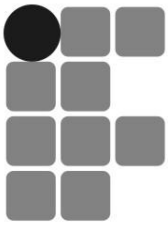


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE
Campus Campos-Centro

Secretaria de Educação Profissional e Tecnológica

Ministério da Educação

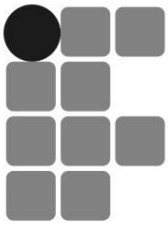
BACHAREL EM SISTEMAS DE INFORMAÇÃO

ALEXANDRE DOS SANTOS SAMPAIO SILVA

LAÍS STELLET DA SILVA

TITUL0

Campos dos Goytacazes/RJ
2014



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE
Campus Campos-Centro

Secretaria de Educação Profissional e Tecnológica

Ministério da Educação

BACHAREL EM SISTEMAS DE INFORMAÇÃO

ALEXANDRE DOS SANTOS SAMPAIO SILVA

LAÍS STELLET DA SILVA

TITUL0

Trabalho de conclusão de curso apresentado ao Instituto Federal Fluminense como requisito obrigatório para obtenção de grau em Bacharel de Sistemas de Informação.

Orientador: Prof. DSC Maurício José Viana de Amorim
Co-orientador: Prof. Fernando Carvalho

Campos dos Goytacazes/RJ
2014

ALEXANDRE DOS SANTOS SAMPAIO SILVA

LAÍS STELLET DA SILVA

Tema – ???

Trabalho de conclusão de curso apresentado ao Instituto Federal Fluminense como requisito obrigatório para obtenção de grau em Bacharel de Sistemas de Informação.

Aprovada em ?? de ?? de 20??

Banca avaliadora:

Prof. Fernando Luiz de Carvalho Silva (Co-Orientador)
Mestre em Engenharia de Produção / UENF
Instituto Federal de Educação, Ciência e Tecnologia Fluminense / Campus Campos Centro

---

Prof. DSC Maurício José Viana Amorim
Instituto Federal de Educação, Ciência e Tecnologia Fluminense / Campus Campos Centro

Dedico este trabalho à minha mãe e amigos por todo apoio, compreensão

e contribuição que me prestaram.

AGRADECIMENTOS

Primeiramente agradeço a minha mãe que sempre esteve ao meu lado nestes longos anos. Agradeço também a todos meus amigos e colegas de trabalho do NSI, que me apoiaram e incentivaram. Bem como aos professores Rogério Atem e Fernando Carvalho que contribuiram muito para que esse projeto fosse realizado.

O computador não é mais o centro do mundo digital.

Tim Cook

RESUMO

Este trabalho descreve sobre o desenvolvimento de novas funcionalidades para uma ferramenta livre de manipulação de documentos em larga escala, sendo o objetivo dessas funcionalidades permitir a ferramenta que se adapte a manipulação outros formatos de arquivos, de multimídia, por exemplo. Ainda é descrito no trabalho um pouco sobre as aplicações e/ou outras ferramentas que foram integradas a fim de tornar possível a existência destas funcionalidades. Através da parceria da empresa francesa Nexedi SA, com o Núcleo de Pesquisa em Sistemas de Informação(NSI-IFF), foi possível implementar as funcionalidades de manipulação de formatos correspondentes a arquivos de vídeo, áudio, imagens e PDF; cabendo a esta manipulação realizar a conversão, inserção e extração de dados destes arquivos. Encontram-se ainda, neste trabalho, a estrutura da ferramenta, conceitos básicos empregados para formação da ferramenta e sua integração com o núcleo linux, bem como conceitos da linguagem Python, a qual foi utilizada para seu desenvolvimento.

PALAVRAS-CHAVE: Serviço Web, Software livre, Cloud Computing

ABSTRACT

This work describes about the development of new functionalities to a free tool of document manipulation on a large scale, being this new functionalities responsible for allowing the tool to adapt in handling other file formats, such as multimedia ones, for example. It stills describes a little about the applications and/or other tools used for integrating its own with propose to handle with this new functionalities. Through this partnership between the French company Nexedi SA with the aid of the Nucleus of Research in Information Systems (NSI-IFF), was possible to implement the ability to manipulate shapes corresponding to video files, audio, images and PDF; being this manipulate as well responsible for converting, inserting and extracting data from this files. Still can find in this job, the structure of this tool, basic concepts used to create the tool and integrate it with the core Linux, as well as concepts of Python, which was used for its development.

KEYWORDS: Web Service, Free software ,Python, Cloud Computing

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Objetivos

1.2 Justificativa

1.3 Organização do Trabalho

2 TECNOLOGIAS EMPREGADAS

Para o desenvolvimento do Atividades Complementares foi essencial o uso de algumas ferramentas as quais pudessem ser incorporadas e tornassem possível a criação de funcionalidades para este projeto.

Ao longo de muitos anos, uma pergunta é feita por vários desenvolvedores. Qual o melhor Framework de desenvolvimento web já criado? Contudo esta é uma pergunta respondida de diversas formas, com base em vários argumentos e situações. E a resposta entretanto continua a mesma - Aquela que aumenta a produtividade e diminui esforços no desenvolvimento. E depois de muito estudo e debate fora concluído que um Framework que reduza códigos gigantes e por muitas vezes difíceis de se ler e que sem margem de duvida diminuem o aprendizado são aqueles que aumentam a produtividade. Se buscarmos mais a fundo iremos nos debater com vários web Framework que focam em produzir aplicações, sejam elas web ou não. No entanto essas pecam em dois pontos vitais: produtividade e aprendizado. Um equilíbrio teve de ser estabelecido entre esses fatores e foi nesse momento que Ruby surgiu precedido do Rails como seu Framework de desenvolvimento web.

Serão apresentadas as seguintes tecnologias: RVM como ambiente virtual para o desenvolvimento, Ruby como linguagem de programação, Rails como web framework, Mysql como base de dados inicial, MongoDB como base de dados conclusiva e o mais importante RubyGems para auxiliar na produção do sistema, git como ferramenta de repositório de código compartilhado com menbros do projeto.

Os capítulos 1 e 2 apresentam um conceito simplificado de Ruby bem como seu Framework o Rails com alguns exemplos de funcionalidades em cada uma.

2.1 RVM (Ruby Version Manager)

RVM é uma ferramenta de linha de comando que permite ao usário facilemte instalar, gerenciar e possuir varios ambiente de desenvolvimento ruby em uma só máquina com suas respectivas versões de rubygems.

2.1.1 Ruby

Foi desenvolvida em 1994, e apresentada ao publico em 1995 por **Yukihiro Matsumoto**, que é mundialmente conhecido como Matz que para criar ruby uniu partes das suas linguagens favoritas (*Perl, Smalltalk, Eiffel, Ada, e Lisp*) para formar uma nova linguagem que equilibra a *programação funcional com a programação imperativa*. Passou a se tornar realmente reconhecida através de **Dave Thomas** , mais conhecido como um dos *"Programadores pragmáticos"*, que adotou o Ruby como uma de suas linguagens preferenciais e acabou escrevendo um dos livros mais completos sobre a linguagem, o Programming Ruby. Com isso o número de adeptos a essa linguagem subiu muito rápido no ocidente. Ruby é uma das únicas linguagens nascidas fora do eixo EUA/Europa, sendo criada no Japão.

Matz criou uma linguagem mais poderosa que Perl e mais orientada a objetos que python. Em Ruby tudo é objeto, e possue métodos que podem ser facilemte acessados e modificados. O tornando assim uma linguagem mais simples de se ler e ser compreendida, facilitando o desenvolvimento e manutenção de sistemas escritos com com essa base. Um dos objetivos principais da linguagem é a praticidade. É possível que seja feito um algoritmo simples, sem precisar que se preocupe com as limitações da linguagem ou do interpretador.

O Ruby possui algumas características para manter a sua praticidade, como, uma linguagem enxuta que quase não há necessidade de colchetes e outros caracteres; a disponibilidade de diversos métodos de geração de código em tempo real, como os "attribute accessors";

> *"Ruby é simples na aparência, mas é muito mais complexo no interior, tal como o corpo-humano!"* - Ruby-Talk, (MATZ, 2000).

Uma forma prática de se instalar bibliotecas em ruby é através de ***RubyGems*** , com ela é possível instalar e atualizar bibliotecas com uma única linha de comando, de maneira muito similar com os gerenciadores de pacotes de ambientes operacionais linux ou unix; Ruby possui uma notação muito utíl aos desenvolvedores, estas são chamadas de blocos de código, que ajudam o programador a passar um ou mais trechos de instruções para o método descrito em suas classes concretas. Ruby possui o contexto de Mixins, que simula herança múltipla, sem cair em seus respectivos problemas encontrados em outras linguagens. Ruby é classificado como dinamicamente tipado, por esse meio todos os tipos são objetos, não há tipos primitivos como era e ainda é feito em outras linguagens do genero como **C++, Java(...)** , bem como suas definições de classes. No entanto variáveis de instância que por sua vez referenciam estes objetos não possuem um tipo específico. Um exemplo clássico em ruby seria uma reescrita da classe Fixnum nativa do ruby.

```
[Irb]
[irb - prompt]
  2.1.0 :001 > class Fixnum
  2.1.0 :002 >  alias_method :old_add, :+
  2.1.0 :003 >  def plus(other)
  2.1.0 :004 >    self.old_add(other) * 2
  2.1.0 :005 >  end
  2.1.0 :006 > end

[irb - prompt]
  2.1.0 :007 > numero = 2 + 2
  2.1.0 :008 > numero
   "8"
  2.1.0 :009 > numero = nil
```

Código 2.1: Exemplo de uma *Reescrita* de Classe

O operador *plus* ou + é um método em Ruby, ao contrário de outras linguagens existentes. O resultado desse método reescrito na classe Fixnum diz que todoo e qualquer numero somado com o método soma, ira sempre executar a operação de soma em conjunto com o operador de multiplicação neste caso todo numero somado será multiplicado por dois após a operação anterior, e logo em seguida pelo fato de ruby ser dinamicamente tipado, o valor da variável **numero** é alterado para nil, o que siguinifica que podemos alterar o contexto de uma variável ou até mesmo inserção de código em tempo de execução.

Ruby possui uma ferramenta muito interessante, semelhante ao array visto em outras linguagens o ruby hashes, bastante similar ao dicionario de dados do python. Um ruby hashe utiliza chaves em vez de colchetes precedido de literais. O literal deve fornecer: uma chave e um valor agregados. Por exemplo, se quiséssemos mapear os dados de um usuário poderíamos faze-lo da seguinte forma:

```
    user_section = {
      name: 'xpto' ,
      email: 'xpto@xpto.com' ,
      password: 'headwind' ,
      nickname: 'headwind' ,
      prefered_language: 'Ruby On Rails with MongoDB'
    }
```

Código 2.2: Exemplo de um *Ruby Hash*

O conceito acima é muito semelhante a forma como o MongoDB manipula seus documentos um contexto que será explicado mais a frente.

Ultimamente a linguagem tem sido foco da mídia especializada devido ao seu web framework feito em Ruby, o Rails desenvolvido por (HANSSON, 2006). Ainda hoje, toda a responsabilidade, quanto a, implementações de novas funcionalidades, é do Matz. Todas as decisões relacionadas à linguagem tem que passar por ele antes de serem implementadas e virem

à publico. E mesmo assim a comunidade Ruby é forte o suficiente pra sobreviver caso alguma coisa aconteça com o Matz. Pois há muitas pessoas que estão conectadas ao código tanto quanto o próprio Matz. Uma das grandes diferenças das outras tecnologias open-source, é que não tem uma empresa bancando os seus custos. O projeto sobrevive de doações feitas pelos usuários satisfeitos e por empresas que conseguiram aumentar sua produtividade e performance usando apenas Ruby ou Ruby On Rails. Em uma de suas declarações Matz fala sobre o que ele esperava obter quando criou a linguagem:

"Eu conhecia muitas linguagens antes de criar o Ruby, mas nunca estava satisfeito com elas. Elas eram feias, rigorósas, mais complexas ou mais simples do que eu esperava. Eu queria criar a minha própria linguagem que me satisfizesse como programador. Eu sabia muito sobre o público a ser alcançado: eu mesmo. Para minha surpresa, muitos programadores do mundo todo sentiam o mesmo que eu. Eles ficaram felizes quando descobriram e programaram no Ruby. Do começo ao fim do desenvolvimento da linguagem Ruby, concentrei minhas energias para fazer uma programação rápida e fácil. Todas as características do Ruby, incluindo as características de orientação a objetos, são designadas a funcionar com programadores comuns (por exemplo: eu) que esperam que elas funcionem. A maior parte dos programadores acha que ele é elegante, fácil de usar e sentem prazer em usá-lo."(MATZ, 2000).

2.1.2 RubyGems

Uma RubyGem ou simplesmente *Gem* é uma biblioteca como em qualquer outra linguagem de programação já criada, por exemplo: **Python, Java, C++, C** , especificas para Ruby, que fornecem um formato padrão para aplicações. Uma Gem é escrita especialmente para facilitar o uso de determinada funcionalidade. Uma Gem é um conjunto de arquivos feitos em Ruby, etiquetados com nome e versão, cada uma possui, em seu escopo todas as características correspondente a sua arquitetura via um arquivo de configuração chamado **"gemspec"**. Tomando como exemplo a gem 'rspec-rails' que possui em seu escopo arquivo rspec-rails.gemspec que possui toda a especificação desta desde qual o grupo responsável por mante-la com atualizações constantes, licenças e dependências. Gems podem ser utilizadas para estender ou modificar certas funcionalidades, geralmente são distribuídas por outros desenvolvedores Ruby mais conhecidos como ***rubistas*** , varías delas possuem até mesmo comandos expecificos auxiliar e agilizar o desenvolvimento, além de que em Ruby rubygems podem ser integradas umas com as outras para facilitar ainda mais os ruby programadores.

2.1.3 Como Ruby carrega RubyGems

Antes de tudo é muito útil saber como o Ruby carrega arquivos de dependência. O Ruby seja na verão 1.9.3 ou mais recente predispõem de um comando bem simples o require "ar-

quivo"com ele é possível carregar arquivo(s) diretamente ou especificando o caminho absoluto. Contudo existe ainda uma outra forma pouco usada o load "arquivo.rb". A única diferença é que com load deve ser colocado a estenção do arquivo e além disso se ao tentar executar o require novamente essa operação será negada pois este carrega uma vez esse arquivo, já o load permite carregar varias vezes. Load é muito útil quando se está testando um arquivo que está sofrendo alterações constantes. Existe ainda uma outra forma de se carregar arquivos é o require File.expand_path("./../spec_helper")

```
[Irb]
[irb - prompt - 1.9.3]
  1.9.3 :001 > require 'lib/fixnum'
    true
  1.9.3 :002 >

[irb - prompt - 1.9.3 - Full-Path]
  1.9.3 :001 > File.expand\_path("./../spec_helper")
    true
  1.9.3 :002 >

[irb - prompt - 2.1.0]
  2.1.0 :001 > load 'lib/fixnum.rb'
    true
  2.1.0 :002 >
```

Código 2.3: Exemplo de require e load Ruby 1.9.3 - 2.1.0

Como pode ser visto o comando require pode carregar arquivos usando caminhos relativos. Porém também pode ser usado caminhos absolutos. Para tal quando se deseja carregar alguma dependência em um projeto Ruby existe um arquivo chamado **"rubygems.rb"** localizado em **/.rvm/rubies/ruby-2.1.0/lib/ruby/2.1.0** ao qual contem toda a expecificação de Gems como carregalás na versão decorrente do ruby instalado. Logo se precisa carregar esse arquivo toda vez que se deseja utilizar alguma gem:

```
  require 'rubygems'
  require 'BCrypt'
```

Código 2.4: Exemplo de require 'rubygems'

2.2 Git

Git é um sistema de controle de versões distribuídas livre e de código aberto, projetado para lidar com qualquer projeto, desde o menor ao maior com rapidez e eficiência (CHACON, 2009).

A historia do Git está muito relacionada a criação do Linux e de Linus Torvalds, seu criador, bem como com toda comunidade de desenvolvimento Linux. Durante anos a comunidade utilizou a ferramenta *BitKeeper* para guardar a modificações do projeto.

Em 2005, após um problema com a proprietária deste, a comunidade decidiu criar sua própria ferramenta a partir da experiência com a anterior, houve um novo foco em: velocidade, *design* simples, suporte para desenvolvimento paralelo, distribuição completa e a habilidade necessária para lidar com projetos grandes sem perda de velocidade e dados.

Assim, esse novo sistema de versionamento permite que qualquer repositório seja o centro do versionamento, deixando todo *log* das modificações guardados nele sem que para isso precise de uma conexão a rede ou servidor geral.

2.2.1 Git e Subversion

Diferentemente do Git, o Subversion é um sistema de controle de versões centralizado, entretanto muito utilizado atualmente, principalmente por projetos livres.

Embora seja consideravelmente rápido, é extremamente desaconselhável para projetos grandes e principalmente desenvolvidos paralelamente.

REFERÊNCIAS BIBLIOGRÁFICAS

CHACON, S. *Pro Git*. São Paulo: Apress, 2009.

HANSSON, D. H. *Criador do Rails*. 2006. Disponível em: <http://eu.dbpedia.org/page-/David_Heinemeier_Hansson, https://github.com/dhh, http://david.heinemeierhansson.com-/about.html>. Acesso em: 04/03/2014.

MATZ. *Ruby-Creator*. 2000. Disponível em: <https://www.ruby-lang.org/pt/about/>.