CONTROLE DE ENTREGAS ONLINE

Daniel de Souza Santana <daniel.santanna@gmail.com>

Ricardo Corrêa Ribeiro <[michels\_ribeiro@ig.com.br](mailto:michels_ribeiro@ig.com.br)>

Vinicius Couto <vc.vine@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 construção do protótipo utilizando o *framework*RubyonRails. Possuindo interfaces *Web* para sua utilização, objetivando controlar entregas entre uma origem e um destino.

**Palavras-chave:**RubyonRails, Web, sistema de entrega.

Abstract

**Title:** “Online Delivery Control”

*This article describes the process of building the prototype using Ruby on Rails framework. Having to use Web interfaces aimed at controlling, supplies between a source and destination.*

**Key-words**: Ruby on Rails, Web, delivery system.

1. Introdução

Este trabalho tem como objetivo a criação de uma aplicação Web baseada no Ruby on Rails. Tal aplicação deve permitir controlar um sistema de entregas *online*.

A construção de sistemas para Web engloba diversas ferramentas e recursos. Tais recursos são empregados nas diversas camadas do desenvolvimento. O *framework* Rails é feito em Ruby e veio para agilizar esse processo, utiliza o conceito MVC: Modelagem (*Model*), Visualização (*View*) e por fim, Controlador (Controller).

O protótipo tem como objetivo criar uma ambiente Web para o cadastro de entregas. O cliente, previamente cadastrado, realizará o seu pedido de entrega e o funcionário irá validar o mesmo.

1. Ruby on Rails

O *Ruby on Rails* é um framework também conhecido como *RoR* ou apenas *Rails*. Um *framework* é um conjunto de bibliotecas e ferramentas que são utilizadas para criação de aplicações de forma dinâmica, neste caso aplicações Web dinâmicas. Este framework é para o desenvolvimento de aplicações Web dirigido a dados usando linguagem de criação de scripts*.* O Ruby on Rails possui várias bibliotecas embutidas tanto para conexão com banco de dados como para: geração de serviços Web, funcionalidades Ajax entre outros (DEITEL, 2008 p.496).

O sistema MVC (Modelo, Visualização e Controlador) divide a aplicação em aspectos de lógica de negócio sendo que: o Modelo trata da modelagem dos dados do sistema, a Visualização trata das interfaces com o usuário (parte gráfica) e o Controlador trata da lógica de processamento de entrada. Todos interagem uns com os outros.

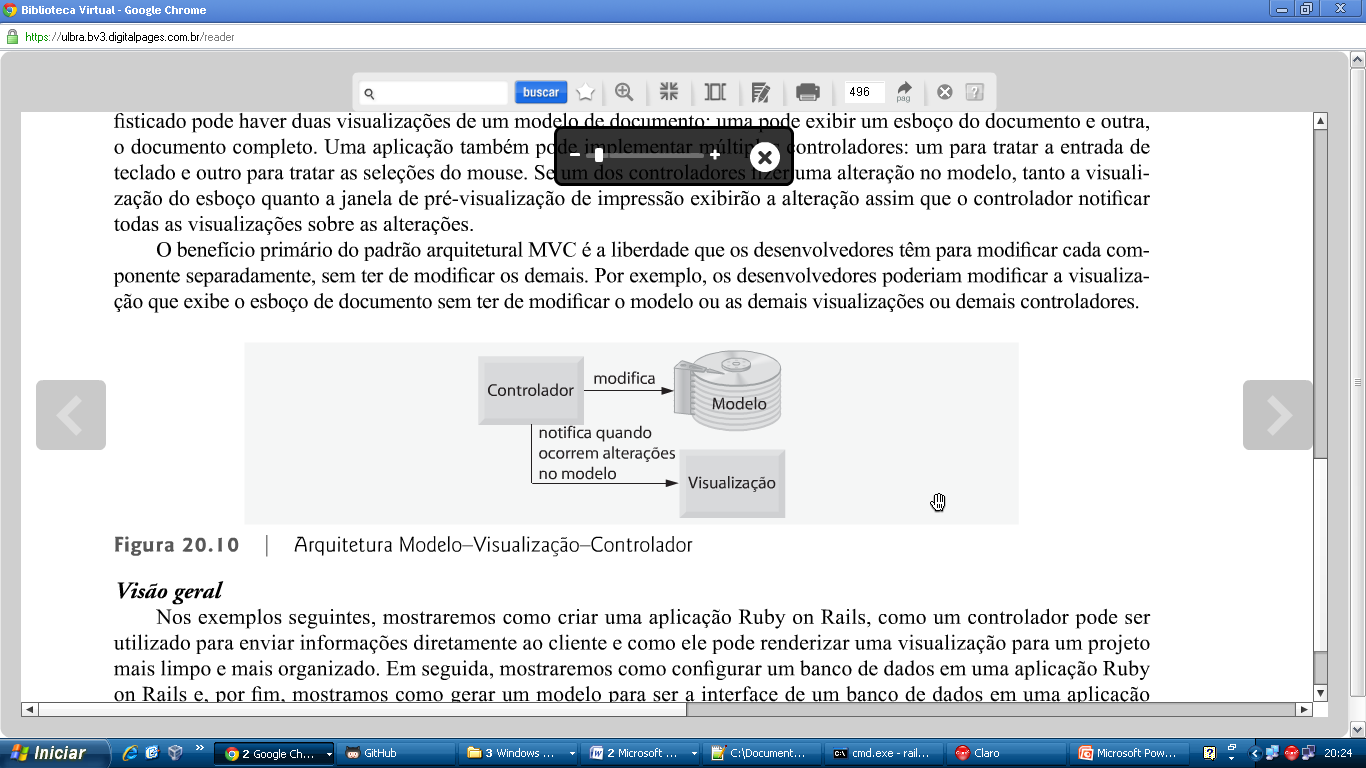


Figura 1 – Arquitetura MVC (DEITEL, 2008 p.502).

A Figura 1 mostra de forma visual a relação entre cada um dos componentes do sistema MVC. Onde a partir do Controlador é processada a lógica e as entradas do usuário. O Modelo sofre alterações de acordo com estas iterações, sendo que o Modelo, depois de modificado notifica a Visualização através do Controlador.

1. DESENVOLVIMENTO DO PROTÓTIPO

O protótipo foi modelado baseando-se nos seus principais requisitos funcionais, onde de forma sintetizada seria: cadastrar em banco de dados as informações necessárias para a busca e entrega de mercadorias. A Figura 2 demonstra o diagrama de classes do sistema.

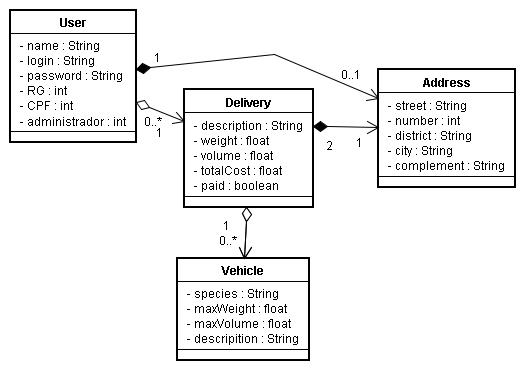


Figura 2 – Diagrama de classes do protótipo.

A partir da Figura 2 tem-se o cenário do sistema, onde a classe “Vehicle” conterá os atributos que definirão o tipo de veiculo a ser utilizado na entrega, contendo os atributos: “species” é um *String* que define o tipo do veículo (moto, carro, caminhão entre outros); “maxWeight” e “maxVolume” do tipo *float*, onde o primeiro define a máxima carga em quilos suportada, e o segundo o volume máximo, por último o “description” onde será possível escrever algum comentário a respeito deste veiculo.

A classe “Address” vai conter os atributos necessários para designação dos endereços, sendo que a mesma vai ser um atributo da classe “User” e também será da classe “Delivery”.

A classe “Delivery” é onde estará o pedido de entrega, seus atributos são: “description” do tipo *String* servindo para colocar alguma observação a respeito da entrega; “weight” um *float* que determina o peso da mercadoria; “volume” dado *float*que define o volume do pacote; “totalCost” dado float que define o custo da entrega, o *boolean* “paid”, o qual indicará se a entrega em questão já foi paga, o objeto da classe “Vehicle” que define o tipo de veiculo utilizado nesta entrega, e por fim, os dois objetos da classe “Address” que definirão os endereços de origem e destino da entrega.

Observa-se também que cada usuário poderá ter nenhuma ou várias entregas (objetos da classe “Delivery”, que cada entrega terá obrigatoriamente: um veiculo associado (objeto da classe “Vehicle”, e dois endereços associados, sendo um destino e o outro origem.

Por fim, a classe “User” vai conter os dados do usuário (administradores e clientes do sistema), possuindo os atributos: “name”, “login”, “password”, “RG”, “CPF” e “administrador”, onde este último definirá o nível de acesso.

As classes foram implementadas com a utilização do “Scaffold”, onde os comandos utilizados foram:

* rails generate scaffold address street:string number:integer district:string city:string complement:string;
* rails generate scaffold vehicle species:string maxWeight:float maxVolume:float description:string;
* rails generate scaffold user name:string login:string password:string rg:integer cpf:integer address:references;
* rails generate scaffold delivery description:string weight:float volume:float totalCost:float paid:boolean addressFrom:integer addressTo:integer vehicle:references.

Um *scaffold* é um meio de criar código para um determinado modelo (que indicamos) através de um determinado controlador (que indicamos também). É um meio de começarmos rapidamente a ver os resultados no nosso navegador web, e um método muito rápido de implementar o CRUD (*Create, Retrieve, Update, Delete*) na sua aplicação. Lembrando que o *scaffold* cria código que, fatalmente, vai ser alterado depois, a não ser que você deseje manter o nível muito básico de interface e controle padrão de campos que ele proporciona (EUSTÁQUIO, 2006, p. 6).

Tais comandos geraram o código que serviu como base para o desenvolvimento. Todo o conjunto MVC criado foi alterado e melhorado em busca dos objetivos do protótipo.

1. Dificuldades Encontradas

A utilização do *framework* RoR na plataforma Ubuntu 10.xx tomou muito tempo do desenvolvimento, muitos problemas com versões de bibliotecas dentre outros componentes. Este fato forçou a migração para o SO Windows XP Sp3, o qual com o instalador RoR apresentou somente erro no formato das datas, fato este que foi rapidamente solucionado.

O entendimento do conceito de funcionamento do RoR é imprescindível para uma boa implementação, no entanto, este entendimento foi sendo adquirido no decorrer do desenvolvimento, prejudicando um pouco o resultado final deste desenvolvimento.

1. Conclusão

O *framework* Ruby on Rails demonstra um grande potencial para desenvolvimentos ágeis. Dentro do tempo de desenvolvimento disponível acredita-se que o protótipo atinge os objetivos, tendo grandes possibilidades de melhorias.

Como melhorias futuras podem-se descrever:

* Adicionar funcionalidade junto ao Google Maps, para que vincule os endereços aos respectivos dados da classe “address”;
* Reorganizar a parte de cadastros, tanto para clientes quanto para os funcionários, colocando as regras de acesso baseado no nível de acesso (atributo “administrador”);
* Realizar o calculo automático do valor do transporte, baseado em distâncias, peso, volume e tipo de veículo utilizado;
* Reajustar os relacionamentos dos endereços, para que force que tanto o destino quanto a origem tenham usuários cadastrados.

Estas são apenas algumas das melhorias futuras que podem ser implementadas no protótipo, sendo necessário um estudo mais aprofundado do *framework* RoR.

Finalizando, acredita-se que a prática continua com o RoR pode permitir a criação rápida de sistemas para Web de forma profissional, aproveitando as capacidades do *framework* que vem crescendo a cada dia.

Referências

DEITEL, Paul J. **Ajax, Rich Internet Applications e desenvolvimento Web para programadores**. São Paulo: Pearson Prentice Hall, 2008.

EUSTÁQUIO, Rangel. **Tutorial de Rails.** São Paulo, 2006.Aceso em 03 de julho 2012. 67 f. Disponível em: http://pt.scribd.com/doc/8160439/Tutorial-Rails.