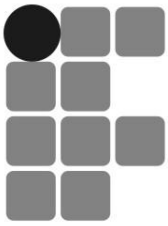


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE
Campus Campos-Centro

Secretaria de Educação Profissional e Tecnológica

Ministério da Educação

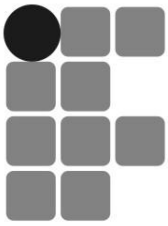
BACHAREL EM SISTEMAS DE INFORMAÇÃO

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.

Este capítulo apresenta um conceito simplificado de cada uma dessas integrações, desde a linguagem principal empregada até algumas das ferramentas utilizadas.

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.

```
1 [Irb]
2 [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 deci-

sões 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* são bibliotecas 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. Cada Gem possui, em seu escopo todas as características correspondente a sua arquitetura (via um arquivo 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 textbf *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.

```
1 >>> import subprocess
2 >>> subprocess.Popen('echo "Hello World!"', shell=True)
3 Hello World!
```

Código 2.3: Exemplo de uso do Subprocess

No código 2.3 existe um pequeno exemplo de como o Subprocess pode ser utilizado.

Neste exemplo ele reproduz um acesso ao *shell* do sistema a fim de demonstrar um *Hello World* através do comando *echo*, é possível observar o uso do *shell* no final da segunda linha em que cita-se “shell=True”.

Em aplicações que utilizam ambientes dedicados, como citado na sessão **??**, é preferencial que a variável *shell* tenha valor *false*,a fim de usar as aplicações deste ambiente, entretanto este já é seu valor por padrão, assim não é preciso declará-lo, como é visto no código 2.4.

Nesses casos também é preferível o uso da variável “env” que representa variáveis de *environment*, isto é, variáveis de ambiente. Ao substituir esta variável o Buildout direciona o processo a utilizar os binários que estão no seu ambiente próprio ao invés do ambiente original, que neste caso é o próprio sistema.

O processo de sobrescrever a variável “env” é similar ao passo de redefinir a variável *PATH* do sistema.

```
1 command = ["convert", self.file, output]
2 stdout, stderr = Popen(command,
3                         stdout=PIPE,
4                         stderr=PIPE,
5                         env=self.environment).communicate()
```

Código 2.4: Exemplo de uso do Subprocess com *PIPE* extraído do CloudOoo

No exemplo, código 2.4, também é utilizada a opção *PIPE* que permite ao comando Popen retornar as mensagens de saída e erro respectivamente através das variáveis *stdout* e *stderr*, para que seu conteúdo possa ser após o uso do comando.

2.1.3 Zope

Zope (Z Object Publishing Environment) é um serviço web livre, de código aberto, desenvolvido em Python (ZOPE FOUNDATION, 2012), em outras palavras, um *framework* Python.

Zope já foi considerado o principal responsável pela popularização do Python. Não existiu ferramenta em Perl que o levasse as pessoas como o Zope levou o Python (UDELL, 2000). Entretanto o Zope se tornou muito extenso e complexo, requerendo uma alta de taxa de aprendizagem e assim sendo ofuscado pela ferramenta Django em questão de popularidade.tanto

Dado seu nível de complexidade e todas as extensões disponíveis, muitos continuam utilizando o Zope, indiferentes a sua queda de popularidade, fazendo com que suas extensões continuem surgindo, sendo mantidas em processo de atualização continua por seus mantenedores.

2.1.3.1 Zope Interfaces

Como citado na sessão 2.1.3, diversas extensões Zope foram desenvolvidas com intuito de auxiliar na criação de outras aplicações Python, de forma a não deixá-las muito extensas ou pesadas. Entre estas a *zope.interface* é um exemplo de extensão independente escrita em Python e mantida pela equipe Zope.

A *zope.interface* foi criada com intuito de permitir a comunicação entre quaisquer componentes externos que possuíssem uma *Application Programming Interface* (API). A idéia de criar uma interface para os componentes de uma aplicação é uma forma elegante de resolver um antigo problema de tipagem dinâmica tratando as informações recebidas de forma genérica, a fim de renderizá-las para um tratamento mais específico.

2.2 XML

Extensible Markup Language (XML) é uma linguagem em formato de texto simples para representação de informações sobre estruturas, sejam elas de documentos, dados, configurações, entre outros.

Esta linguagem foi derivada de um formato antigo chamado SGML, e adaptada para ser mais flexível ao uso Web (QUIN, 2010).

Atualmente se tornou um dos formatos mais comuns para compartilhamento de informações em aplicações web, por ter uma escrita simples e corresponder a um padrão similar ao HTML, que por muitos anos já vem sendo utilizado.

Além disso esta linguagem dispõe de outras vantagens como: padrão de *markup*, cujo o nome é detalhado de forma redundante; a descrição da estrutura, que de forma geral também permite uma compreensão bem literal, como se fossem textos; todas suas versões são correspondentes, isto é, mesmo que algumas delas procurem por *markups* específicos, qualquer versão mais nova de XML funcionará em uma aplicação que utilizasse uma versão anterior ou mais antiga.

2.3 Formato Aberto

O formato aberto é uma especificação publicada para armazenar dados digitais, mantido geralmente por uma organização de padrões não proprietária, e livre de limitações legais no uso.

Um formato aberto deve ser implementável tanto em software proprietário quanto em livre, usando as licenças típicas de cada um. Em oposição a esta idéia o formato proprietário é controlado e defendido por interesses particulares de empresas proprietárias detentoras de seus

direitos.

O objetivo principal dos formatos abertos é garantir o acesso a longo prazo aos dados sem incertezas atuais ou futuras no que diz respeito às direitos legais ou à especificação técnica. Um objetivo secundário dos formatos abertos é permitir a competição de mercado, em vez de deixar que o controle de um distribuidor sobre um formato proprietário iniba o uso de um produto de competição.

2.3.1 Formatos abertos de Documentos

O *.txt* é o formato aberto mais comum para arquivos de texto por ser pequeno e na maioria dos casos dispor de vários programas de edição em qualquer plataforma operacional, entretanto não é considerado o formato ideal para documentos, uma vez que não possui opções de formatação; tais como itálico e negrito.

Em 01 de maio de 2005, surgiu o *OpenDocument Format*(ODF), um conjunto de formatos para aplicações de escritório com o objetivo de padronizar os formatos abertos para documentos. Seu nome original era *Open Document Format for Office Application*, uma iniciativa da *Organization for the Advancement of Structured Information Standards*(OASIS), baseado em um XML criado por desenvolvedores do OpenOffice.org. Esta aplicação na época era uma das poucas capazes de utilizar sua estrutura.

Atualmente os formatos ODF existem a 7 anos e foram adotados por diversas aplicações. Mesmo em sotwares proprietário entre elas o Microsoft Office, mesmo sendo um software originalmente proprietário através de um *pluggin* ele permite a edição e manipulação destes formatos.

Através da figura 2.1, adaptada de (SILVA, 2010), é possível acompanhar o crescimento do uso do ODF até 2010, ano em que fez 5 anos.

> O *OpenDocument Format* ainda é uma incógnita à grande maioria dos usuários comuns, mas sua adoção cresce em várias partes do mundo, especialmente nos meios corporativos e governamentais. No Brasil, por exemplo, o ODF já conta inclusive com aprovação da Associação Brasileira de Normas Técnicas (ABNT), que aconteceu em 2008 (norma NBR ISO/IEC 26300)(ALECRIM, 2011).

2.3.1.1 Formatos de documentos ODF

O embora o mesmo padrão seja aplicado de forma geral em documentos ODF, esse padrão possui variações de extensões, no que diz respeito a documentos são elas:

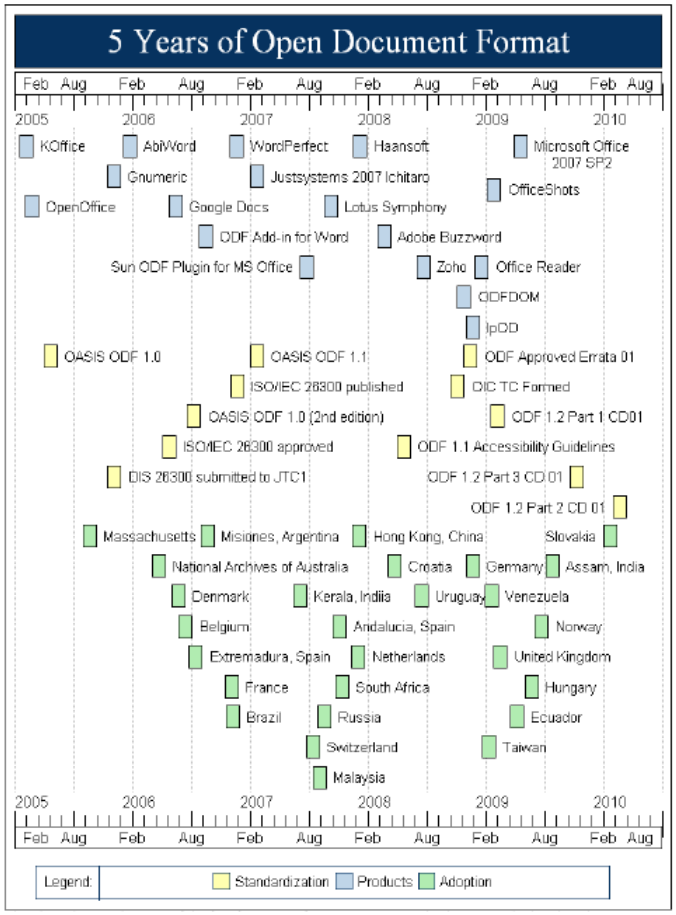


Figura 2.1: Crescimento do ODF nos primeiros cinco anos.

- Para documentos de texto: .odt, .fodt;
- Para planilhas eletrônicas: .ods, .fods;
- Para apresentações: .odp, .fodp;
- Para bancos de dados: .odb;
- Para desenhos: .odg;
- Para fórmulas: .odf.

E ainda existe um padrão a parte para modelos(*templates*), são eles:

- Para documentos de texto: .ott;
- Para planilhas eletrônicas: .ots;
- Para apresentações: .otp;
- Para bancos de dados: .otb.

Por guardar desde estrutura e dados textos até imagens presentes em seus arquivos a estrutura ODF é considerada um padrão comprimido assim como arquivo ZIP. É com base nesse padrão que diversas bibliotecas foram desenvolvidas para trabalhar com esses arquivos, entre elas a PyUno 2.4.1.1.

2.3.1.2 Estrutura de documentos ODF

Como citado na sessão 2.3.1 um documento ODF é uma estrutura de padrão aberto semelhante a arquivos comprimidos, como por exemplo arquivos ZIP.

Esta estrutura é composta principalmente por:

- mimetype: arquivo de linha única constituído pelo mimetype do documento;
- content.xml: arquivo que armazena o conteúdo criado pelo usuário do documento;
- meta.xml: arquivo responsável por armazenar os metadados do documento, ou seja, dados como autor, data de criação, data de modificação e outros;
- styles.xml: arquivo que contém os estilos do documento, tais como formatações de texto, parágrafos e outros;
- Pictures: pasta que armazena figuras existentes no documento.

Existem ainda diversos arquivos e pastas que podem compor um documento ODF, mas que não são muito referidos em uso pratico.

2.3.2 Formatos de Imagens

No que diz respeito a imagens os formatos abertos tratam sobre patentes dos formatos, ou inclusa neles.

Assim em 1996 surgiu o formato *Portable Network Graphics*(PNG) que tinha como principal objetivo substituir o formato GIF, portador de inúmeros algoritmos patenteados, que no entanto vieram a expirar em 2003.

Desde o princípio sua principal intenção foi sua utilização em qualquer aplicação sem necessidade de conflitos sobre patentes.

Além disso este formato permite comprimir imagens sem perda de qualidade e também a retirada do fundo de imagens através do canal alfa; possui suporte de milhões de cores, diferentemente do GIF, cujo suporte era de 256 cores; e ainda permite a criação de animações, cujas extensões podem variar em *.mng* e *.apng*, mas são igualmente livres.

Este formato é apoiado pela *World Wide Web Consortium*(W3C), e se tornou um padrão internacional.

2.3.2.1 Estrutura do PNG

Um arquivo PNG consiste de uma assinatura PNG e seguido de vários blocos em serie (ROELOFS, 1999).

Sua assinatura equivale aos primeiros oito *bytes*, consiste da serie “137 80 78 71 13 10 26 10”.

Cada bloco consiste de:

- length: inteiro correspondente ao numero de *bytes* dos dados da imagem;
- chunk type: código equivalente ao tipo do bloco;
- chunk data: dados da imagem;
- Cyclic Redundancy Check (CRC): campo que contém o valor total de *bytes* do bloco, o *chunk type* e o *chunk data*, mas sem o valor do *length*.

O inicio da serie de blocos deve conter um *IHDR chunk* e o ultimo bloco deve conter um *IEND chunk* como *chunk type* sinalizando que representam o inicio e fim da serie respectivamente.

2.3.2.2 Exif

O *Exchangeable Image File Format*(Exif) é um conjunto de metadados a respeito da imagem em questão, ou seja, são dados como autor, dia em que a foto foi tirada, câmera utilizada, entre outros listados conforme um padrão.

Todas essas informações ficam dentro da própria imagem, no entanto é preciso ter uma aplicação especifica para vê-lo, e em casos de informações mais abrangentes, as vezes é necessário possuir também uma aplicação para manipular a imagem e inserir nela metadados a respeito da imagem, aplicações estas como 2.5.

2.3.3 Formatos de Áudio

Os formatos abertos de áudio tratam sobre codecs disponíveis sem uma patente aplicada aos mesmos. Nesta categoria conta-se com os formatos Vorbis(.ogg) e *Free lossless Áudio Codec*(.flac).

O .flac é um formato livre de áudio comprimível e sem perda de qualidade e dados durante a o processo de compreensão. Seu algoritmo permite que o arquivo reduza em até 60% seu tamanho original, e também permite a manipulação de metadados.

O .ogg é um novo formato comprimível de áudio. É comparado de forma grosseira com outros formatos utilizados para guardar e reproduzir musicas, tais como MP3, VQF, AAC, e outros formatos de áudio digital. Mas é diferente de todos estes formatos porque é livre, aberto e sem patentes (XIPH.ORG FOUNDATION, 2012).

2.3.4 Formatos de Vídeo

Assim como os arquivos de áudio, a licença dos formatos de vídeo tratam sobre patentes de codecs, e neste caso as extensões mais comuns são conhecidas por Theora(.ogv) e Matroska(.mkv).

Theora é de propósito geral, um codec de vídeo com perda de dados. É baseado no codec de vídeo VP3 produzido pela *On2 Tecnologies* (FOUNDATION, 2011).

Matroska é o nome de uma iniciativa ousada para a criação de formatos universais de *containers* de áudio e vídeo digitais integrados, também chamados de formato de vídeo (WIKIPEDIA, 2012).

Assim o .mkv trata-se de um "contentor" de padrão aberto para vídeos, que pode conter vários dados de diferentes tipos de codificações.

2.4 LibreOffice

O LibreOffice é um pacote de software, ou seja, uma suíte para documentos de escritório, de produtividade compatível com a maioria dos pacotes semelhantes, e que esta disponível para várias plataformas. É um software de código aberto, livre para baixar, utilizar e distribuir (PARKER; JR, 2010).

Seu inicio foi marcado em 2000, quando a Sun Microsystems liberou o código de seu produto StarOffice. Neste momento se chamava OpenOffice.Org.

Em 2010 a comunidade que desenvolvia o projeto anunciou uma fundação independente, *The Document Foundation*, a fim de cumprir com a independência explicita da carta anunciada ao inicio do projeto.

Assim no inicio de 2011 foi lançado o LibreOffice 3.3, que bem como o OpenOffice.org 2.0, já suportava a edição da suíte *OpenDocument*.

2.4.1 UNO

O projeto LibreOffice possuía uma característica muito útil e pouco utilizada que era a capacidade de integrar seu funcionamento com outros aplicativos, isto é, um componente foi disponibilizada a fim de que aplicações em outras linguagens que estivessem fora do projeto pudessem interagir com o mesmo. Esse componente é conhecido por UNO (Universal Network Objects), que por sua vez é composto por um modelo dos componentes do LibreOffice.

O UNO oferece interoperabilidade entre diferentes linguagens de programação, diferentes modelos de objetos, diferentes arquiteturas e processos, em uma rede local ou mesmo através da internet. Seus componentes podem ser implementados e acessados através de *bindings* deste (PYTHON.ORG, 2008).

Atualmente existem *bindings* para as linguagens C, C++, Java e Python. Desde a versão 1.1 o LibreOffice dispõe do PyUNO em suas instalações por padrão.

2.4.1.1 PyUNO

O PyUNO representa uma "ponte" entre o LibreOffice e aplicações Python. Através dele é possível a manipulação do componente UNO, seção 2.4.1, para utilizar praticamente todas funcionalidades disponíveis no LibreOffice por *scripts* Python.

No entanto, segundo Thomas (2007), essa ferramenta ainda não atingiu seu uso absoluto podendo conter diversos *bugs*, erros, e assim dependendo em grande parte de colaboração por parte da comunidade que utiliza a mesma.

No código 2.5 é possível ver um exemplo pratico do uso do PyUNO em um código Python:

```
1  import sys
2  import os
3  import uno
4
5  # Get the uno component context from the PyUNO runtime
6  uno_context = uno.getComponentContext()
7  # Create the UnoUrlResolver on the Python side.
8  url_resolver = "com.sun.star.bridge.UnoUrlResolver"
9  resolver = uno_context.ServiceManager.createInstanceWithContext(
10   url_resolver,
11   uno_context)
12 # Connect to the running OpenOffice.org and get its context.
13 uno_connection = resolver.resolve("uno:socket,host=%s,port=%s;urp;
       StarOffice.ComponentContext" % (host, port))
14 # Get the ServiceManager object
15 return uno_connection.ServiceManager
```

Código 2.5: Exemplo de uso do Uno

Neste exemplo extraído diretamente do CloudOoo utiliza-se o contexto do processo ativo do PyUNO para retorna um serviço de gerenciamento do UNO, isto é, uma conexão é estabelecida entre o principal serviço de gerenciamento do UNO e o PyUNO utilizado pelo CloudOoo para que este possa por fim controlar as ações deste serviço.

2.5 ImageMagick

ImageMagick é uma suíte de aplicações para criar, editar,compor ou converter imagens *bitmap* (IMAGEMAGICK STUDIO, 2012). Na realidade o ImageMagick disponibiliza um conjunto de binários, compatíveis com varias plataformas, que no Linux são dados como comandos de níveis, separados para diversas funcionalidades.

Para Tesla (1994) é uma ferramenta, originalmente criada por John Cristy, para visualizar e manipular imagens, que esta amplamente disponível na Internet.

As funcionalidades que podem ser consideradas mais comuns e utilizadas são *convert* e *identify*, essas funcionalidades podem respectivamente: converter imagens para outros formatos ou formatação, como invertido ou de girar em 180 graus; e identificar os metadados disponíveis na imagem, como autor, ou data que foi criada ou tirada.

Como as demais ferramentas citadas por este trabalho, é um *software* livre, disponível para baixar e utilizar da forma desejada ao usuário.

2.6 XPDF

Xpdf é uma suíte de binários de código aberto para visualização e manipulação básica de FFMPEG arquivos *Portable Document Format*(PDF), criada por Glyph e Cog (GLYPH COG, 2011).

Além de permitir a visualização de arquivos PDF o XPDF é uma ferramenta que permite a extração de textos dentro de documentos PDF e a conversão dos mesmos para o formato *postscript*, formato este especialmente composto de informações e desenvolvido originalmente pela *Adobe System*.

Assim como a maioria das aplicações para Linux é utilizada através de comandos, neste caso o mais comum *pdftotext* o qual captura o texto disponível no documento PDF e retorna este texto através de um arquivo de texto simples.

2.6.1 Poppler

O poppler é uma biblioteca para renderizar PDF baseada no xpdf 3.0 (FREEDESKTOP.ORG, 2011).

É uma das bibliotecas de código livre mais utilizada pelos sistemas Linux para leitores PDF, seu desenvolvimento é idealizado pela FreeDesktop.Org.

2.7 PDFTk

Se um documento PDF é um trabalho eletrônico, então o PDFTk é utilizado de removedor de grampos, furador, pasta, entre outros utilitários de documentos. o PDFTk é uma ferramenta consideravelmente simples para fazer as tarefas diárias com documentos PDF (STEWARD, 2011).

Esta ferramenta esta sob licença GPL e utiliza bibliotecas que possuem suas próprias licenças de uso.

Na realidade de simples o PDFTk tem apenas sua forma de uso, pois realiza tarefas complicadas no contexto de documentos PDF. Sua confiabilidade é altamente elogiável dado que seu criador e principal mantenedor, Sid Steward, é também autor do livro “PDF Hacks”, que é considerado uma das referências do ramo.

2.8 FFMPEG

Para Zhang (2011), a melhoria constante do uso do processamento de multimídia, requerida e obtida pela expansão multi funcional e acelerada dos equipamentos de hardware, requer também aplicações eficientes e escaláveis a medida que este processo avança. E partindo deste principio uma das ferramentas ideais para projetos escaláveis é o FFMPEG.

FFMPEG é um rápido conversor de vídeo e áudio que também consegue tratar informações de ambos. Ele é capaz de converter faixas arbitrarias de amostras e redimensionar vídeos através de filtros polifásicos de alta qualidade (FFMPEG.ORG, 2012).

Mais do que isso, o FFMPEG é uma suíte de aplicações via linha de comando capaz de converter, extrair e inserir metadados em arquivos de áudio e vídeo de simples entendimento, de fácil uso.

2.9 SERVIÇOS WEB

Segundo Pirnau (2011), os serviços web representam a metodologia em que aplicações podem se comunicar através de mensagens assíncronas ou chamadas remotas. Assim pode se dizer que serviços web são aplicações acessáveis remotamente.

Toda empresa tem por objetivo prover serviços, sejam esses para própria empresa, ou para clientes que por sua podem ser outras empresas. A anos esses serviços têm sido automatizados, inicialmente aplicações *desktop* eram criadas quando a empresa era pequena e possuía poucos computadores, ou a comunicação entre elas não era tão necessária.

Quando a rede passou a estar presente no dia a dia de forma geral essas aplicações foram evoluindo e buscando a comunicação entre as mesmas.

Este conceito na verdade trata da iniciativa por parte dessas empresas de retirar suas aplicações dos computadores e passá-las para potente servidores que disponibilizarão esta na internet, assim basta ter acesso a internet e é possível utilizar esta aplicação.

2.10 XML-RPC

É um protocolo para chamadas remotas que utiliza HTTP para transporte e XML para codificação. O XML-RPC foi desenhado para ser o mais simples o possível, enquanto permite que uma estrutura de dados complexa seja transmitida, processada e retornada (SCRIPTING NEWS, 2011).

Foi originalmente criado por Dave Winer na *UserLand Frontier*, e inspirado por outros dois protocolos, um deles também desenvolvido pelo próprio Dave Winer e outro que representava o começo do protocolo SOAP. Entretanto seu uso é bem mais simples de se utilizar e entender que o SOAP. Suas mensagens correspondem a uma requisição *HTTP-POST*, enquanto composição do corpo da mensagem é escrita em XML, bem como a resposta que a requisição recebe.

Dada sua simplicidade e eficiência se tornou um protocolo muito popular e utilizado hoje nos dias atuais.

2.11 WSGI

O *Web Service Gateway Interface*(WSGI) é uma interface de entrada de dados para serviços web. É também uma especificação para que serviços e aplicações web se comuniquem com outras aplicações web, embora possa ainda ser utilizada para outras funções. É um padrão Python, escrito sob a *PEP* 333 (BROWN, 2009).

Esta interface foi escrita com o objetivo de fornecer uma forma relativamente simples e compreensiva de comunicação entre aplicações e servidores, ou pelo menos com a maioria das aplicações web em Python, e que ainda pudesse suportar componentes *middleware*.

2.11.1 Paster

Paster se trata de um componente para serviços web, composto pelo Python *Paste*, que segue o padrão da interface Python WSGI.

Possui dois níveis de linha de comando composto inicialmente por *paster*, onde o segundo comando especifica o serviço desejado, como *serve* no caso de estabelecer o servidor, seguindo como parâmetros o restante das informações necessárias para estabelecer o serviço desejado.

É considerado um dos mais simples servidores web para Python, no entanto pode ser utilizado assincronamente e manter uma escalabilidade considerável até 2000*rps*.

2.12 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.

2.12.1 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

ALECRIM, E. *OpenDocument Format*. 2011. Disponível em: <http://www.infowester.com-/odf.php>. Acesso em: 15/05/2012.

BROWN, T. *An Introduction to the Python Web Server Gateway Interface (WSGI)*. 2009. Disponível em: <http://ivory.idyll.org/articles/wsgi-intro/what-is-wsgi.html>. Acesso em: 15/05/2012.

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

FFMPEG.ORG. *Ffmpeg Documetation*. 2012. Disponível em: <http://ffmpeg.org/ffmpeg-.html>. Acesso em: 15/05/2012.

FOUNDATION, X. *Theora Specification*. [S.l.]: Xiph.Org Foundation, 2011.

FREEDESKTOP.ORG. *Poppler*. 2011. Disponível em: <http://poppler.freedesktop.org/>. Acesso em: 15/05/2012.

GLYPH COG. *XPdf*. 2011. Disponível em: <http://www.glyphandcog.com/Xpdf.html>. Acesso em: 15/05/2012.

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.

IMAGEMAGICK STUDIO. *ImageMagick: Convert, Edit or Compose Bitmap Images*. 2012. Disponível em: <http://www.imagemagick.org/script/index.php>. Acesso em: 15/05/2012.

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

PARKER, H.; JR, R. F. *Getting Started with OpenOffice.org*. São Paulo: LibreOffice, 2010.

PIRNAU, M. P. C. A. M. B. G. The soap protocol used for building and testing web services. *Proceedings of the World Congress on Engineering*, v. 1, p. 475–480, july 2011.

PYTHON.ORG. *Python Uno*. 2008. Disponível em: <http://www.python.org.br/wiki-/PythonUno>. Acesso em: 15/05/2012.

QUIN, L. R. *What is XML?* 2010. Disponível em: <http://www.w3.org/standards/xml/core>. Acesso em: 15/05/2012.

ROELOFS, G. *PNG The Definitive Guide*. United States of America: OREILLY, 1999.

SCRIPTING NEWS. *What is XML-RPC?* 2011. Disponível em: <http://xmlrpc.scripting.com-/>. Acesso em: 15/05/2012.

SILVA, J. *5 anos de ODF*. 2010. Disponível em: <http://homembit.com/2010/05/5-anos-de-odf-2.html>. Acesso em: 15/05/2012.

STEWARD, S. *PDFtk the pdf toolkit*. 2011. Disponível em: <http://www.pdflabs.com/tools-/pdftk-the-pdf-toolkit/>. Acesso em: 15/05/2012.

TESLA, B. M. Graphical treasures on the internet. *Computer Graphics World*, n. 34, november 1994.

THOMAS, R. *Python-Uno Bridge*. 2007. Disponível em: <http://www.openoffice.org/udk-/python/python-bridge.html>. Acesso em: 15/05/2012.

UDELL, J. *Zope Is Python's Killer App'*. 2000. Disponível em: <http://web.archive.org-/web/20000302033606/http://www.byte.com/feature/BYT20000201S0004>. Acesso em: 15/05/2012.

WIKIPEDIA. *Matroska*. 2012. Disponível em: <http://pt.wikipedia.org/wiki/Matroska>. Acesso em: 15/05/2012.

XIPH.ORG FOUNDATION. *What is Vorbis?* 2012. Disponível em: <http://www.vorbis.com-/faq/>. Acesso em: 15/05/2012.

ZHANG, B. Design and implementation of a scalable system architecture for embedded multimedia terminal. *International Conference on Electrical and Control Engineering*, n. 6057581, p. 618–621, 2011.

ZOPE FOUNDATION. *Zope2*. 2012. Disponível em: <http://pypi.python.org/pypi/Zope2>. Acesso em: 15/05/2012.