Web.

COMPARING DEVELOPMENT ENVIRONMENT OF JAVA AND RUBY ON RAILS

ABSTRACT

With the fast evolution and availability of new technologies, it becomes difficult to decide if a new tool could be helpful in the development process of web applications. This is the case of framework Ruby on Rails, that, in spite of being recent, is well evaluated. However, in the majority of the cases, it is enthusiasticaly celebrated considering the easinesses of development observed in a first contact.

In order to offer a more precise evaluation, this work will draw a comparison between the results of measurements taken in the development of an application in Java environment and in Ruby on Rails environment. The option of comparing Ruby on Rails with Java is supported by the fact it is a very consolidated and spread out technology.

The main measures that will be evaluated during the development of a web application web are the development time, the size of the generated code, the easiness of finding information on the language, performance of execution and easiness of maintenance of the application.

Keywords: Ruby on Rails, *Framework*, Ruby, Web Development.

1 INTRODUÇÃO

Após quase dois anos do curso de especialização em *Web* e Sistema de Informação (edição 2006/2007) pudemos aprimorar nossos conhecimentos e conhecer diversas tecnologias disponíveis atualmente para o desenvolvimento dos mais variados aplicativos. Tivemos oportunidade de estudar metodologias e ferramentas que nos auxiliam na tarefa de análise e desenvolvimento, mas, como o tempo era "enxuto", muitas vezes não se conseguia o aprofundamento necessário e, conseqüentemente, era necessário que avançássemos sozinhos utilizando nossa disponibilidade de tempo.

Então, à medida que se avançava nas diferentes disciplinas, éramos municiados com conhecimentos para desenvolvermos aplicativos, principalmente para a *Web*, que se tornou um recurso poderosíssimo para encurtar distâncias e possibilitar soluções de problemas que exigem informações atualizadas de forma rápida e segura.

O desenvolvimento na *Web* vem avançando a ritmo acelerado. O que começou com conteúdo estático no passado e evoluiu para sistemas dinâmicos com acesso a bancos de dados em tempo real, chega nos dias de hoje a permitir a criação de aplicativos com processamento distribuído ou paralelo, aplicativos que possibilitam a obtenção de informações relevantes (*Web* Semântica) e outros que obtém conhecimento e fazem uso deles. Todas essas possibilidades de desenvolvimento cercadas de segurança e criptografia para acesso aos dados.

O crescimento é tanto que, para facilitar o desenvolvimento na *Web*, fica cada vez mais relevante à necessidade de termos ferramentas que tornem mais produtivo nosso trabalho. Nesse item percebe-se o surgimento e aperfeiçoamento de muitas metodologias e linguagens de programação que evoluíram desde a programação estruturada, passando pela orientação a objetos e orientação a eventos.

Na mesma proporção que as metodologias e linguagens de programação foram sendo aperfeiçoadas ou criadas, tivemos um crescimento nas opções de editores de linguagens, ferramentas para modelagem de dados e outras mais, para atender essa nova demanda.

1.1 Objetivos

Embora o avanço das tecnologias venha permitindo o oferecimento de novas e poderosas soluções para as necessidades do universo da informática, uma exigência continua sendo imposta. A diminuição do ciclo produtivo de aplicativos.

Para que a produção de aplicativos seja agilizada e demande o menor tempo possível, vem se verificando o crescimento de ambientes de desenvolvimento que minimizam tarefas simples e repetitivas para que o desenvolvedor possa usar o tempo disponível para tarefas mais complexas.

E é nesse ponto específico que vem se tornando muito conhecido o *framework* Ruby on Rails. Este *framework* é um projeto *open source* baseado na linguagem de programação Ruby, a qual é totalmente orientada a objeto. Seu destaque principal, enfatizado por Tate e Hibbs (2006), é que tem se tornado um dos *frameworks* de desenvolvimento para *web* mais produtivo de todos os tempos.

1.2 Resultados Esperados

O presente trabalho tem como objetivo estabelecer um comparativo entre resultados obtidos no desenvolvimento de uma aplicação em ambiente Java e em ambiente Ruby on Rails no que diz respeito a tempo de desenvolvimento, tamanho de código, facilidade na obtenção de informações sobre a linguagem, performance de execução e facilidade na manutenção do aplicativo.

Com a comparação dos ambientes de desenvolvimento Java e Ruby on Rails, se espera obter uma relação das vantagens e desvantagens de cada ambiente, focando-se, principalmente, nos aspectos produtivos (tempo de desenvolvimento) e manutenção de código.

Além dos pontos de comparação, se pretende verificar os principais recursos do *framework* Ruby on Rails e entender sua estrutura e organização para auxiliar a quem for se aventurar na sua utilização para o desenvolvimento na *web*.

1.3 Abordagem Adotada

Para isso, esse trabalho apresentará no capítulo 2 as principais características de um ambiente Ruby on Rails, abordando aspectos inerentes a linguagem Ruby e detalhando a estrutura deste poderoso *framework*. Também será fornecida uma idéia de sua utilização na construção de aplicativos na *Web*, bem como a sua difusão quanto a provedores de hospedagem e disponibilização de IDEs para auxílio nos desenvolvimentos em Ruby on Rails.

Após, no capítulo 3, será apresentada a definição do aplicativo proposto para efetuar a comparação dos dois ambientes de desenvolvimento. Neste capítulo ainda serão comentadas as fases do desenvolvimento no ambiente Ruby on Rails para que se possa ter uma idéia mais precisa do que esse *framework* oferece para tornar o desenvolvimento de aplicativos *Web* mais produtivo e fascinante.

Por fim no capítulo 4 serão apresentadas as comparações efetuadas entre os ambientes de desenvolvimento Java e Ruby on Rails, abordando aspectos de métrícas relacionadas a código fonte da aplicação gerada, avaliação de performance de execução, facilidade na manutenção do aplicativo e na obtenção de informações sobre cada ambiente.

2 CARACTERÍSTICAS DO AMBIENTE RUBY ON RAILS

2.1 Linguagem Ruby

Ruby é uma linguagem interpretada utilizada para programação orientada a objetos criada por Yukihiro Matsumoto em 1994, o qual a qualifica como sendo "simples, direto ao ponto, extensível e portável".

A linguagem Ruby foi desenvolvida com diversos recursos para processar arquivos de texto e para realizar tarefas de gerenciamento de sistema (OLIVEIRA JUNIOR, 2006). Ela foi baseada em diversas linguagens, procurando tirar o que há de melhor de cada uma. Sua sintaxe simples, parcialmente inspirada por *Eiffel* e *Ada*, enquanto que seus recursos de tratamento de exceções têm similaridades com Java e Python. Ruby é uma linguagem completa e puramente orientada a objetos. Isso significa que todo dado em Ruby é um objeto, exatamente como *SmallTalk*.

2.1.1 Recursos do Ruby

Abaixo são relacionados alguns dos principais recursos da linguagem Ruby (OLIVEIRA JUNIOR – tutorial s.d.):

- A orientação a objetos do Ruby foi elaborada para ser completa e aberta a melhorias. Por exemplo, Ruby tem a habilidade de adicionar métodos em uma classe, ou até mesmo em uma instância durante o *runtime*.
- Ruby tem, intencionalmente, herança única, pois se entende que, utilizando o conceito de módulos (chamados de *Categories* no *Objective-C*), obtém-se uma forma mais limpa que herança múltipla.
- Ruby implementa *closures* verdadeiras, permitindo que funções criadas no interior de outras funções possam referenciar as variáveis mesmo após estar fora do escopo.
- Ruby, como em outras linguagens, tem blocos em sua sintaxe (código delimitado por '{'. . . '}' ou 'do'. . . 'end'), porém com a vantagem que eles podem ser passados para os métodos, ou, até mesmo,convertidos em *closures*.
- O *garbage collector* do Ruby é totalmente funcional, ou seja, ele marca e limpa, pois atua em todos os objetos do Ruby de maneira que não há a necessidade de manter contagem de referências em bibliotecas externas.

- Em Ruby há inteiros com representação interna para números pequenos (instância da classe *Fixnum*) e para números grandes (*Bignum*), entretanto, não é necessário se preocupar qual deve ser utilizada, pois ocorre uma conversão automaticamente quando necessário.
- Ao invés de utilizar declaração de variáveis, Ruby usa, apenas, convenção de nomenclatura para delimitar o escopo das variáveis. Assim temos: 'var' para variável local, '@var' para variável de instância e '$var' para variável global.
- Ruby tem um sistema de *threading* independente do sistema operacional, assim em qualquer plataforma que se rode Ruby, se poderá fazer uso de *multithreading*.
- A portabilidade do Ruby é muito alta, pois funciona em muitos tipos de UNIX, DOS, Windows 95/98/Me/NT/2000/XP, MacOS, BeOS, OS/2, além, é claro, do Linux.

A linguagem Ruby possui um gerenciador de pacotes, o *RubyGems*, com o qual se pode instalar facilmente vários plugins com a seguinte sintaxe*: gem install* <*nome_plugin*>. O RubyGems é capaz de fazer todo o trabalho de instalação de diversos *plugins*, baixando os arquivos necessários e instalando-os nos locais apropriados nas bibliotecas do Ruby

2.2 O *Framework*

O Ruby on Rails é um *framework* para a linguagem Ruby que, como qualquer outro, visa criar uma estrutura de auxílio no desenvolvimento de softwares. Com ele o desenvolvedor não precisa gastar esforço em código repetitivo.

Entretanto, o Ruby on Rails vem se destacando como um dos *frameworks* para *Web* mais produtivos até o momento (HIBBS & TATE, 2006). Com ele é possível criar pequenas aplicações com acesso à banco de dados e validações em menos de 10 minutos e mesmo assim ter um código gerado rápido e bem organizado, utilizando a filosofia modelo-visualização-controlador (MVC).

2.2.1 Os recursos do Rails

O poder produtivo do Rails está embutido em diversos recursos e muitos deles foram criados com a utilização dos demais. A seguir veremos os principais recursos:

- Metaprogramação é a técnica utilizada para que programas possam criar programas. Nesse quesito a linguagem de programação Ruby aparece atualmente como uma das melhores e em consequência o Rails faz uso da metaprogramação de uma forma muitíssimo bem explorada;
- Active Record é um *framework* dentro do Rails que, utilizando-se da metaprogramação, cria objetos para acesso ao BD de uma forma simples, elegante e poderosa;
- Convenção sobre configuração é o ponto chave do Rails, pois, seguindo as convenções previamente estabelecidas, o *framework* consegue encontrar a maioria das informações que necessita e executar muitas tarefas automaticamente sem haver a necessidade de escrever longos trechos de configuração para sua aplicação;

- O Rails disponibiliza três ambientes para sua aplicação. Uma para desenvolvimento, outra para teste e mais uma para produção. Cada uma possui um comportamento um pouco diferente, associando um BD para cada ambiente.

2.2.2 A organização

Além de utilizar esses recursos e outros mais como *Ajax* e testes incorporados, o Rails mantém, em sua estrutura, uma organização de pastas e subpasta para armazenar tudo que é necessário a cada aplicação. As principais pastas, cuja estrutura o Rails cria automaticamente, são as seguintes (OLIVEIRA JUNIOR – tutorial s.d.):

- app: uma das principais pastas da estrutura do Rails onde são armazenadas em suas subpastas os componentes de cada aplicação (*views*, *models* e *controllers*);
- config: como já foi comentado anteriormente o Rails faz uso de convenções ao invés de configurações, mas mesmo assim necessita de um mínimo de configuração (banco de dados, estrutura do ambiente e roteamento de solicitações), as quais são armazenadas nessa pasta;
- doc: toda a documentação de uma aplicação é armazenada nessa pasta. Salienta-se que Ruby possui um gerador de documentação, o *RubyDoc*, que gera automaticamente a documentação da aplicação;
- log: armazena os diferentes arquivos de "log" de erro gerados em cada ambiente (server.log, development.log e outros);
- public: aqui são mantidos os arquivos para acesso a toda aplicação (*javascripts*, folhas de estilo e arquivos *html*);
- script: mantém todos os *scripts* necessários para a geração da aplicação (server, generate e outros).

2.2.3 Servidores para aplicação

Como qualquer aplicação *web*, as aplicações criadas com o Ruby on Rails necessitam de um servidor para que se possa executá-la. Assim o Rails disponibiliza um servidor próprio para sua aplicação, o *WEBRick*, que é escrito totalmente em Ruby e vem incorporado ao Rails. Entretanto, se pode executar uma aplicação Ruby em vários servidores web diferentes, entre eles:

- Apache: por ser uma opção de servidor *web* mais escalável e flexível, é a opção mais indicada para se utilizar com segurança. Com o Apache se pode fazer uso da variedade de *plug-ins* existentes para suportar diversas linguagens de programação e obter vários tipos de conteúdo dinâmico;
- Lighttpd: é um servidor muito leve que oferece conteúdo estático com muita rapidez utilizando uma interface *FastCGI* e assim atende a aplicações Rails com uma velocidade muito boa.
- Mongrel: é um servidor *web* bem recente que combina a simplicidade do *WEBRick* e a velocidade do *Lighttpd*. Tem se mostrado como uma ótima escolha para uso em desenvolvimento e produção.

Além desses servidores *web*, teoricamente, o Rails pode ser executado em qualquer servidor *web* que suporte CGI. Inclusive já se encontram na internet instruções disponibilizadas por desenvolvedores Rails para executar suas aplicações no *Internet*

Information Services (IIS) da Microsoft como, por exemplo, é apresentado no endereço *http://wiki.rubyonrails.org/rails/pages/HowtoSetupIIS*.

2.2.4 Active Record

De todos os recursos do Rails, o centro nervoso é o *Active Record*. Ele trabalha utilizando-se dos recursos das convenções, para reduzir drasticamente o número de linhas de configuração, e da metaprogramação, a qual adiciona dinamicamente diversos recursos às classes da aplicação tendo como base o conteúdo e estrutura do banco de dados.

O *Active Record* do Rails é uma implementação do padrão de projeto catalogado por Martin Fowler em seu livro *Patterns of Enterprise Architecture* que pode ser acessado em http://www.martinfowler.com/eaaCatalog/activeRecord.html. Porém o Rails estendeu esse padrão como mostrado na tabela 2.1.

Tabela 2.1: Diferenças e Vantagens do Active Record do Rails

Diferença	Vantagem
O Rails adiciona atributos automaticamente, com base nas colunas do banco de dados.	Os desenvolvedores não precisam especificar atributos em mais de um local.
O Rails adiciona gerenciamento e validação de relacionamento por meio de uma linguagem interna personalizada.	Os desenvolvedores de Rails podem declarar relacionamentos e validação baseada em modelo para que sejam gerenciados pelo *framework* sem se basearem na geração do código.
As convenções de nomenclatura do Rails permitem que o banco de dados descubra campos específicos.	Os desenvolvedores de Rails não precisam configurar chaves primárias e estrangeiras porque o *Active Record* as descobre automaticamente.

Fonte: HIBBS & TATE, 2006. p. 19.

Além do uso de convenções, o *Active Record* do Rails utiliza uma estratégia de envolvimento do banco de dados e não o mapeamento relacional de objetos (ORM) utilizado por desenvolvedores de Java.

Com isso cada tabela do BD é envolvida em classe do *Active Record* e assim o *framework* Rails pode descobrir automaticamente as colunas das tabelas do BD e adicioná-las de forma dinâmica à classe do *Active Record* (HIBBS & TATE, 2006).

2.2.5 Console do Rails

Com a finalidade de permitir a interação com os modelos das aplicações, o Rails cria, juntamente com cada projeto, uma *console*. Essa ferramenta, ao ser inicializada, executa a conexão com o BD e carrega todas as classes do *Active Record* que estão em aap/model. Dessa forma é possível interagir com o modelo, inclusive nas operações de BD.

Para que uma *console* seja inicializada, basta executar o comando: *ruby script/console*. Assim você pode manipular os dados de sua aplicação, visualizando, inserindo, modificando e excluindo-os.

2.3 Principais sites que o utilizam

Sem muito esforço consegue-se facilmente encontrar diversos sites que estão utilizando a linguagem de programação Ruby. Em especial, encontra-se uma lista com diversos sites criados com ruby no site *http://happycodr.com/biglist*, entre eles:

- http://www.12stoneart.com/: uma galeria de arte online que expõe imagens de obras de arte;
- http://acunote.com/: ferramenta para gerenciamento de projetos;
- http://agilewebdevelopment.com/: disponibiliza diversos *plugins* de Ruby on Rails;
- http://atopsites.com/: definido com sendo um "clone" do *alexa.com*;
- http://www.bandsintown.com/: site com informações sobre bandas, músicas e locais de shows disponíveis próximo ao local onde se está acessando o site;
- http://www.baom.net/: site de *download* de músicas mediante cadastro prévio;
- http://www.benevolus.com: site para envio de cartões mediante cadastro prévio para obter *login* e senha;
- http://www.brownandco-sf.com/: site de compra e venda de imóveis de uma empresa de *San Franciscan*;
- http://www.my3w.org/: site com diversos exemplos de folhas de estilo. Embora muito funcional, há muito conteúdo em uma língua oriental que torna difícil a tradução de alguns conteúdos;
- http://www.dailyoped.com/: site para leitura e procura de notícias em diversos jornais americanos, utilizando *Rick Site Sumary* ou *Really Simple Syndication* (RSS).

Abaixo são apresentados alguns sites desenvolvidos pela empresa 37Signals que é uma das pioneiras na utilização de Ruby on Rails:

- http://www.basecamphq.com: site lançado pela 37Signals em fevereiro de 2004 para gerenciamento de projetos de forma facilitada e muito elegante;
- http://www.highrisehq.com: site de gerenciamento de contatos e tarefas onde cada usuário tem sua própria página e pode interagir com os demais contatos marcando reuniões, trocando mensagens e realizando conversações;
- http://www.backpackit.com: outro site de gerenciamento de tarefas que mantém um calendário que permite fazer anotações e agendar compromissos;
- http://www.campfirenow.com: site de controle de grandes equipes de projetos que permite o trabalho colaborativo entre as equipes com troca de arquivos compartilhada.

2.4 IDE e hospedagem

2.4.1 IDEs

Atualmente já existem diversos IDE's (*Integrated Development Environment*) para se desenvolver no *framework* Ruby on Rails. Os que se destacam são os seguintes:

- *RadRails* da *Aptana* foi desenvolvido e totalmente baseado no *Eclipse*. Vem sendo considerado um dos melhores IDE's atualmente. Maiores detalhes podem ser encontrados no endereço *http://radrails.org* ;
- *jEdit* é um IDE que possui bons *plugins* para trabalhar com Ruby and Rails. Pode-se obtê-lo em *http://rubyjedit.org/*;
- *Eclipse* também é uma boa opção. Basta utilizar um *plugin* que pode ser encontrado em *http://www.aptana.com/download_rails_rdt.php*;
- *Mondrian* é um IDE escrito totalmente em Ruby e se pode fazer o *download* de uma versão *trial* em *http://www.mondrian-ide.com/*;
- *Gedit* também pode ser utilizado desde que utilize um *plugin* disponibilizado em *http://rec6.via6.com/link.php?nv=1&l=10117*;
- *Zeusedit* é mais um IDE que já vem pré-configurado para trabalhar com a linguagem Ruby. O mesmo está disponível em *http://www.zeusedit.com/ruby.html*.

2.4.2 Hospedagem

Já com relação à hospedagem facilmente se encontram inúmeros provedores de hospedagem, alguns gratuitos e outros pagos, com uma variedade de recursos que são disponibilizados. Abaixo segue uma relação de alguns desses provedores:

- Dreamhost.com (*http://wiki.dreamhost.com/*): em termos de números, se tem aproximadamente 500 GB de espaço em disco**,** aproximadamente 4 TB de transferência mensal disponibilizado em servidores de alta disponibilidade;
- No site *http://www.rubyonrailswebhost.com/* são indicados os provedores de hospedagem *Site5-Tied For First* (*http://www.site5.com/hosting/rails.php*) e *HostGator* (*http://hostgator.com/*);
- Vlexo (*http://vlexo.com/*) provedores com opção gratuita, porém com restrição no tráfego de dados mensal (1 a 10MB) e páginas gratuitas devem ser digitadas em inglês.
- Vilago (*http://www.vilago.com.br/hospedagem-php5-ruby-on-rails*) possui diversos planos de hospedagem que suportam Ruby on Rails com 500MB de espaço em disco, 25GB de transferência por mês e outros recursos mais;
- TeHospedo (*http://www.tehospedo.com.br/hospedagem/*) de forma similar ao Vilago, disponibiliza vários recursos, porém para Ruby on Rails está em fase de testes.
- MegaInternet (*http://www.megainternet.com.br/hospedagem-ruby-on-rails*) oferece o serviço de hospedagem para Ruby on Rails no Brasil desde 2006 com diversas opções de planos.

- Delix DataCenter é outra opção de hospedagem no Brasil para Ruby on Rails (*http://www.delix.com.br/servicos/hospedagem_de_sites/ruby_on_rails/*).

3 O AMBIENTE DE TESTE

3.1 Definição da Aplicação a ser Desenvolvida

Para efeito de possibilitar a realização do comparativo entre o desenvolvimento nos ambiente Java e Ruby on Rails foi desenvolvida a mesma aplicação de acordo com a especificação a seguir:

- A aplicação tem por objetivo permitir o controle das atividades que se está realizando em determinado projeto;
- Esse controle possibilita identificar a pessoa que está realizando essa atividade do projeto informado, bem como a data de início da atividade e a data limite para sua conclusão, além do *status* dessa atividade;
- Os *status* das atividades são assim identificados: 0 (Não Iniciada), 1 (Iniciada) ou 2 (Concluída);
- Uma atividade pode ser executada por uma pessoa;
- Uma atividade pertence a apenas um projeto;
- Uma pessoa pode estar trabalhando em mais de uma atividade;
- Um projeto pode conter uma ou mais atividades;
- O controle dos projetos deve englobar a manutenção das tabelas envolvidas (pessoas, projetos e atividades), sendo possível incluir, excluir, listar e alterar suas informações;
- O aplicativo deverá fazer as validações do preenchimento dos campos, bem como do conteúdo dos campos conforme necessário;
- O visual da aplicação deverá ser o mesmo, sendo controlado por folha de estilo.

3.1.1 Modelo Entidade-Relacionamento da Aplicação

Para o desenvolvimento das aplicações foi utilizado o mesmo modelo ER (Entidade-Relacionamento) conforme figura 3.1, onde temos apresentado as relações entre as tabelas envolvidas.

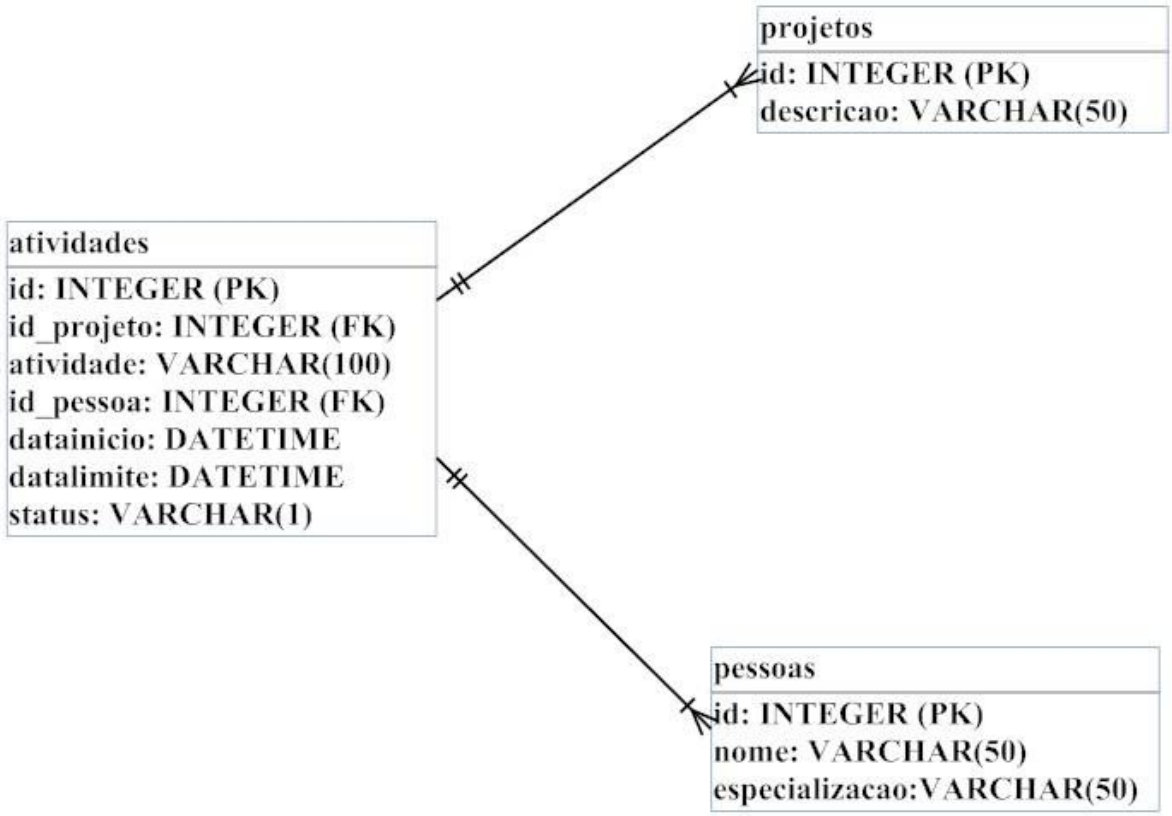


Figura 3.1: Modelo ER

3.2 O Ambiente de Desenvolvimento Java

No ambiente de desenvolvimento Java foi utilizado a versão jdk1.6.0_01 que foi obtida no endereço http://java.sun.com/javase/downloads/index.jsp.

No ambiente Java utilizado para o desenvolvimento do aplicativo proposto utilizou-se o NetBeans IDE versão 5.5.1, o qual foi obtido no endereço http://java.sun.com/javase/downloads/index.jsp. Na sua configuração optou-se por utilizar o servidor TomCat versão 5.5.17 que já veio embutido no IDE utilizado.

O equipamento utilizado no desenvolvimento do aplicativo proposto tanto para o ambiente Java quanto para Ruby on Rails possui a seguinte configuração:

- Intel Celeron M CPU 410
- Processador 1.46 GHz
- 512 MB de memória RAM
- Sistema Operacional Windows XP Professional

3.3 Preparação de um Ambiente de Desenvolvimento Ruby on Rails

Como já foi mencionado anteriormente, a linguagem Ruby, bem como o *framework* Rails, são disponibilizados para diversas plataformas, porém para o desenvolvimento da aplicação proposta foi utilizado o ambiente Windows. Para as demais plataformas podem-se obter os pacotes de instalação em *http://www.ruby-lang.org/en/downloads/*.

Para o ambiente Windows utilizou-se um instalador que pode ser obtido em *http://rubyforge.org/projects/rubyinstaller/*, com o qual se tem um ambiente Ruby completo que inclui toda sua documentação e editores de texto (RUBYONBR – site s.d.). A versão atua,l e que foi utilizada para o desenvolvimento da aplicação, é a 1.8.6-25.

Uma vez tendo sido instalado o ambiente Ruby, o próximo passo é fazer a instalação do *framework* Rails. Para isso devemos nos posicionar na pasta onde o Ruby foi instalado (usualmente C:\Ruby) e executar o seguinte comando (RUBYONBR – site s.d.):

C:\Ruby> gem install rails --include-dependencies

Ao término desse processo, e tendo de antemão um banco de dados disponível, se terá o ambiente Ruby on Rails preparado para se iniciar o desenvolvimento de aplicação. Para a aplicação proposta foi utilizado o banco de dados MySQL que pode ser obtido em *http://www.mysql.org/*.

3.4 Etapas do desenvolvimento

Para o desenvolvimento da aplicação no *framework* Ruby on Rails, basicamente, deve-se executar algumas tarefas listadas abaixo e que serão detalhadas mais adiante (HIBBS & TATE, 2006):

- Configurar o *Active Record*;
- Gerar os modelos para o *Active Record*;
- Alterar os métodos das migrações do BD geradas junto com os modelos do *Active Record*;
- Criar os relacionamentos entre as tabelas;
- Criar os Controladores e Visualizadores para a aplicação;
- Criar as validações de preenchimento de campos obrigatórios;
- Definir folhas de estilo para a aplicação;
- Ajustar os visualizadores conforme a especificação da aplicação

3.4.1 Configurar o Active Record

Consiste em criar o projeto proposto executando o comando *rails controle*. Assim será criado toda a estrutura da aplicação, bem como o três ambiente de desenvolvimento (teste, desenvolvimento e produção).

Após o término do comando anterior, basta acessar o *MySQL* e criar a base de dados com o nome *controle_development* e em seguida configurar o BD a ser utilizado no arquivo *database.yml* da subpasta *config* conforme abaixo:

```
development:
        adapter: mysql
    database: controle_development
    username: <seu usuário>
    password: <sua senha>
    host: localhost
```

3.4.2 Gerar os modelos para o Active Record;

Para a aplicação proposta será necessário a geração de modelos para as tabelas pessoas, projetos e atividades. Esses modelos serão gerados pelos comandos *ruby script/generate model Pessoa*, *ruby script/generate model Projeto* e *ruby script/generate model Atividade*, respectivamente.

Com isso teremos criado, na estrutura do projeto existente, os modelos necessários para atender a definição da aplicação.

3.4.3 Alterar os métodos das migrações do BD geradas junto com os modelos do Active Record

Nesse ponto podemos decidir entre utilizar *scripts* SQL ou o conceito de migrações do Rails.

Se forem utilizados *scripts* SQL, haverá a necessidade de se manter um controle manual de qualquer atualização que for efetuada no BD, fazendo-se necessária a criação de novos dados de teste a cada modificação.

Agora se decidirmos utilizar o conceito de migrações de esquema do Rails, ao invés de criarmos, diretamente no *MySQL*, as tabelas da aplicação, faremos isso no arquivo *001_create_controle.rb* que foi criado ao ser gerado os modelos da aplicação, na subpasta *db/migrate*. Para tanto basta editar o arquivo e alterar seu métodos *up* e *down* para que fique conforme abaixo, observando que deve-se repetir os comandos *create_table* e *drop_table* para as demais tabelas:

```
class CreateControle < ActiveRecord::Migration
    def self.up
        create_table "pessoas" do |pessoa|
            pessoa.column "nome", :string
            pessoa.column "especialização", :string
        end
    end
    def self.down
        drop_table "pessoas"
    end
end
```

Ao terminar de alterar o arquivo da migração, deve ser executado o comando *rake migrate* que o Rails criará as tabelas com seus campos e manterá um controle do BD.

3.4.4 Criar os relacionamentos entre as tabelas

Agora que as tabelas do BD da aplicação já estão criadas precisamos informar ao Rails quais são os relacionamentos existentes entre elas. Para isso deve-se editar os

modelos de cada tabela e modificá-los para que fiquem semelhantes ao controle da tabela *pessoas*:

```
class Pessoa < ActiveRecord::Base
    has_many :atividades
end
```

Isso indica ao Rails que uma pessoa pode realizar muitas atividades. Então na tabela projeto se utilizará o mesmo relacionamento e na tabela atividades será usado *belong_to :pessoas* e *belong_to :projeto* que informarão ao Rails que uma atividade é realizada por uma pessoa referente a um projeto.

O Rails dispõe de outros tipos de relacionamentos, entre eles:

- has_one: relacionamento um-para-um;
- has_and_belongs_to_many: relacionamento muitos-para-muitos;
- acts_as_list: relacionamento que permite que itens sejam expressos como uma lista ordenada;
- acts_as_tree: relacionamento que permite que itens sejam expressos como uma estrutura de árvore.

3.4.5 Criar os Controladores e Visualizadores para a aplicação;

Com os relacionamentos criados, passa-se a geração dos controladores e suas respectivas visualizações. Isso é efetuado com o *comando ruby script/generate scaffold pessoa* o que irá gerar toda a estrutura e códigos fontes para a manutenção da tabela de *pessoas*.

A estrutura gerada possibilitará, então, que se inclua uma pessoa, altere seus dados, exclua do BD ou apenas liste as pessoas que estejam cadastradas. Obviamente que a aparência das visualizações que são criadas não apresenta um acabamento elegante, mas é totalmente funcional.

Assim a partir desse ponto passa-se a "lapidar" a aplicação, incluindo validações a campos de formulários, layouts com aparência controlada por folhas de estilo e utilização de *javascript* conforme necessário.

Ainda nessa parte do desenvolvimento se fará a geração dos controladores para os modelos de projetos e atividades, utilizando-se do mesmo comando.

3.4.6 Criar as validações de preenchimento de campos obrigatórios

O Rails permite o controle de algumas validações com relação à obrigatoriedade no preenchimento, tamanho e até mesmo de formato de campos dos modelos.

Essas validações devem ser incluídas nos respectivos modelos como no exemplo abaixo com o qual a aplicação exigirá o preenchimento do campo relativo ao nome da *pessoa*:

```
class Pessoa < ActiveRecord::Base
    ...
    validates_presence_of :nome
```

```
end
```

O Rails permite ainda a utilização da validação *validates_format_of* e *validates_length_of*, que avaliam o formato de algum campo ou o tamanho exigido, respectivamente.

3.4.7 Definir folhas de estilo para a aplicação

Como em qualquer aplicação, para se ter uma agilidade na manutenção do *design*, o Rails permite a utilização de folhas de estilo. Inclusive no código gerado podemos encontrar a utilização de uma folha de estilo padrão do Rails.

Mas, antes de avançarmos nos ajustes e alterações dos códigos para atender as especificações da aplicação proposta, será efetuada a inclusão de uma nova folha de estilo que dará o acabamento desejado.

Para manter o mesmo visual nas aplicações em Rails e Java foi utilizada, basicamente, a mesma folha de estilo conforme segue abaixo:

```
/*
//*******************************************************
*
// Folha de Estilo Para o Controle de Atividades
//
// Cheferson W. Lovatto
//
//*******************************************************
*
*/
body {
    background-color : #ffffff;
}
h1, h2, h3, h4, h5 {
    font-family : Verdana, Geneva, Arial, Helvetica, sans-
serif;
}
h1 {
    font-size: 25px;
}
h2 {
    font-size: 25px;
}
h3 {
    font-size: 20px;
}
p, td, th, br {
    font-family : Verdana, Geneva, Arial, Helvetica, sans-
serif;
    font-size: 13px;
}
a {
    color : blue;
```

```
    font-weight: bold;
}
a:visited {
    color : blue;
    font-weight: bold;
}
a:hover {
    color : #898989;
    background-color : #ffffff;
    font-weight: bold;
}
span.instr {
    color : red;
    font-weight: bold;
}
table.dados {
    background-color : #EBD490;
}
tr.impar {
    background-color : #EBD490;
}
tr.par {
    background-color : #ffffcc;
}
tr.head {
    background-color: #EBD490;
}
.destaca{
    font: bold;
    font-style: italic;
    color: Red;
}
```

Figura 3.2: Código da Folha de Estilo

Para que a aplicação passe a utilizá-la, se deve alterar o layout para que faça menção ao arquivo que contém a folha de estilo. Isso é realizado incluindo no arquivo do layout o comando *<%= stylesheet_link_tag 'estilo' %>*

3.4.8 Ajustar os visualizadores conforme a especificação da aplicação

Por fim se faz necessário efetuar ajustes nos visualizadores para que os mesmos atendam as especificações da aplicação proposta.

Entre os ajustes e alterações que foram realizados, destacam-se:

- Inclusão de *links* para as telas de ajuda;
- Modificação do visual de cada página da aplicação
- Inclusão de *links* para navegação entre as telas de cada modelo e para retorno ao menu inicial;

- Tradução das mensagens enviadas após cada operação da aplicação, pois o Rails as gera em inglês.

4 COMPARANDO OS DOIS AMBIENTES DE DESENVOLVIMENTO

Para efeito de comparação entre os dois ambientes de desenvolvimento estabeleceram-se os critérios conforme relacionado a seguir:

- *Tempo de desenvolvimento*: por se tratar do critério mais importante quando se fala em desenvolvimento de aplicativos. Seus resultados estão diretamente relacionados à simplicidade, poder e flexibilidade da linguagem de cada ambiente;
- *Tamanho do código gerado*: embora muito subjetivo esse critério serve para dar uma idéia da flexibilidade e poder de cada linguagem;
- *Performance na execução*: sempre se espera que um aplicativo tenha uma execução rápida, pois do contrário pode inviabilizar sua utilização;
- *Facilidade de manutenção*: um aplicativo, geralmente, necessita de atualização e assim esse critério é importante para avaliar o grau de facilidade que cada ambiente pode propiciar;
- *Facilidade na obtenção de conteúdo sobre o ambiente*: por se traduzir na difusão de cada ambiente, pois quanto mais se tem facilitado a obtenção de conteúdo, mais se percebe o quanto um ambiente é utilizado.

É importante salientar que as comparações realizadas levam em conta a experiência que o autor tem de desenvolvimento nos dois ambientes.

Para o ambiente Java, o conhecimento atual resume-se a um curso básico, trabalhos elaborados em algumas disciplinas do Curso de Especialização em *Web* e Sistemas de Informação, leitura de livros de programação e pesquisas na *web*, ou seja, experiência de programador iniciante.

Já com relação ao ambiente Ruby on Rails não havia nenhuma experiência anterior até o momento em que tomou-se conhecimento desse ambiente e foi decido por avaliá-lo. A partir desse ponto adquiriu-se conhecimento por meio de leitura de livros relacionados e pesquisa na *web*.

4.1 Tempo de desenvolvimento

Visto que tanto para a aplicação em Java quanto para Ruby on Rails se fez necessária pesquisa de alternativas de solução para implementar as especificações da

aplicação, devido a pouca utilização dos dois ambientes, os tempos dispendidos foram um pouco elevados, conforme mostrado na tabela 4.1.

Tabela 4.1: Tempo de desenvolvimento Java X Ruby on Rails

Java (horas)	Ruby on Rails (horas)
8,5	4,0

Entende-se que, uma vez conhecendo melhor os ambientes de desenvolvimento, o tempo de desenvolvimento reduz-se muito, principalmente no Ruby on Rails, pois para se ter todas as operações para manter as tabelas envolvidas na aplicação proposta foi necessário apenas 5 minutos de desenvolvimento enquanto que o restante do tempo foi necessário para trabalhar o visual (folha de estilo e *layout*) e atender algumas especificações da aplicação (tela de ajuda, montagem de comandos de seleção conforme critérios informados nas telas).

4.2 Tamanho do código gerado

Com relação ao código gerado foi elaborada uma comparação dos dois ambientes considerando o tamanho físico ocupado pelo código gerado, quantidade de caracteres e número de linhas. Essa comparação foi efetuada apenas para se ter idéia da ocupação do código gerado, pois acredita-se que esses apontamentos não devem servir como avaliadores para um ambiente de desenvolvimento.

De qualquer forma, na tabela 4.2 são relacionados os apontamentos levantados nos dois ambientes.

Tabela 4.2: Comparativo do código gerado Java X Ruby on Rails

	Java	Ruby on Rails
Espaço físico (MB)	1,88	1,41
Número de caracteres	788129	253150
Número de linhas	8838	7676

Percebe-se que, para a aplicação proposta, a qual é bem simples, o ambiente Ruby on Rails obteve medições melhores, principalmente quanto ao número de caracteres pelo fato que esse *framework* gera dinamicamente boa parte de seu código conforme a utilização do aplicativo.

No demais, são diferenças que, normalmente, não interferem na escolha de um ambiente para desenvolvimento em ambiente *Web*.

4.3 Performance na execução

Para efeito de comparação dos dois ambientes, avaliou-se o tempo de carga das páginas de cada aplicação, bem como sua ocupação de memória.

Com relação ao tempo de carga das aplicações foi utilizado o *plugin* do *Firefox* conhecido por *Yslow* que é integrado à ferramenta de desenvolvimento *Firebug*, o qual pode ser obtido em *http://developer.yahoo.com/yslow/*.

Essas avaliações foram efetuadas num primeiro momento tendo como carga apenas um processo em cada aplicação e foram observados os resultados mostrados na tabela 4.3.

Tabela 4.3: Performance de Execução com um processo

	Java	Ruby on Rails
Tempo de carga *JavaScript*	79 ms	48 ms
Tempo de carga CSS	141 ms	73 ms
Tempo de carga de página	15 ms	15 ms
Tempo de carga total	235 ms	136 ms
Ocupação de memória	68.252 KB	33.684 KB

Posteriormente, inicializaram-se 3 processos simultâneos para cada aplicação e foram obtidos os seguintes resultados, conforme a tabela 4.4.

Tabela 4.4: Performance de Execução com três processos simultâneos

	Java	Ruby on Rails
Tempo de carga JavaScript	97 ms	44 ms
Tempo de carga CSS	155 ms	61 ms
Tempo de carga de página	47 ms	42 ms
Tempo de carga total	299 ms	147 ms
Ocupação de memória	76.521 KB	39.453 KB

Com as comparações efetuadas, verifica-se um melhor desempenho no ambiente Ruby on Rails uma vez que o mesmo utiliza menos memória e apresenta tempos de carga menores, tanto para o carregamento separado de *javascript* e CSS, quanto para a carga total da página do aplicativo.

4.4 Facilidade de Manutenção

Ao avaliar as aplicações desenvolvidas nos dois ambientes de desenvolvimento, Java e Ruby on Rails, quanto a facilidade de manutenção de seus códigos, percebe-se que, devido a sua estrutura e convenções pré-definidas, o ambiente Ruby on Rails apresenta mais vantagens sobre o ambiente Java. Isso fica evidenciado em alterações do BD e alterações no *layout* de formulários.

No início do desenvolvimento das aplicações propostas, foi utilizado um BD em *Access* e depois alterado para MySQL. Para o ambiente Java foi necessário percorrer todos os programas para alterar o *drive* de acesso ao BD e corrigir a sintaxe de alguns

comandos. Já para o ambiente Ruby on Rails, bastou alterar o nome do BD a ser utilizado no arquivo *config/database.yml* de *access* para *mysql*, visto que a abstração de dados em Ruby é nativa.

Com relação a modificações em formulários, para Java foi necessário que em cada página de alteração e inclusão de uma tabela, essas modificações fossem replicadas, enquanto que para Ruby on Rails bastou alterar o arquivo *_form.html* para que as páginas que utilizem contemplassem as modificações. Caso a alteração fosse no *layout* da aplicação, o trabalho necessário em Java seria maior, pois exigiria que todas as páginas tivessem seu *layout* modificado, o que para Ruby on Rails bastaria alterar o *layout* padrão definido em *app/views/layouts/padrão.rhtml* que é utilizado por todas as páginas.

4.5 Facilidade na obtenção de conteúdo sobre o ambiente

Embora Ruby on Rails tenha sido desenvolvido recentemente, houve facilidade na obtenção de informações sobre seu uso, desenvolvimento de aplicações, dicas e auxílio para a implementação de uma aplicação.

Até mesmo frente a ampla utilização do Java, para o qual se obtém com muita facilidade conteúdo e informações para o desenvolvimento de aplicações, também não se encontram dificuldades para com o ambiente RubynRails e mesmo da linguagem Ruby.

Entretanto, em se tratando de publicações (livros e revistas) que abordem a linguagem Ruby e o *framework* Ruby on Rails, ainda hoje se encontra poucas opções. Mas é importante salientar que as publicações existentes são abrangentes e apresentam uma linguagem de fácil entendimento.

Isso é percebido ao averiguar a bibliografia deste trabalho, onde são listados diversos endereços de informações sobre a linguagem Ruby e o Ruby on Rails contra poucas opções de livros.

4.6 Resumo da Comparação

Abaixo é apresentado um resumo das comparações realizadas entre os dois ambientes de desenvolvimento:

Tabela 4.5: Resumo da Comparação dos Dois Ambientes

Critério	**Java**	**Ruby on Rails**
Tempo de desenvolvimento	Apresenta um bom tempo de desenvolvimento (8,0 h).	Apresenta um tempo de desenvolvimento muito bom (4,5 h).
Tamanho do código	Tamanho: 1,88 MB	Tamanho: 1,41 MB
	Caracteres: 788129	Caracteres: 253150
	Linhas: 8838	Linhas: 7676
Performance de execução Para 1 Processo	Carga total: 235 ms	Carga total: 136 ms
	Ocupação: 68.252 KB	Ocupação: 33.684 KB
Performance de execução Para 3 Processos	Carga total: 299 ms	Carga total: 147 ms
	Ocupação: 76.521 KB	Ocupação: 39.453 KB
Facilidade de manutenção	Apresenta uma facilidade regular na manutenção do aplicativo.	Apresenta uma boa facilidade na manutenção do aplicativo.
Obtenção de conteúdo	Enorme facilidade na obtenção de conteúdo sobre o ambiente.	Enorme facilidade na obtenção de conteúdo sobre o ambiente.

5 CONCLUSÃO

Com este trabalho, se disponibilizou um levantamento dos principais recursos do *framework* Ruby on Rails e assim pôde-se entender sua estrutura e organização para que as pessoas interessadas tenham como decidir pela sua utilização ou não no desenvolvimento de aplicativos para a *web*.

Na avaliação do *framework*, além de abordar aspectos relacionados aos seus recursos, se procurou verificar o crescimento no interesse por desenvolvimento com Ruby on Rails.

Nesse ponto, embora esse *framework* seja muito recente, apontou-se no item 2.4.1 vários IDEs que já estão disponibilizados para o auxílio no desenvolvimento de aplicativos *Web*.

Da mesma forma ficou bem evidenciado que o interesse por sua utilização vem crescendo muito rapidamente, conforme os vários exemplos de sites citados no item 2.3, além dos serviços de hospedagem que já oferecem suporte ao ambiente Ruby on Rails, inclusive em provedores no Brasil, conforme listado no item 2.4.2.

Tudo isso salienta a evolução no crescimento do interesse por esse novo ambiente de desenvolvimento e motivou a sua comparação com o ambiente de desenvolvimento em Java.

Com a comparação realizada, observou-se que, com relação às métricas do código fonte dos aplicativos, os resultados obtidos com Ruby on Rails são um pouco melhores que com Java. Evidentemente, isso não deve ser tomado como parâmetro para se definir por sua utilização ou não, mas evidencia um, possível, melhor desempenho.

Essa evidência com relação ao desempenho ficou demonstrada ao se comparar a performance de execução dos dois ambientes de desenvolvimento, pois o Ruby on Rails se mostrou mais rápido tanto para carga dos componentes (CSS e *javascripts*) individualmente quanto na execução da aplicação como um todo. Além disso, percebeu-se uma melhor utilização da memória do servidor, o que se torna um fator importante a se considerar no momento da escolha de um ambiente para se desenvolver uma aplicação na *Web*.

Outro ponto comparado foi com relação a facilidade de manutenção. Aqui se identificou alguns recursos que fazem com que o Ruby on Rails leve algumas vantagens sobre o ambiente Java, principalmente por que o Rails utiliza muito mais convenções do que configuração. Com isso se tem uma estrutura organizada de forma a exigir um menor esforço para efetuar modificações nas aplicações. Mas é importante frisar que,

como a aplicação proposta é muito simples, não foi possível explorar de forma mais profunda os aspectos inerentes a manutenção das aplicações, requerendo assim que no futuro se proceda um aprofundamento desse tópico.

Isso é necessário mesmo tendo conhecimento que Java disponibilize um ambiente robusto e adequado ao desenvolvimento de aplicações complexas, visto sua ampla utilização, e que se encontre informações no site oficial do Ruby on Rails que afirmam que este também o é preparado e adequado para desenvolvimentos de maior grau de complexidade.

Por fim foi levantado o quanto se pode conseguir de informações e literatura para auxiliar a quem estiver iniciando com o desenvolvimento nesses dois ambientes: Java e Ruby on Rails. Ao se comparar esse tópico percebeu-se claramente que o *framework* Ruby on Rails mesmo sendo tão recente, tem oferecido muita informação sobre seu uso, desenvolvimento de aplicações, dicas e auxílio para a implementação de uma aplicação com a mesma facilidade que se tem com o ambiente Java.

Existem muitos fóruns e comunidades para os dois ambientes, o que se traduz em disseminação de informações, dicas e auxílios no desenvolvimento de aplicações na *Web*. Apenas nota-se que, pelo pouco tempo de existência do ambiente Ruby on Rails, a literatura disponível ainda é pequena, pois todo o processo de editoração é mais demorado e as publicações existentes são na maioria na língua inglesa. Mas mesmo assim essas publicações são bem abrangentes e apresentam textos de fácil entendimento devido as suas linguagens claras e objetivas.

Para finalizar, espera-se que esse trabalho tenha fornecido informações o suficiente que possam orientar os desenvolvedores de aplicativos na *Web* na definição pelo uso ou não do Rails e que sirva de um ponto de partida para novas e mais profundas comparações, quer seja com Java ou com outro ambiente de desenvolvimento que acharem pertinente.

REFERÊNCIAS

HIBBS, C.; TATE, B. A. **Ruby on Rails - Executando.** Rio de Janeiro: Alta Books, 2006.

OLIVEIRA JUNIOR, E. R. de. **Quer Aprender Ruby?** Disponível em:< http://www.eustaquiorangel.com/downloads/tutorialruby.pdf >. Acesso em: jun. 2007.

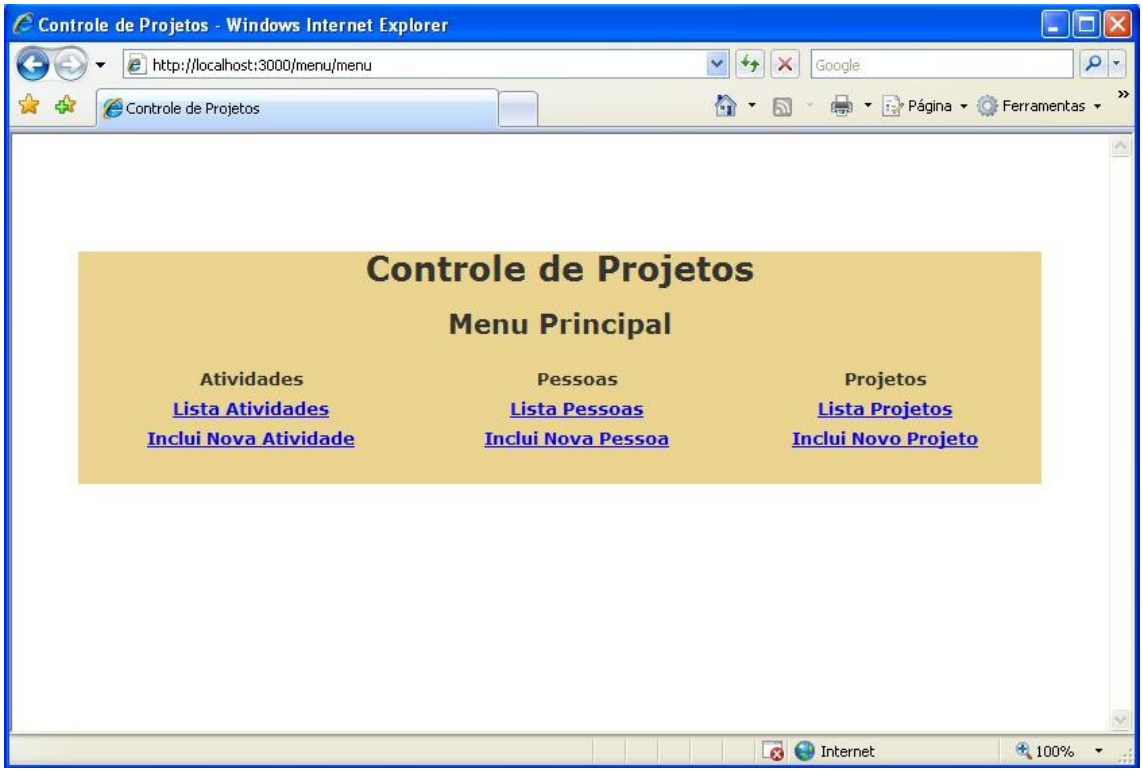
OLIVEIRA JUNIOR, E. R. de. **Quer Aprender Rails?** Disponível em:< http://www.eustaquiorangel.com/downloads/tutorialrails.pdf >.Acesso em: jun. 2007.

OLIVEIRA JUNIOR, E. R. de. **Ruby Conhecendo a Linguagem.** Rio de Janeiro: Ed. Brasport, 2006.

RUBYONBR. **Instalando Ruby.** Disponível em: < http://www.rubyonbr.org/articles/2006/08/24/instalando-ruby/ >. Acesso em: jun. 2007.

RUBYONBR. **Instalando Rails.** Disponível em: <http://www.rubyonbr.org/articles/2006/08/24/instalando-rails/ >.Acesso em: jun. 2007.

ANEXO A MENU - RUBY ON RAILS



Controller menu_controller.rb

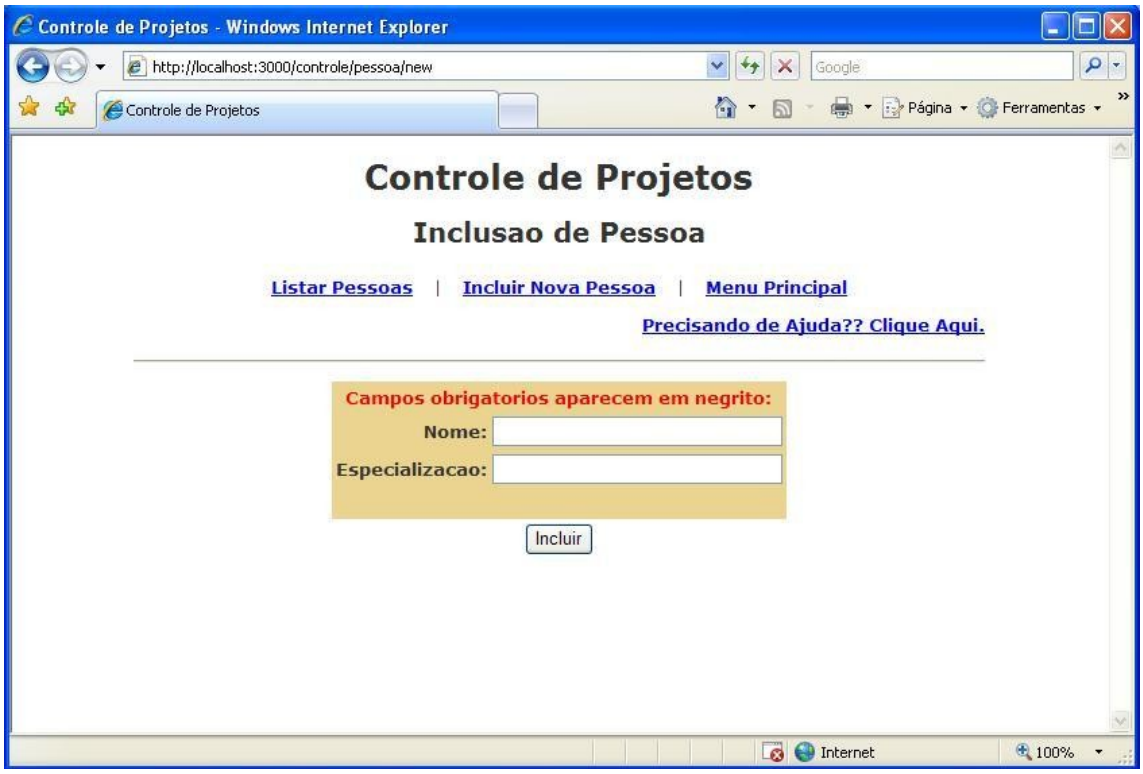
```
class MenuController < ApplicationController
  layout "menu"
  def index
      menu
      render :action => 'menu'
  end
end
```

View menu.rhtml

```
<br><br><br><br>
<table width="90%" align="center" class="dados">
  <tr>
    <td align="center" colspan="3"><h2>Controle de Projetos</h2></td>
  </tr>
  <tr>
    <td align="center" colspan="3"><h3>Menu Principal</h3></td>
  </tr>
```

```
  <tr>
    <td align="center"><strong>Atividades</strong></td>
    <td align="center"><strong>Pessoas</strong></td>
    <td align="center"><strong>Projetos</strong></td>
  </tr>
  <tr>
    <td align="center"><%= link_to 'Lista Atividades',
:controller=>"controle/atividade", :action=>'list' %></td>
    <td align="center"><%= link_to 'Lista Pessoas',
:controller=>"controle/pessoa", :action=>'list' %></td>
    <td align="center"><%= link_to 'Lista Projetos',
:controller=>"controle/projeto", :action=>'list' %></td>
  </tr>
  <tr>
    <td align="center"><%= link_to 'Inclui Nova Atividade',
:controller=>"controle/atividade", :action=>'new' %></td>
    <td align="center"><%= link_to 'Inclui Nova Pessoa',
:controller=>"controle/pessoa", :action=>'new' %></td>
    <td align="center"><%= link_to 'Inclui Novo Projeto',
:controller=>"controle/projeto", :action=>'new' %></td>
  </tr>
  <tr>
    <td align="center" colspan="3"> </td>
  </tr>
</table>
```

ANEXO B INCLUSÃO DE PESSOA - RUBY ON RAILS



Controller pessoa_controller.rb

```
class Controle::PessoaController < ApplicationController

  layout "padrao"

  def index
    list
    render :action => 'list'
  end

  # GETs should be safe (see
http://www.w3.org/2001/tag/doc/whenToUseGet.html)
  verify :method => :post, :only => [ :destroy, :create, :update ],
         :redirect_to => { :action => :list }

  def list
    if request.post?
      @strnome = params[:pessoa][:nome]
      @strespe = params[:pessoa][:especializacao]
```

```
    else
      @strnome = "X1r5"
      @strespe = "X1r5"
    end
    @pessoas = Pessoa.find(:all,:conditions=>["nome like ? and
especializacao like ?","%#{@strnome}%","%#{@strespe}%"])
#    @pessoa_pages, @pessoas = paginate :pessoas, :per_page => 3
  end

  def show
    @pessoa = Pessoa.find(params[:id])
  end

  def new
    @pessoa = Pessoa.new
  end

  def create
    @pessoa = Pessoa.new(params[:pessoa])
    if @pessoa.save
      flash[:notice] = 'Inclusao de pessoa efetuada com sucesso.'
      redirect_to :action => 'new'
    else
      render :action => 'new'
    end
  end

  def edit
    @pessoa = Pessoa.find(params[:id])
  end

  def update
    @pessoa = Pessoa.find(params[:id])
    if @pessoa.update_attributes(params[:pessoa])
      flash[:notice] = 'Alteracao de pessoa efetuada com sucesso.'
      redirect_to :action => 'list', :id => @pessoa
    else
      render :action => 'edit'
    end
  end

  def destroy
    Pessoa.find(params[:id]).destroy
    redirect_to :action => 'list'
  end
end
```

Model pessoa.rhtml

```
class Pessoa < ActiveRecord::Base
  validates_presence_of :nome
  validates_presence_of :especializacao
  has_many :atividades
end
```

View new.rhtml

```
<table width="90%" align="center">
  <tr>
    <td align="center">
      <% form_tag :action => 'create' do %>
```

```
      <%= render :partial => 'form' %>
      <%= submit_tag "Incluir" %>
      <% end %>
    </td>
  </tr>
</table>
```

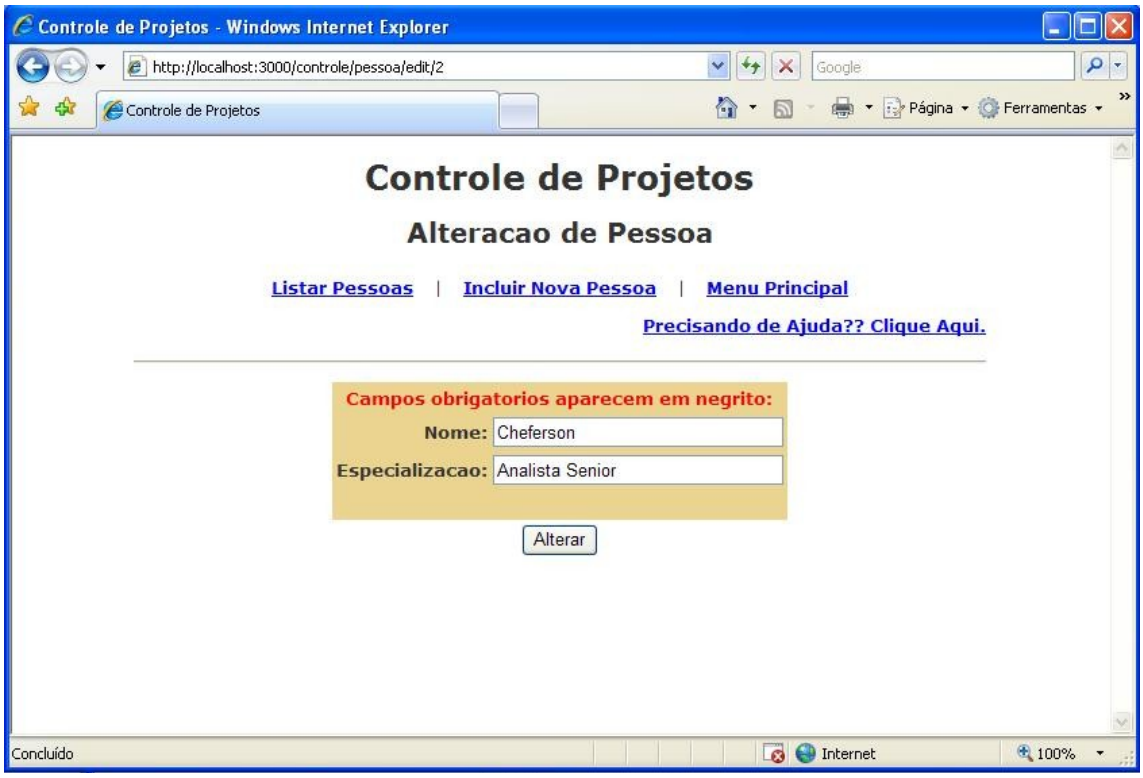
ANEXO C LISTAGEM DE PESSOA - RUBY ON RAILS

View list.rhtml

```
<table width="90%" align="center">
    <tr>
        <td align="center">
        <% form_tag :action => 'list', :id => @pessoa do %>
            <%= render :partial => 'form' %>
            <%= submit_tag 'Listar' %>
        <% end %>
        </td>
    </tr>
    <% if @pessoas %>
    <tr>
        <td align="center">
            <table>
                <tr>
                    <th>Nome</th>
                    <th>Especializacao</th>
                </tr>
```

```
                <% for pessoa in @pessoas %>
                <tr>
                    <td align="left"><a href="edit/<%=h pessoa.id
%>"><%=h pessoa.nome %></a></td>
                    <td align="left"><%=h pessoa.especializacao
%></td>
                    <td><%= link_to 'Excluir', { :action => 'destroy',
:id => pessoa }, :confirm => 'Voce tem certeza?', :method => :post
%></td>
                </tr>
                <% end %>
            </table>
        </td>
    </tr>
    <% end %>
</table>
```

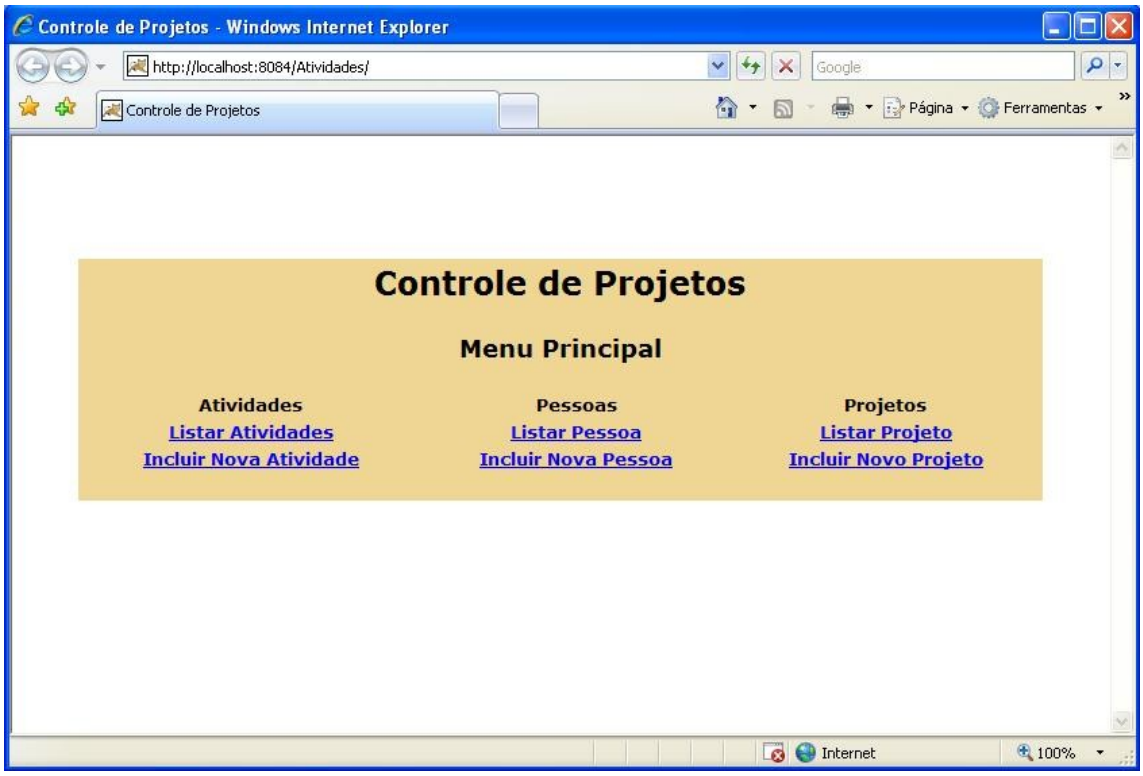
ANEXO D ALTERAÇÃO DE PESSOA - RUBY ON RAILS



View edit.rhtml

```
<table width="90%" align="center">
  <tr>
    <td align="center">
      <% form_tag :action => 'update', :id => @pessoa do %>
        <%= render :partial => 'form' %>
        <%= submit_tag 'Alterar' %>
      <% end %>
    </td>
  </tr>
</table>
```

ANEXO E MENU - JAVA



Classe Controle.java

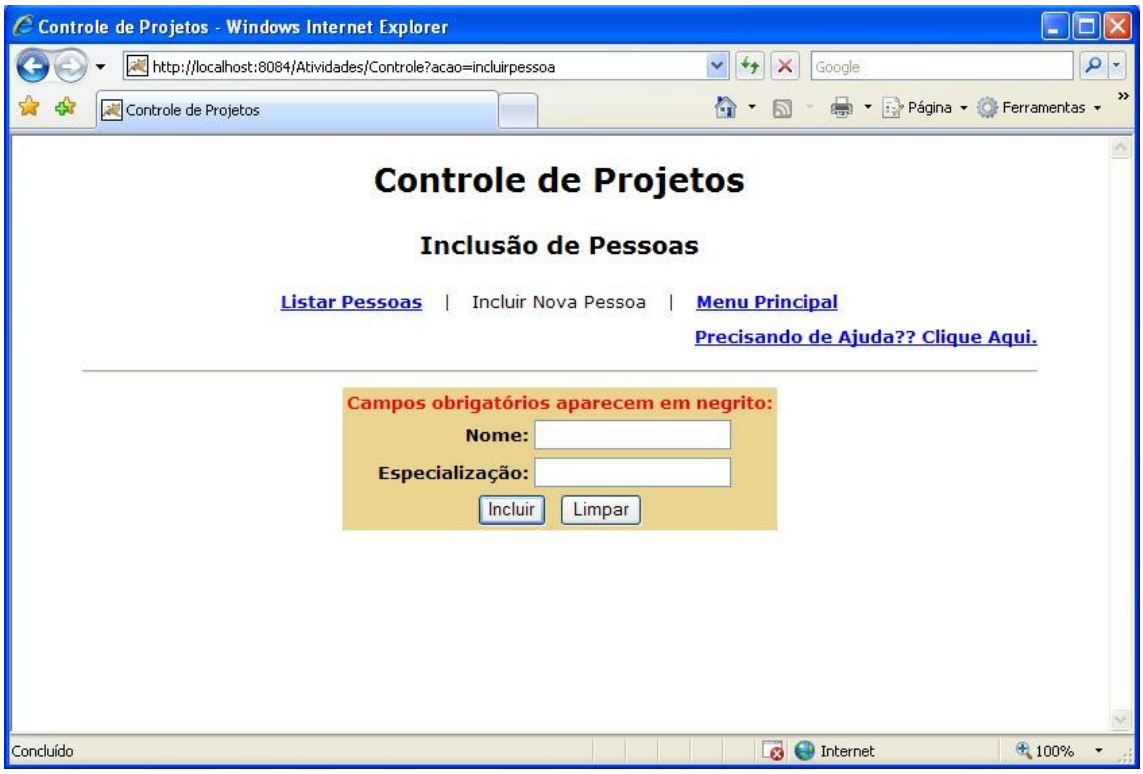
```
//************************************************************
// Classe Controle                                           //
// Cheferson W. Lovatto                                      //
//************************************************************

package controle;

import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;

public class Controle extends HttpServlet {

        public void doGet(HttpServletRequest request,
HttpServletResponse response)
                  throws ServletException, IOException {

            String acao = request.getParameter("acao");
```

```
            if (acao == null) {
                  acao = "vazio";
            }

            // Listar Atividades
            if (acao.equals("listaratividade")) {
                  irParaPagina("/listaratividade.jsp", request,
response);
                  // Incluir Atividades
            } else if (acao.equals("incluiratividade")) {
                  irParaPagina("/incluiratividade.jsp", request,
response);
                  // Alterar Atividades
            } else if (acao.equals("alteraratividade")) {
                  irParaPagina("/alteraratividade.jsp", request,
response);
                        // Listar Pessoas
                } else if (acao.equals("listarpessoa")) {
                  irParaPagina("/listarpessoa.jsp", request,
response);
                  // Incluir Pessoas
            } else if (acao.equals("incluirpessoa")) {
                  irParaPagina("/incluirpessoa.jsp", request,
response);
                  // Alterar Pessoas
            } else if (acao.equals("alterarpessoa")) {
                  irParaPagina("/alterarpessoa.jsp", request,
response);
                        // Listar Projetos
                } else if (acao.equals("listarprojeto")) {
                  irParaPagina("/listarprojeto.jsp", request,
response);
                  // Incluir Projetos
            } else if (acao.equals("incluirprojeto")) {
                  irParaPagina("/incluirprojeto.jsp", request,
response);
                  // Alterar Projetos
            } else if (acao.equals("alterarprojeto")) {
                  irParaPagina("/alterarprojeto.jsp", request,
response);
                  // Op&ccedil;&atilde;o inv&aacute;lida
            } else {
                  irParaPagina("/index.jsp", request, response);
            }
      }

      private void irParaPagina(String address, HttpServletRequest
request,
                  HttpServletResponse response) throws
ServletException, IOException {

            RequestDispatcher dispatcher = getServletContext()
                        .getRequestDispatcher(address);
            dispatcher.forward(request, response);
      }

}
```

Página index.jsp

```
<%
//************************************************************
// Tela Inicial do Controle de Atividades                    //
// Cheferson W. Lovatto                                      //
//************************************************************
%>

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
    <head>
        <link rel="stylesheet" type="text/css" href="estilo.css">
        <title>Controle de Projetos</title>
    </head>
<body>
    <br><br><br><br>
    <table width="90%" align="center" class="dados">
        <tr>
            <td align="center" colspan="3"><h2>Controle de
Projetos</h2></td>
        </tr>
        <tr>
            <td align="center" colspan="3"><h3>Menu
Principal</h3></td>
        </tr>
        <tr>
            <td align="center"><strong>Atividades</strong></td>
            <td align="center"><strong>Pessoas</strong></td>
            <td align="center"><strong>Projetos</strong></td>
        </tr>
        <tr>
            <td align="center"><a
href="Controle?acao=listaratividade">Listar Atividades</a></td>
            <td align="center"><a
href="Controle?acao=listarpessoa">Listar Pessoa</a></td>
            <td align="center"><a
href="Controle?acao=listarprojeto">Listar Projeto</a></td>
        </tr>
        <tr>
            <td align="center"><a
href="Controle?acao=incluiratividade">Incluir Nova Atividade</a></td>
            <td align="center"><a
href="Controle?acao=incluirpessoa">Incluir Nova Pessoa</a></td>
            <td align="center"><a
href="Controle?acao=incluirprojeto">Incluir Novo Projeto</a></td>
        </tr>
        <tr>
            <td align="center" colspan="3"> </td>
        </tr>
    </table>
</body>
</html>
```

ANEXO F INCLUSÃO DE PESSOA - JAVA



Classe IncluiPessoa.java

```
//************************************************************
// Classe IncluiPessoas                                      //
// Cheferson W. Lovatto                                      //
//************************************************************

package controle;

import java.sql.*;
import java.text.*;

public class IncluiPessoas {

    public String strNome;
    public String strEspecializacao;
    private String reqSubmit;

    public IncluiPessoas() {
    }
```

```
    private String isNull(String param) {
        if (param == null)
            return "";
        else
            return param;
    }

    public boolean incluir() throws SQLException {
        try {
            // Conexão para ODBC
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/atividade","","");

            Statement stmt = con.createStatement();

            String strSQL = "insert into pessoas (nome,
especializacao)" +
                " values ('"+ getNome()+"' , '"+getEspecializacao()+
"')";

            // Executa o comando SQL
            System.out.println("*** Debug *** " + strSQL);
            stmt.executeUpdate(strSQL);

            con.close();
            con = null;

            stmt.close();
            stmt = null;
        }
        catch (Exception ex) {
            System.out.println("Erro SQL: " + ex.getMessage());
            return(false);
        }

        return(true);
    }

    public String getNome() {
        return strNome;
    }

    public String getEspecializacao() {
        return strEspecializacao;
    }

    public String getSubmit() {
        return reqSubmit;
    }

    public void setNome(String nome) {
        strNome = isNull(nome);
    }

    public void setEspecializacao(String especializacao) {
        strEspecializacao = isNull(especializacao);
    }
```

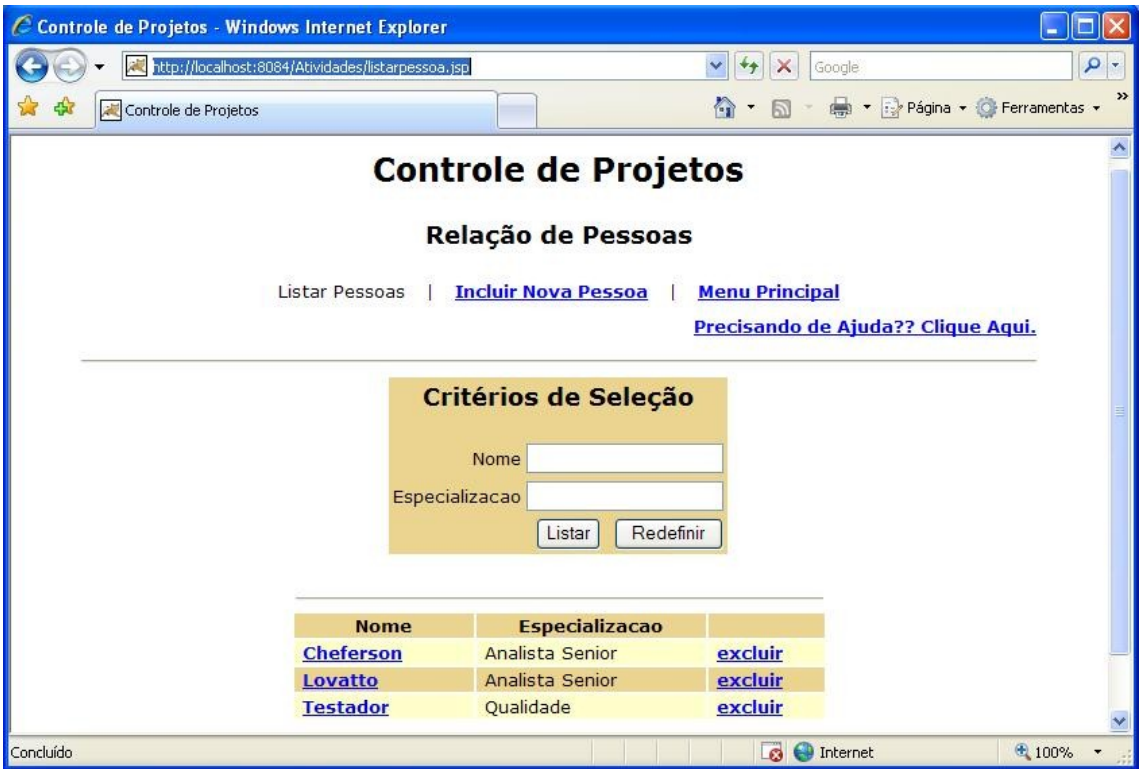
```
    public void setSubmit(String submit) {
        reqSubmit = submit;
    }

}
```

Página incluirpessoa.jsp

```
<%
//************************************************************
// Inclusão de Pessoa                                        //
// Cheferson W. Lovatto                                      //
//************************************************************
%>

<%@ page import="java.sql.*" %>
<%@ page buffer="none" %>
<%@ page import = "controle.*"%>
<jsp:useBean id="IncluiPessoas" class="controle.IncluiPessoas"
scope="request" type="IncluiPessoas"/>
<jsp:setProperty name="IncluiPessoas" property="*"/>
<%
String acao= "";
String nome = "";
String especializacao = "";
String submit = null;
%>

<html>
<head>
    <title>Controle de Projetos</title>
    <link rel="stylesheet" type="text/css" href="estilo.css" />
    <script type="text/javascript" language="JavaScript1.2"
src="valida.js"></script>
</head>

<body onload="formulario.nome.select(); formulario.nome.focus();">
<table width="90%" align="center">
    <tr>
        <td align="center"><h2>Controle de Projetos</h2></td>
    </tr>
    <tr>
        <td align="center"><h3>Inclusão de Pessoas</h3></td>
    </tr>
    <tr>
        <td align="center">
            <A href="Controle?acao=listarpessoa">Listar Pessoas</A>
              |  
           Incluir Nova Pessoa
              |  
            <A href="Controle?acao=menu">Menu Principal</A>
        </td>
    </tr>
    <TR>
        <TD align="right" height="30" VALIGN="middle"><a
href="javascript:abreJanela('ajudaincluipes.html','400','370');">Preci
sando de Ajuda?? Clique Aqui.</a></TD>
    </TR>
    <tr>
        <td><hr></td>
```

```
    </tr>
    <% if (IncluiPessoas.getSubmit() != null ) {
    if( IncluiPessoas.incluir() ){  %>
    <tr><td align="center">Pessoa <strong><%= IncluiPessoas.getNome()
%></strong> incluída com sucesso</td></tr>
    <% } else { %>
    <tr><td align="center"><p>Ocorreu erro na inclusão de
pessoa!</p></td></tr>
    <% }
    } %>
    <tr>
        <td align="center">
            <form method="POST" action="incluirpessoa.jsp"
name="formulario" onsubmit="return fValidaFormPes();">
                <table class="dados">
                    <tr>
                        <td colspan="2" align="center"><span
class="instr">Campos obrigatórios aparecem em negrito:</span></td>
                    </tr>
                    <tr>
                        <td  align="right"><strong>Nome:</strong></td>
                        <td><input type="text" name="nome"
value=""></td>
                    </tr>
                    <tr>
                        <td
      align="right"><strong>Especialização:</strong></td>
                        <td><input type=text name="especializacao"
value=""></td>
                    </tr>
                    <tr>
                        <td colspan=2 align="center">
                            <input type="Submit" name="submit"
value="Incluir">  <input type="Reset" name="limpar"
value="Limpar">
                        </td>
                    </tr>
                </table>
            </form>
        </td>
    </tr>
</table>
</body>

</html>
```

ANEXO G LISTAGEM DE PESSOA - JAVA



Classe ListaPessoa.java

```
//*************************************************************
// Classe ListaPessoas                                        //
// Cheferson W. Lovatto                                       //
//*************************************************************

package controle;

import java.sql.*;
import java.text.*;

public class ListaPessoas {

      private String reqId;
      private String reqNome;
      private String reqEspecializacao;
      private String reqSubmit;
      private ResultSet rs;
```

```
	public ListaPessoas() {
	}

	private String isNull(String param) {
		if (param == null)
			return "";
		else
			return param;
	}

	public boolean fazerQuery() {
		try {

				// Conex&atilde;o para ODBC
		Class.forName("com.mysql.jdbc.Driver");
		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/atividade","","");
			String table = "pessoas";

			Statement stmt = con.createStatement();
			String strSQL;

			// Monta o comando SQL
			strSQL = "select * from " + table + " where
('1'='1'";

			if (reqNome != null) {
				strSQL = strSQL + " and nome like '%" + reqNome
+ "%'";
			}
			if (reqEspecializacao != null) {
				strSQL = strSQL + " and especializacao like '%"
+ reqEspecializacao + "%'";
			}
			strSQL = strSQL + ") order by nome";

			// Para debug do string de query SQL - imprime em
			// logs/stdout_aaaammdd.log
			System.out.println(strSQL);

			// Executa o SQL, criando o recordset
			rs = stmt.executeQuery(strSQL);

		} catch (Exception ex) {
			System.out.println("Erro SQL: " + ex.getMessage());
			return (false);
		}

		return (true);
	}

	public boolean proximaPessoa() {

		try {
			if (!rs.next()) {
				return (false);
			} else {
				reqId = rs.getString("id");
				reqNome = rs.getString("nome");
```

```
                        reqEspecializacao =
rs.getString("especializacao");
                        return (true);
                  }
            } catch (Exception ex) {
                  System.out.println("Erro SQL: " + ex.getMessage());
                  return (false);
            }
      }

      public String getId() {
            return reqId;
      }

      public String getNome() {
            return reqNome;
      }

      public String getEspecializacao() {
            return reqEspecializacao;
      }

      public String getSubmit() {
            return reqSubmit;
      }

      public void setId(String id) {
            reqId = isNull(id);
      }

      public void setNome(String nome) {
            reqNome = isNull(nome);
      }

      public void setEspecializacao(String especializacao) {
            reqEspecializacao = isNull(especializacao);
      }

      public void setSubmit(String submit) {
            reqSubmit = submit;
      }

}
```

Página listarpessoa.jsp

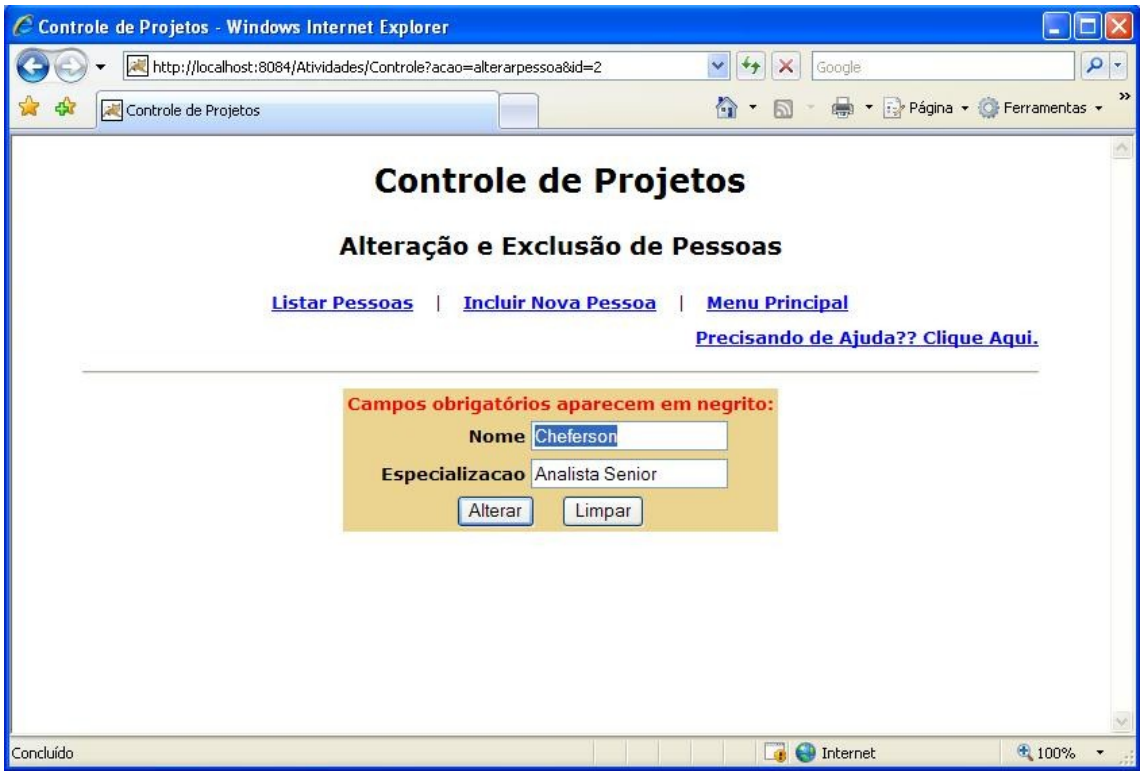
```
<%
//************************************************************
// Relação de Pessoas                                        //
// Cheferson W. Lovatto                                      //
//************************************************************
%>

<%@ page import="java.sql.*" %>
<%@ page buffer="none" %>
<%@ page import = "controle.*"%>
<jsp:useBean id="ListaPessoas" class="controle.ListaPessoas"
scope="request" type="ListaPessoas"/>
<jsp:setProperty name="ListaPessoas" property="*"/>
<%
```

```
boolean impar = true;
String acao= "";
String nome = "";
String especializacao = "";
String submit = null;

// Se acao for "", entao o script foi chamado diretamente.
// Caso contrario, o script foi chamado atraves de um submit
// desta ou de outra pagina

if (ListaPessoas.getSubmit() == null ) {

    nome = "";
    especializacao = "";
} else { // botao "Listar" clicado na pagina anterior, obtem dados do
Form

    if (!ListaPessoas.fazerQuery()){  %>
<p><br /><br />Erro no acesso ao banco de dados<br /><br /></p>
<% }
}

%>
<%-- --------- Inicio do codigo HTML ----------- --%>
<html>
<head>
    <title>Controle de Projetos</title>
    <link rel="stylesheet" type="text/css" href="estilo.css" />
    <script type="text/javascript" language="JavaScript1.2"
src="valida.js"></script>
</head>
<body onload="formulario.nome.select(); formulario.nome.focus();">
<table width="90%" align="center">
    <tr>
        <td align="center"><h2>Controle de Projetos</h2></td>
    </tr>
    <tr>
        <td align="center"><h3>Relação de Pessoas</h3></td>
    </tr>
    <tr>
        <td align="center">
            Listar Pessoas
              |  
            <A href="Controle?acao=incluirpessoa">Incluir Nova
Pessoa</A>
              |  
            <A href="Controle?acao=menu">Menu Principal</A>
        </td>
    </tr>
    <TR>
        <TD align="right" height="30" valign="middle"><a
href="javascript:abreJanela('ajudalistapes.html','400','430');">Precis
ando de Ajuda?? Clique Aqui.</a></TD>
    </TR>
    <tr>
        <td><hr></td>
    </tr>
    <tr>
        <td align="center">
```

```
            <form action="listarpessoa.jsp" name="formulario"
method="POST">
                <table class="dados">
                    <tr>
                        <td colspan=2 align="center">
                            <h3>Critérios de Seleção</h3>
                        </td>
                    </tr>
                    <tr>
                        <td align="right">Nome</td>
                        <td><input type=text name="nome"
value="<%=nome%>"></td>
                    </tr>
                    <tr>
                        <td  align="right">Especializacao</td>
                        <td><input name="especializacao"
value="<%=especializacao%>"></td>
                    </tr>
                    <tr>
                        <td colspan=2 align="right">
                            <input type="Submit" name="submit"
value="Listar">  <input type="Reset">
                        </td>
                    </tr>
                </table>
            </form>
        </td>
    </tr>
</table>

<%
if (ListaPessoas.getSubmit() != null ) {
%>

<!-- Gera os titulos da tabela-->
<table border="0" width="50%" align="center">
    <tr>
        <td colspan="3"><hr></td>
    </tr>
    <tr class="head">
        <th>Nome</th>
        <th>Especializacao</th>
        <th> </th>
    </tr>
    <%
    // Gera as linhas da tabela para cada linha do recordset
    while (ListaPessoas.proximaPessoa() ) {
        // Pinta o zebrado do fundo das linhas
        if (impar = !impar) {
            out.println("<tr class=\"impar\">");
        } else {
            out.println("<tr class=\"par\">");
        }
    %>
    <td> <A href='Controle?acao=alterarpessoa&id=<%=
ListaPessoas.getId() %>'><%= ListaPessoas.getNome() %></a> </td>
    <td> <%= ListaPessoas.getEspecializacao() %> </td>
    <td> <A href='Controle?acao=alterarpessoa&id=<%=
ListaPessoas.getId() %>'>excluir</td>
    </tr>
```

```
    <%
    }
    %>
</table>
<%
}
%>

<table width="90%" align="center">
    <tr>
        <td><hr></td>
    </tr>
    <tr>
        <td align="center">
            Listar Pessoas
              |  
            <A href="Controle?acao=incluirpessoa">Incluir Nova
Pessoa</A>
              |  
            <A href="Controle?acao=menu">Menu Principal</A>
        </td>
    </tr>
</table>
</body>
</html>
```

ANEXO H ALTERAÇÃO DE PESSOA - JAVA



Classe AlteraPessoa.java

```
//************************************************************
// Classe AlteraPessoas                                      //
// Cheferson W. Lovatto                                      //
//************************************************************

package controle;

import java.sql.*;
import java.text.*;

public class AlteraPessoas {

    public String strId;
    public String strNome;
    public String strEspecializacao;
    private String strSubmit;
    private ResultSet rs;
```

```
    public AlteraPessoas() {
    }

    private String isNull(String param) {
        if (param == null)
            return "";
        else
            return param;
    }

    public boolean consultar(String id) {
        try {
            // Conexão para ODBC
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/atividade","","");

            Statement stmt = con.createStatement();

            rs = stmt.executeQuery("SELECT * FROM pessoas WHERE id=" +
id);

            proximaPessoa();

        } catch (Exception ex) {
            System.out.println("Erro SQL: " + ex.getMessage());
            return(false);
        }

        return(true);
    }

    public boolean proximaPessoa() {

        try {
            if (!rs.next()) {
                return (false);
            } else {
                strId             = rs.getString("id");
                strNome           = rs.getString("nome");
                strEspecializacao = rs.getString("especializacao");
                return (true);
            }
        } catch (Exception ex) {
            System.out.println("Erro SQL: " + ex.getMessage());
            return(false);
        }
    }

    public boolean alterar(String id) {
        try {
            // Conexão para ODBC
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/atividade","","");

            Statement stmt = con.createStatement();

            String strSQL = "update pessoas set" +
                    " nome='"+ getNome()+"', " +
```

```
                    " especializacao='" + getEspecializacao()+"'" +
                    " WHERE id=" + id;

            // Executa o comando SQL
            System.out.println("*** Debug *** " + strSQL);
            stmt.executeUpdate(strSQL);

            con.close();
            con = null;

            stmt.close();
            stmt = null;
        } catch (Exception ex) {
            System.out.println("Erro SQL: " + ex.getMessage());
            return(false);
        }

        return(true);
    }

    public boolean excluir(String id) {
        try {
            // Conexão para ODBC
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/atividade","","");

            Statement stmt = con.createStatement();

            String strSQL = "delete from pessoas " +
                    " where id=" + id;

            // Executa o comando SQL
            System.out.println("*** Debug *** " + strSQL);
            stmt.executeUpdate(strSQL);

            con.close();
            con = null;

            stmt.close();
            stmt = null;
        } catch (Exception ex) {
            System.out.println("Erro SQL: " + ex.getMessage());
            return(false);
        }

        return(true);
    }

    public String getId() {
        return strId;
    }

    public String getNome() {
        return strNome;
    }

    public String getEspecializacao() {
        return strEspecializacao;
    }
```

```
    public String getSubmit() {
        return strSubmit;
    }

    public void setID(String id) {
        strId = isNull(id);
    }

    public void setNome(String nome) {
        strNome = isNull(nome);
    }

    public void setEspecializacao(String especializacao) {
        strEspecializacao = isNull(especializacao);
    }

    public void setSubmit(String submit) {
        System.out.println("*** Debug *** " + submit);
        strSubmit = isNull(submit);
    }

}
```

Página alterarpessoa.jsp

```
<%
//************************************************************
// Alteração e Exclusão de Pessoas                           //
// Cheferson W. Lovatto                                      //
//************************************************************
%>

<%@ page import="java.sql.*" %>
<%@ page buffer="none" %>
<%@ page import = "controle.*"%>
<jsp:useBean id="AlteraPessoas" class="controle.AlteraPessoas"
scope="request" type="AlteraPessoas"/>
<jsp:setProperty name="AlteraPessoas" property="*"/>
<%
String acao= "";
String nome = "";
String especializacao = "";
String submit = null;
%>

<html>
<head>
    <title>Controle de Projetos</title>
    <link rel="stylesheet" type="text/css" href="estilo.css" />
    <script type="text/javascript" language="JavaScript1.2"
src="valida.js"></script>
</head>

<body onload="formulario.nome.select(); formulario.nome.focus();">
<table width="90%" align="center">
    <tr>
        <td align="center"><h2>Controle de Projetos</h2></td>
    </tr>
    <tr>
```

```
        <td align="center"><h3>Alteração e Exclusão de
Pessoas</h3></td>
    </tr>
    <tr>
        <td align="center">
            <A href="Controle?acao=listarpessoa">Listar Pessoas</A>
              |  
            <A href="Controle?acao=incluirpessoa">Incluir Nova
Pessoa</A>
              |  
            <A href="Controle?acao=menu">Menu Principal</A>
        </td>
    </tr>
    <TR>
        <TD align="right" height="30" VALIGN="middle"><a
href="javascript:abreJanela('ajudaalterapes.html','400','420');">Preci
sando de Ajuda?? Clique Aqui.</a></TD>
    </TR>
    <tr>
        <td><hr></td>
    </tr>
    <% if("Excluir".equals(request.getParameter("submit"))) {
        if (AlteraPessoas.excluir(request.getParameter("id"))) { %>
            <tr><td align="center">Pessoa <strong><%=
AlteraPessoas.getNome() %></strong> excluida com sucesso</td></tr>
        <% } else { %>
           <tr><td align="center"><p>Ocorreu erro na
exclusão!</p></td></tr>
        <% }
    } else {
        if ("Alterar".equals(request.getParameter("submit"))) {
            if( AlteraPessoas.alterar((request.getParameter("id")))) {
%>
                <tr><td align="center"><p>Pessoa <strong><%=
AlteraPessoas.getNome() %></strong> alterada com sucesso</p></td></tr>
            <% } else { %>
                <tr><td align="center"><p>Ocorreu erro na
alteração!</p></td></tr>
            <% }
        }
    }

    if(!"Excluir".equals(request.getParameter("submit"))) {
        if( !AlteraPessoas.consultar(request.getParameter("id")) )
{  %>
        <tr><td align="center"><p>Erro no acesso ao banco de
dados</p></td></tr>
        <% }
    } else {
        AlteraPessoas.setNome("");
        AlteraPessoas.setEspecializacao("");
        AlteraPessoas.setSubmit(null);
    } %>
    <tr>
        <td align="center">
            <form method="POST" action="alterarpessoa.jsp?id=<%=
request.getParameter("id")%>" name="formulario" onsubmit="return
fValidaFormPes();">
            <table class="dados">
                <tr>
```

```
                    <td colspan="2" align="center"><span
class="instr">Campos obrigatórios aparecem em negrito:</span></td>
                </tr>
                <tr>
                    <td align="right"><strong>Nome</strong></td>
                    <td><input type=text name="nome" value="<%=
AlteraPessoas.getNome() %>"></td>
                </tr>
                <tr>
                    <td
      align="right"><strong>Especializacao</strong></td>
                    <td><input type="text" name="especializacao"
value="<%= AlteraPessoas.getEspecializacao() %>"></td>
                </tr>
                <tr>
                    <td colspan=2 align="center">
                        <input type="submit" name="submit"
value="Alterar">   
                        <input type="reset" name="reset"
value="Limpar">   
<!--                        <input type="submit" name="submit"
value="Excluir">-->
                    </td>
                </tr>
                </table>
            </form>
        </td>
    </tr>
</table>
</body>
</html>
```