UMA BIBLIOTECA VIRTUAL UTILIZANDO RUBY ON RAILS

Thiago Bandeira[1] <thiagobandeirars@gmail.com>
Thiago Dorneles[2] <thiagodornelesrs@gmail.com>
Vagner Martins[3] <vagner.mmartins@gmail.com>

Universidade Luterana do Brasil (Ulbra) – Curso de Ciência da Computação – Câmpus Gravataí
Av. Itacolomi, 3.600 – Bairro São Vicente – CEP 94170-240 – Gravataí - RS

04 de Julho de 2012

RESUMO

Este artigo descreve o processo de desenvolvimento de uma aplicação para web sob o framework Ruby On Rails.

Palavras-chave: Ruby; On Rails; Web; Framework.

ABSTRACT

Title: *"A library online using Ruby On Rails"*

This article describes the process of developing a web application framework under the Ruby On Rails.

Key-words: *Ruby; On Rails; Web.*

INTRODUÇÃO

O processo de criação de um sistema envolve várias etapas. Uma delas é a implementação, em código de programação do sistema. Crescendo a cada dia, em termos de variedade e recursos, os frameworks são um conjunto de ferramentas responsáveis por gerenciar os recursos de uma processo de desenvolvimento de uma aplicação. O Rails é um framework de desenvolvimento web escrito na linguagem Ruby, ele oferece uma alta produtividade ao programador, tornando o desenvolvimento de aplicações web mais agéis.

Este artigo apresenta o processo de desenvolvimento de uma biblioteca virtual utilizando o framework Ruby On Rails com um controlador de versão web, GitHub.

Serão aplicados diversos recursos do framework para o desenvolvimento da aplicação web, e também serão apresentadas as dificuldades e problemas encontrados em todo o processo.

O SISTEMA

A Biblioteca virtual será um sistema simples onde os usuários poderão fazer cadastros de grupos, editoras, livros, novos usuários e efetuar locações de livros. O sistema possuirá dois layouts, um para a administração e outro para os usuários efeturarem locações.

1 Regras de negócio

Em seguida, serão descritas as regras de negócio.

- O sistema deve possuir um grupo de administradores, com ID igual a 1, que será responsável pelo gerenciamento da aplicação, como manter grupos, categorias, livros, editoras, usuários e locações;

1 Aluno da disciplina de Linguagem de Programação para WEB do curso de Ciência da Computação na Ulbra Gravataí..

2 Aluno da disciplina de Linguagem de Programação para WEB do curso de Ciência da Computação na Ulbra Gravataí.

3 Aluno da disciplina de Linguagem de Programação para WEB do curso de Ciência da Computação na Ulbra Gravataí.

- Os usuários comum, poderão efetuar somente locações;
- Não será permitida locação de dois livros de mesmo título e autor para uma mesma locação;
- A multa será definida pelo grupo a qual o usuário pertence;
- Não será permitida renovação de livros, para isso, deve-se entregar o livro locado e fazer uma nova locação.

2 Diagrama de Classes

Na Figura 1, é apresentado o diagrama de classes da aplicação.

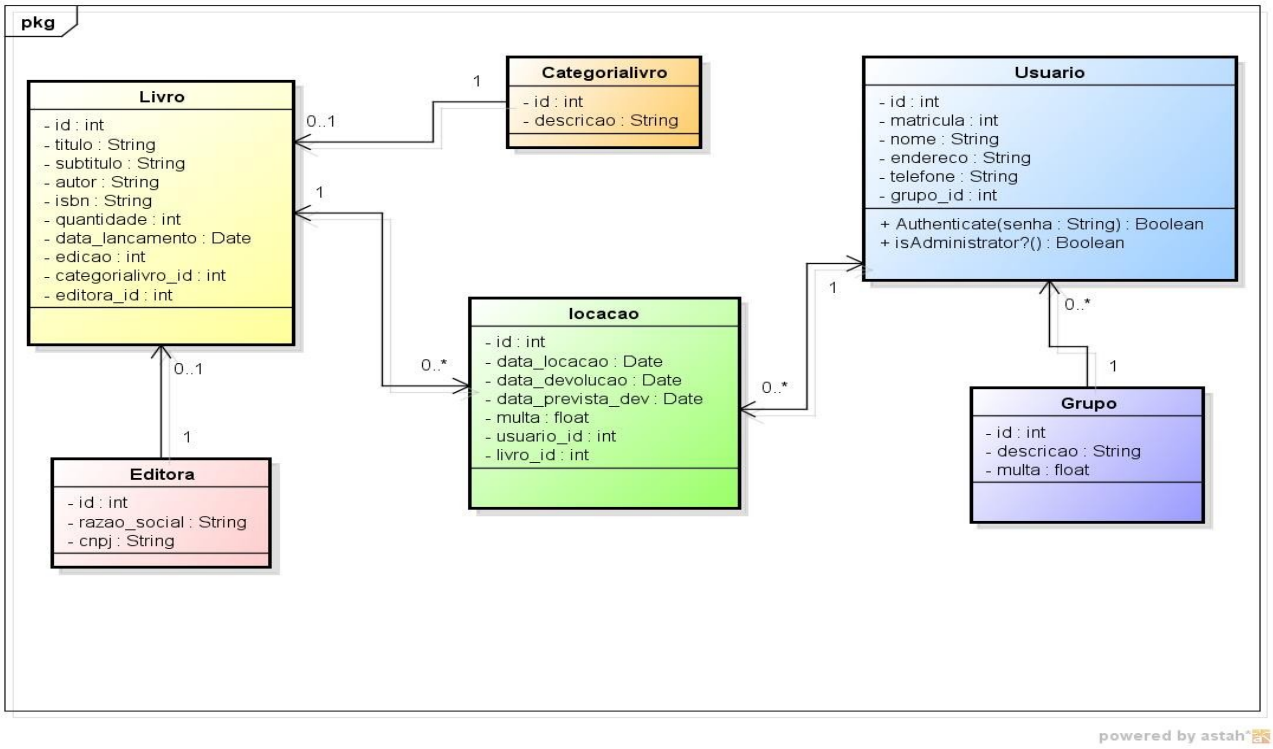

Figura 1 – Diagrama de Classes da aplicação

DESENVOLVIMENTO DA APLICAÇÃO

A criação da aplicação é feita através de comandos do framework Rails. A estrutura do projeto é criada no padrão de projetos MVC (Model-View-Controller), que não será abordado neste artigo.

A aplicação foi construída no sistema operacional Linux, abaixo serão listados as ferramentas utilizadas para o desenvolvimento da aplicação.

- Framework Rails 3.2.3;
- Linguagem Ruby 1.8;
- Baco de dados MySQL 5.1;
- Controle de versão online GitHub;
- Gerenciador de Pacotes RubyGem 1.7.2.

1 Criando da estrutura

O Rails cria toda a estrutura de diretórios necessária para o desenvolvimento do projeto, o comando abaixo, cria a estrutura básica do projeto, na qual utilizará o banco de dados MySQL.

$ rails -new biblioteca -d mysql

2 Criação dos Scaffolds

Os scaffolds são uma forma de criar rapidamente todos os recursos em uma única operação, tais como Create, Update, Delete e Insert.

Em seguida, será apresentado os comandos para a criação de todas as classes.

- Criação do Grupo

  $ rails generate scaffold Grupo descricao:string multa:float

- Criação do Usuário

  $ rails generate scaffold Usuario nome:string endereco:string telefone:string matricula:integer

- Criação da Editora

  $ rails generate scaffold Editora razao_social:string cnpj:string

- Criação da Categorialivro

  $ rails generate scaffold Categorialivro descricao:string

- Criação do Livro

  $ rails generate scaffold Livro titulo:string subtitulo:string autor:string quantidade:integer prazo_entrega:integer isbn:string data_lancamento:date edicao:integer editora:references categorialivro:references

- Criação da Locação - Relacionamento N x N da Tabela Livros com Tabela Usuarios

  $ rails generate scaffold Locacao data_locacao:datetime data_devolucao:datetime data_prevista:date multa:float livro:references usuario:references

3.2 Criação do banco de dados

Após criar todas as classes do projeto, utilizamos os comando abaixo para criar o banco de dados e criar as tabelas apartir das classes. A criação do banco de dados é configurada no arquivo /config/database.yml, deve-se adicionar o nome de usuário e senha do usuário administrador do SGDB (sistema gerenciador de banco de dados).

- Comando para criar o banco de dados

  $ rake db:create

- Comando para criar as tabelas apartir das classes

  $ rake db:migrate

- Comando para deletar o banco de dados

  $ rake db:drop

- Comando para popular alguns registro já pré-definidos

  $ rake db:fixtures:load

3 Controle de versão com o GitHub

O GitHub é um sistema web de controle de versão de arquivos, e tem a finalidade controlar as diferentes versões de um documento. O desenvolvimento de uma aplicação criado sem este tipo de controle pode gerar diversos problemas como:

- Arquivos duplicados;
- Arquivos alterados por um programador sem o conhecimento de outro;

- O time não consegue visualizar em tempo real o que outros estão desenvolvendo;
- O programador também não consegue voltar a um estado anterior.

Normalmente, existe um equipe de programadores que estão trabalhando em conjunto para desenvolver um projeto e nada mais eficiente do que um controle correto das versões dos arquivos fontes.

Para o correto funcionamento do controle de versão foi registrado no GitHub todos os desenvolvedores.

4 PROBLEMAS E DIFICULDADES

No andamento do desenvolvimento tivemos vários problemas e dificuldates encontradas, a seguir será apresentado as principais:

4.1 Adicionando o projeto ao GitHub

Nosso grupo é constituido de 3 integrantes e nos organizamos para que cada um fica-se responsáveis por determinadas partes do sistema, porém tivemos problemas ao enviar o projeto ao GitHub.

Ao criar o repositorio e após fazer o clone do projeto, o GitHub adiciona um arquivo chamado *.gitignore*, que é um arquivo texto oculto no sistema, e possui diretórios e extenções de arquivos que não devem ser enviados para o controle de versão. Logo após criar o projeto, nosso grupo fez o *commit* de uma parte do sistema, adicionando a estrutura ao GitHub, porém não haviamos configurando o arquivo *.gitignore*, ocasionando versionamento dos arquivos do diretório /*log* e /*tmp*, que são os logs do sistema e arquivos temporários, respectivamente. Com isso ao baixar o projeto, gerava *merges* conflitantes, arquivos incompletos e diversos outros problemas. Como não conseguiamos resolver o problema sempre que um integrante enviava o projeto ao GitHub o outro era obrigado a fazer um clone, ou recriavamos o repositório do GitHub e sicronizavamos novamente.

Depois de muita pesquisa, e o auxílio da documentação disponível em GITHUB, conseguimos contorar este problema configurando corretamente o arquivo *.gitignore*. Nosso repositório no GitHub foi criado para a linguagem Ruby, posteriormente criamos novamente o repositório compativel com o framework *Ruby On Rails* e o arquivo *.gitignore* apresentou-se configurado corretamente, resolvendo o problemas do envio dos fontes ao versionador.

4.2 Criação do Scaffold de Categoria Livros

Quanto geramos um *scaffold* para gerenciar as categorias para os livros, criou-se varios problemas pois, colocarmos o nome “CategoriaLivro”. O *framework* rails gerou o model chamado “CategoriaLivro”, porém as rotas de URL foram geradas com o nome *categoria_livros*, mesmo assim o *controller* e suas *views* funcionavam. Mas quando fomos gerar o *scaffold* para os “Livros”, e nele colocamos uma propriedade para referênciar as categorias, ocasionava erro. A propriedade/atributo da model gerada também não conseguia conectar na *model* “CategoriaLivro” correta, gerando conflitos intermitentes. Para solucionar este problema e não despadronizar a forma de como o *rails* cria seus objetos e arquivos, criamos o *scaffold* com o nome de “categorialivro”.

4.3 Criação do novo Layout

Necessitavamos de um layout padrão para utilizar na página de locações, para que o usuário consiga efetuar as locações, sendo que este layout é diferente do acessado pelos administradores do sistema. Por padrão o *framework rails* é criado um *layout* padrão, porém necessitavamos de dois layouts para fazer nossa aplicação funcionar. Com base na referência de TUTORIALS POINT, criamos um layout chamado *estrutura.html.erb*, este foi inserido na pasta padrão dos layout (app/views/layouts), tornando-se a estrutura base HTML. Para reutilizar o *layout* padrão nas páginas filhas (pesquisa) é necessário explicitar nas classes dos *controller*.

Por exemplo:
```
class SiteController < ApplicationController
        layout 'estrutura'
```

4.4 Pesquisa dos livros

Na página de locação, existe pesquisa por título e por categoria do livro. Tivemos problemas nos métodos para buscar todos os livros conforme a pesquisa requisitada. Primeiramente utilizamos os métodos do *model* “Livro” chamados “Livro.find_by_titulo” e “Livro.find_by_categoria”, porém no momento da geração do html, onde tentavamos percorrer um *array* para exibir todos registros encontrados, gerava erro. Este erro acontecia pois os métodos executados traziam um objeto, que nem sempre é um *array*. Para solução do problema foi utiliza o método *where*, onde este sim retorna um *array* com a lista de “Livros” conforme a pesquisa.

4.5 Customização das rotas do Rails

Em duas situações precisamos declarar as rotas explicitamentes para o funcionamento do sistema. A primeira para criar o controle de login e a segunda para a página de locação. É necessário declarar estas rotas, pois as rotas padrões do *rails*, quando gera-se um *scaffold* por exemplo, são para os *actions* padrões, e como neste nosso caso nós criamos um *controller* manualmente específico com nossa necessidade, com auxílio de ENGINE YARD, foi customizado as rotas para atender o comportamento adequado.

Para o controle de sessão foi necessário a criação das rotas para o login e logout, estas rotas quando invocadas chamam o *controller Sessions* e os métodos *new* e *destroy* respectivamente. Cada um destes métodos tem suas rotinas. O *create* serve para autenticar o usuário e redirecionar o usuário corrente para o sistema gerencial ou para a página de locaçao, conforme o perfil do usuário. Já o método *destroy* é utilizado para destruir a sessão corrente do usuário e redirecioná-lo para o login.

Para as pesquisas funcionarem foi mais complexo, a pesquisa por categoria possui um rota, pequisa por título outra, detalhes do livro outra, sem contar é claro da rota principal do site externo, pois para exibir corretamente as URLs foi necessário implementar rotas alinhadas.

5 CONCLUSÃO

Este artigo apresentou conceitos e técnicas para o desenvolvimento de uma aplicação web sob o *framework Ruby on Rails*. Apresentou as dificuldades e problemas encontrados ao longo do desenvolvimento da aplicação em uma equipe.Também ratificar os conhecimentos adquiridos na disciplina de linguagem de programação web.

REFERÊNCIAS

ENGINE YARD. **THE LOWDOWN ON ROUTES IN RAILS 3.** Disponível em: <http://www.engineyard.com/blog/2010/the-lowdown-on-routes-in-rails-3/>. Acesso em: 02 julho 2012.

GITHUB. **HELP.** Dispnível em <http://help.github.com/>. Acesso em: 30 junho 2012.

RUBY ON RAILS. **DOCUMENTATION.** Disponível em: <http://api.rubyonrails.org/>. Acesso em: 30 junho 2012.

TUTORIALS POINT. **RUBY ON RAILS QUICK REFERENCE GUIDE.** Disponível em: <http://www.tutorialspoint.com/ruby-on-rails/rails-quick-guide.htm>. Acesso em: 01 julho 2012.