CONTROLE DE ENTREGAS ONLINE

Daniel de Souza Sant'Anna <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* Ruby on Rails. Possuindo interfaces *Web* para sua utilização, objetivando controlar entregas entre uma origem e um destino.

**Palavras-chave:** Ruby on Rails, 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*.

O trabalho de construção de sistemas para Web engloba diversas ferramentas e recursos. Tais recursos são empregados nas diversas camadas do desenvolvimento. O *framework* Ruby on Rails veio para agilizar este processo, utilizando um sistema de organização que compreende a: modelagem (*model*), visualização (*view*) e por fim, parte controle, esta organização também é conhecida por MVC (*Model, View e Control*).

Com base neste *framework* foi construído este protótipo, valendo-se de todas as técnicas disponíveis para o desenvolvimento ágil.

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.

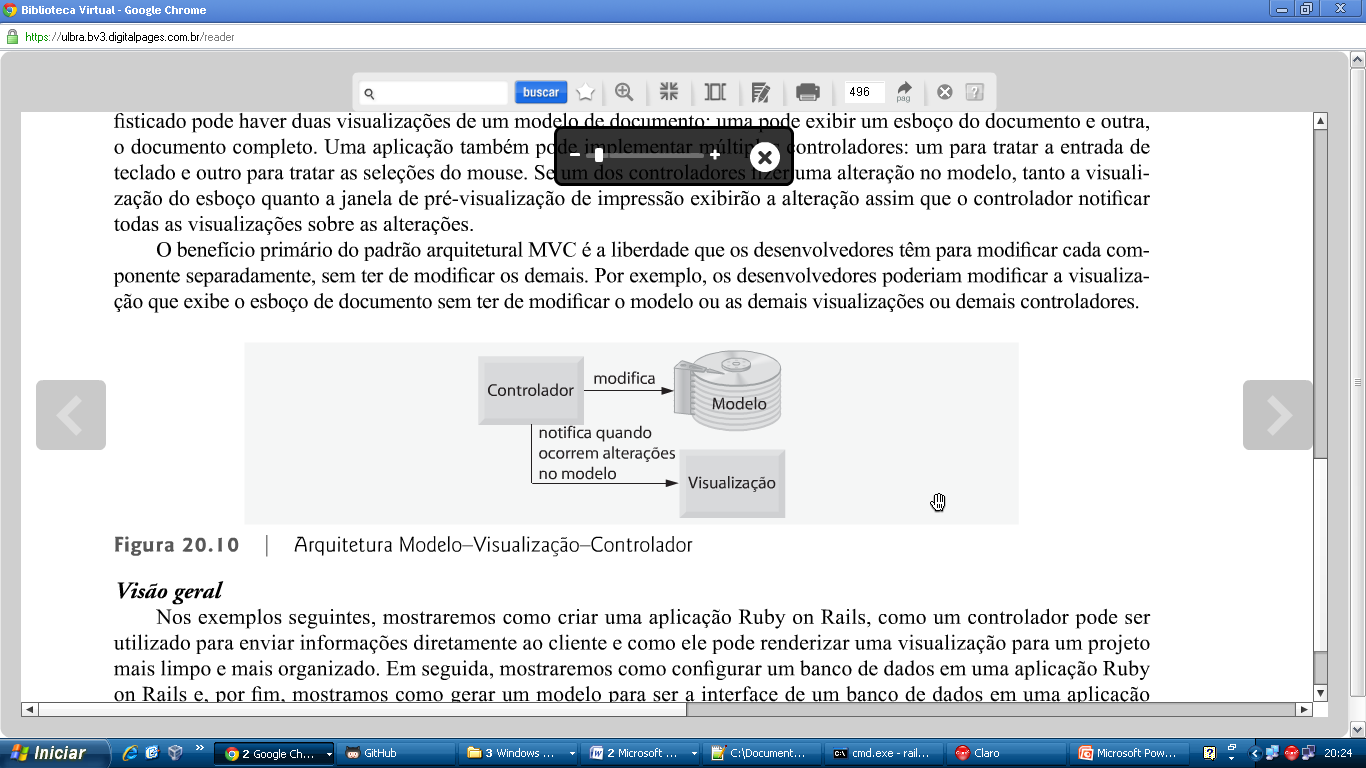


Figura 1 – Arquitetura MVC (DEITEL, 2008 p.502).

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

Inicialmente foi pensada a modelagem do sistema com base nas necessidades. Onde o objetivo do protótipo e 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.

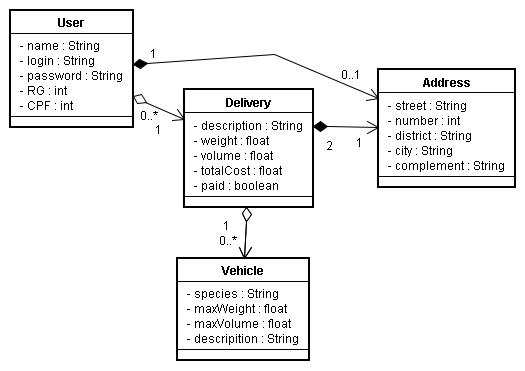


Figura 2 – Diagrama de classes do protótipo.

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, onde o atributo: “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 o origem.

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.

Referências

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