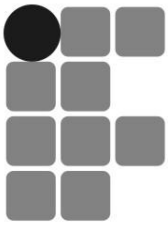


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE
Campus Campos-Centro

Secretaria de Educação Profissional e Tecnológica

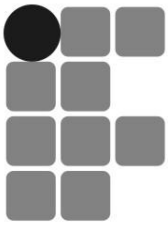
Ministério da Educação

BACHAREL EM SISTEMAS DE INFORMAÇÃO

PRISCILA MANHÃES DA SILVA

DESENVOLVIMENTO DE NOVAS FUNCIONALIDADES PARA A FERRAMENTA DE MANIPULAÇÃO DE ARQUIVOS EM NUVEM

Campos dos Goytacazes/RJ
2012



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

PRISCILA MANHÃES DA SILVA

DESENVOLVIMENTO DE NOVAS FUNCIONALIDADES PARA A FERRAMENTA DE MANIPULAÇÃO DE ARQUIVOS EM NUVEM

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. Rogério Atem Carvalho
Co-orientador: Prof. Fernando Carvalho

Campos dos Goytacazes/RJ
2012

PRISCILA MANHÃES DA SILVA

DESENVOLVIMENTO DE NOVAS FUNCIONALIDADES PARA A FERRAMENTA DE MANIPULAÇÃO DE ARQUIVOS EM NUVEM

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 29 de junho de 2012

Banca avaliadora:

Prof. Rogério Atem Carvalho (Orientador)
Doutor em Ciências de Engenharia / IFF Campus Campos
Instituto Federal de Educação, Ciência e Tecnologia Fluminense / Campus Campos Centro

---

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

---

Prof. Fábio Duncan de Souza
Mestre em Pesquisa Operacional e Inteligência Computacional / UCAM Campos
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 o desenvolvimento de novas funcionalidades para uma ferramenta livre de manipulação de documentos em larga escala, com o objetivo de permitir a mesma adaptar-se na manipulação outros formatos de arquivos através do acréscimo de aplicações comuns à sistemas operacionais baseados em linux. Assim com esta integração a ferramenta que encontra-se em constante desenvolvimento sob responsabilidade da empresa francesa Nexedi SA, com auxilio do Núcleo de Pesquisa em Sistemas de Informação(NSI-IFF), adquiriu a capacidade de manipular formatos correspondentes a arquivos de vídeo, áudio, imagens e PDF. Neste trabalho encontram-se ainda 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, Escalabilidade, Software livre, Python

ABSTRACT

This paper describes the development of new functionalities to a free tool of document manipulation on a large scale in order to allow it to adapt in handling other file formats through the addition of common applications for Linux-based operating systems. So with this integration tool that is constantly evolving under the responsibility of the French company Nexedi with the aid of the Nucleus of Research in Information Systems (NSI-IFF), acquired the ability to manipulate shapes corresponding to video files, audio, images and pdf. In this work are still used for training basic concepts of the tool and its integration with the core linux, as well as concepts of Python, which was used for its development.

KEYWORDS: Web Service, Scalability ,Free software ,Python

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Segundo (TESLA, 1994), a Internet é um produto descendente de um experimento militar americano que tentava formar uma rede que não fosse vulnerável ao ataque inimigo, e que desde seu processo de desmilitarização expandiu ao redor mundo, mesmo em continentes afastados e pouco populosos como a Antártica.

Somada essa expansão com o processo de evolução constante da Internet, onde atualmente existe um universo crescente de paginas, aplicações e arquivos amplamente disponíveis e interativos, os serviços web provaram ser uma forma rápida e consideravelmente simples de comunicação. Apesar de se provarem mais lentos que determinadas aplicações de nível de *desktop*, e com implementações mais simples. Atualmente, acompanhamos uma migração não só de tarefas antiquadas de ampla exigência manual para serviços disponibilizados em redes e nuvens.

O *cloud computing*, em português computação nas nuvens, consiste em permitir ao usuário acesso a diversas aplicações e ao máximo de funcionalidades que esta nova forma de interação entre maquina e usuário disponibiliza,independente da plataforma em que o mesmo se encontre, tornando esta interface cada vez mais ampla a medida que o avanço tecnológico permite maior e mais rápido acesso a Internet.

Seguindo esta tendência, mesmo antes de sua popularização, o Núcleo de Pesquisa em Sistemas de Informação(NSI) já trabalha anos no desenvolvimento e melhoria do projeto Biblioteca Digital da RENAPI, o qual visa disponibilizar um acervo bibliográfico digital de forma a contribuir na disseminação deste material científico tecnológico produzido na rede de Instituições de Educação Profissional Científica e Tecnológica (EPCT).

Assim o NSI passou a utilizar, entre outras, a ferramenta OpenOffice.org Daemon, a qual foi desenvolvida originalmente pela empresa francesa Nexedi SA, igualmente desenvolvedora do projeto ERP5. Essa ferramenta possuía por objetivo a conversão de documentos.

No entanto a partir do uso prolongado da mesma, foram identificados erros considerados em parte graves para uma aplicação desta extensão. Entre é possível citar perda de conexão, *deadlock* no OpenOffice.org utilizado pela aplicação, entre outros.

Assim dada a necessidade de corrigir a ferramenta a fim de obter estabilidade, através

da parceria do NSI com a Nexedi,foi realizada a analise desta que provou inviável corrigi-la em função de novas demandas também percebidas. Definiu-se nova meta, a proposta da criação de uma ferramenta que fosse capaz de manter as conversões anteriores desta, mas que tivesse seus principais erros corrigidos, e que ainda fosse capaz de extrair e manipular informações pertencentes aos documentos em questão.

Em 2010, a ferramenta Web Service OOOD 2.0 foi lançada. Seguindo em parte o nome da antecessora, esta correspondia as expectativas implícitas na ultima analise realizada, no entanto, dado que os objetivos anteriores foram alcançados, e com base no aprendizado envolvido neste desenvolvimento, novas metas foram traçadas.

Além de poder manipular documentos tornou-se desejável também que a ferramenta fosse capaz de manipular arquivos de imagem, vídeo, áudio e PDF.

1.1 Objetivo

Este trabalho tem como principal objetivo apresentar a ferramenta sucessora ao OOOD 2.0, com intuito de demonstrar suas atualizações e ampliações. O CloudOoo provou-se capaz de manipular outros formatos de arquivos, ainda que prematuramente ainda não tenha mesma estabilidade, mas que caminha para isto.

1.2 Estrutura do trabalho

Este trabaho se divide em cinco capítulos a partir deste primeiro:

O segundo capítulo deste trabalho cita e explica sobre as principais ferramentas que permitiram ao CloudOoo alcançar os objetivos inicialmente traçados, e que estão diretamente envolvidas a este, demonstrando um pouco sobre cada uma delas.

No terceiro capítulo é apresentada a nova estrutura sobre a qual o CloudOoo foi desenvolvido, e sobre sua forma de uso.

No quarto capítulo é apresentado um breve estudo de caso do desenvolvimento desta, seguindo com um estudo anterior onde esta foi utilizada para demonstrar sobre desenvolvimento de ferramentas usando metodologias ágeis.

Por fim no quinto capítulo são apresentadas as conclusões sobre o trabalho com a mesma e de proposta futuras para a melhoria desta.

2 TECNOLOGIAS EMPREGADAS

Este capítulo apresenta um conceito simplifica sobre cada ferramenta utilizada para o desenvolvimento desta aplicação, ou que utilize a mesma.

2.1 Python

Python é uma linguagem de programação poderosa e fácil de aprender. Tem estruturas eficientes e de alto nível de dados além de uma abordagem simples, entretanto eficaz para programação orientada a objeto. Sua sintaxe elegante, de tipagem dinâmica, juntamente com a sua natureza interpretada, tornam-na uma linguagem ideal para *scripts* e desenvolvimento rápido de aplicações em muitas áreas, na maioria das plataformas (ROSSUM, 2003).

2.1.1 Buildout

Buildout é uma ferramenta desenvolvida em python, capaz de criar um ambiente isolado para desenvolvimento de novas aplicações. Ele fornece suporte para criação diversas aplicações, especialmente em Python, mas não somente. Fornece ainda ferramentas para montagem de receitas para aplicações de múltiplas partes. Uma única aplicação pode conter vários programas, processos e definições de configuração (BRANDOM, 2008).

Levando em consideração o ideal do ambiente separado, também chamado de puro, a instalação do *buildout* é extremamente simples a partir de um arquivo de *script* python, que ao ser executado gera um segundo *script* 'bin/buildout' e a partir deste toda aplicação pode ser gerada com base no arquivo de configuração como do *script* 2.1:

```
1  [buildout]
2  develop = .
3  parts =
4    xprompt
5    test
6
7  [xprompt]
8  recipe = zc.recipe.egg:scripts
9  eggs = xanalogica.tumbler
10 interpreter = xprompt
11
```

```
12 [test]
13 recipe = zc.recipe.testrunner
14 eggs = xanalogica.tumbler
```

Código 2.1: Exemplo de um *script buildout*

A descrição superior *[buildout]* é a única realmente obrigatória para identificação de um arquivo de configuração. Na segunda linha definimos que o diretório em que o arquivo se encontra será um local de desenvolvimento da aplicação, nas três seguintes estão definidas suas partes, ou seja, a ordem em que cada parte, seja uma operação ou aplicação acoplada, será executada.

O *recipe* trata-se do componente responsável pela parte a qual é citado. *Egg* trata-se da opção referente as dependências de cada parte e por fim *interpreter* representa ao *buildout* que deseja-se um interpretador python contendo as dependências desta parte e que seja renomeado para *xprompt*.

Assim com poucas especificações o *script* 'bin/buildout' executa todos os passos necessários e gera uma nova aplicação através dessas especificações.

Ainda que incomum, volto a ressaltar que aplicações não desenvolvidas em python podem ter seu ambiente gerado pelo *Buildout*, basta mudar algumas especificações e ter de fato o python instalado como dependência do *buildout* e não da aplicação desejada.

2.1.2 Subprocess

O *subprocess* é um modulo python que permite a aplicação expandir processos executados pelo *shell*, ou que transmita-se informações de entrada, saída e erro de aplicações. Este módulo foi criado na intenção de substituir outros módulos e funções obsoletas, além de permitir também maior iterabilidade entre a aplicação e o sistema no qual se encontra.

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

Código 2.2: Exemplo de uso do *subprocess*

No código 2.3 existe um pequeno exemplo de como o *subprocess* pode ser utilizado, neste exemplo ele é utilizado para acessar o *shell*, como pode ser verificado no final da segunda linha em 'shell=True', e demonstrar um *Hello World* através do comando *echo*.

Em casos que aplicações utilizam outros ambientes, em muitos casos, isolados, como citado na sessão anterior sobre *Buildout*, é preferencial que a variável *shell* seja declarada como *false* na chamada da função a fim de usar as aplicações criadas neste ambiente, como na código **??**:

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

Código 2.3: Exemplo de uso do *subprocess*

Embora tenha uma descrição um pouco mais complexa, esta figura esta representando uma chamada do *subprocess* em que utiliza-se o comando *convert* em arquivos pré determinados e utiliza-se a opção *PIPE* para que após executado o comando retorne as mensagens de saída e erro respectivamente. Além disso na chamada também utiliza-se a variável *env* a fim de informar uma informação referente a ambiente de desenvolvimento da aplicação.

Assim sendo, o *subprocess* pode ser considerado um dos melhores módulos python para trabalhar com ambientes isolados.

2.1.3 Zope

Zope (Z Object Publishing Environment) é uma serviço web livre de código aberto desenvolvida em python (ZOPE FOUNDATION, 2012).

Zope já foi considerada a aplicação que popularizou o python, Não existiu ferramenta em Perl que o levasse as pessoas como o Zope levou o python (UDELL, 2000), no entanto se tornou muito extensa e complexa, com alto nível de taxa de aprendizagem, sendo mais tarde ofuscada pela ferramenta Djando em questão de popularidade. Mas essa expansão resultou na criação de vários produtos independentes que continuam facilitando na construção de aplicações python.

2.1.3.1 Zope Interfaces

Como citado na sessão anterior, diversos pacotes Zope foram criados individualmente com intuito de auxiliar na criação de outras aplicações python, de forma a não deixá-la muito extensa ou pesada, entre elas a *zope.interface* é uma aplicação independente escrita em python e mantida pela equipe Zope.

O *zope.interface* foi criado com intuito de permitir a comunicação entre qualquer componentes externos que possuissem uma *Application Programming Interface* (API). A ideia de criar uma interface para os componentes de uma aplicação é uma forma quase elegante de resolver um antigo problema de tipagens dinâmicas tratando as informações recebidas de forma genérica, a fim de renderizá-las para um tratamento mais específico.

2.2 XML

O *Extensible Markup Language* (XML) é um formato de texto simples para representação de informações sobre estruturas: documentos, dados, configurações, livros, transações, faturas entre outros. Foi derivado de um formato antigo chamado SGML (ISO 8879), 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 ser simples de ser escrito e corresponder a um padrão comum ao HTML, que por muitos anos já vem sendo utilizado pelas mesmas.

Além disso este formato dispõe de outras vantagens como: padrão de *markup*, cujo o nome é detalhado de forma redundante; a descrição da estrutura de forma geral também permite uma compreensão de código bem literal, como se fossem textos; todas suas versões são capazes de processar e ler qualquer documento XML, mesmo que algumas delas procurem por *markups* específicos.

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 tanto implementável em software proprietário quanto software livre, usando as licenças tipicas de cada um. Em contraste com o formato proprietário que é controlado e defendido por interesses particulares da empresa detentora 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 diretas legais ou à especificação técnica. Um objetivo secundário dos formatos abertos é permitir a competição, em vez de permitir que o controle de um distribuidor sobre um formato proprietário iniba o uso de um produto de competição.

2.3.1 Formatos de Documentos

Embora não seja considerado o formato ideal para documentos uma vez que não possui opções de formatação; tais como itálico e negrito, 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 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. Seu nome original era *Open Document Format for Office Application*, uma iniciativa da *Organization for the Advancement of Structured Information Standards*(OASIS) em cima da base XML criada por desenvolvedores do OpenOffice.org, na época uma das poucas aplicações capazes de utilizar

sua estrutura.

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

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

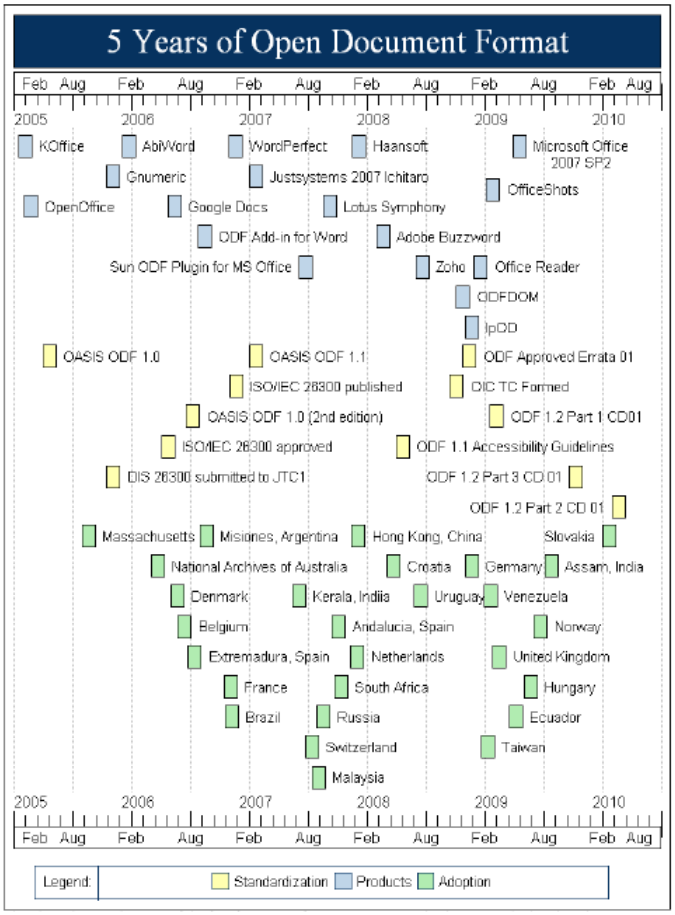
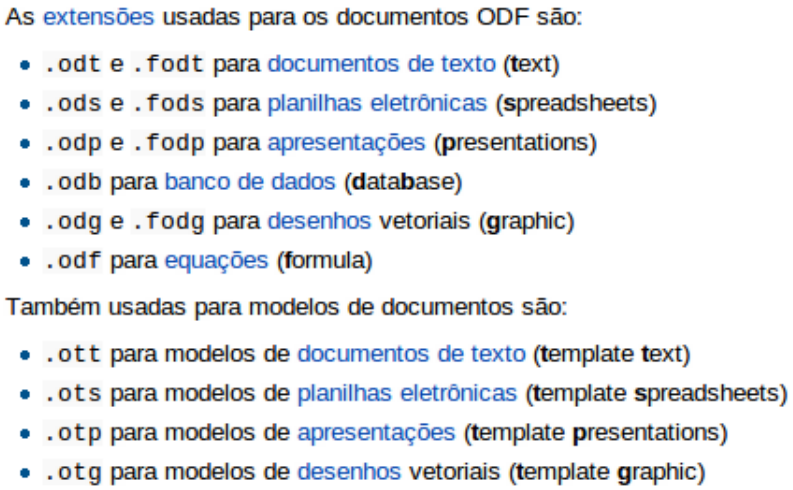


Figura 2.1: Crescimento do ODF nos primeiros cinco anos. Adaptado de (SILVA, 2010)

> 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

Embora exista um único padrão para identificação da estrutura de um ODF, como citado anteriormente, existem diversas extensões para o mesmo, como é possível observar na figura 2.2:


As extensões usadas para os documentos ODF são:

- `.odt` e `.fodt` para documentos de texto (**t**ext)
- `.ods` e `.fods` para planilhas eletrônicas (**s**preadsheets)
- `.odp` e `.fodp` para apresentações (**p**resentations)
- `.odb` para banco de dados (**d**ata**b**ase)
- `.odg` e `.fodg` para desenhos vetoriais (**g**raphic)
- `.odf` para equações (**f**ormula)

Também usadas para modelos de documentos são:

- `.ott` para modelos de documentos de texto (template **t**ext)
- `.ots` para modelos de planilhas eletrônicas (template **s**preadsheets)
- `.otp` para modelos de apresentações (template **p**resentations)
- `.otg` para modelos de desenhos vetoriais (template **g**raphic)

Figura 2.2: Extensões do ODF. Adaptado de (WIKIPéDIA, 2012b)

Além destas extensões, é possível que arquivos ODF possam ainda utilizar da extensão de arquivo comprimido ZIP, outra opção de formato aberto só que para diversos arquivos a fim de diminuir seu tamanho.

2.3.1.2 Estrutura de documentos ODF

Sendo o ODF um conjunto de formatos abertos padrão, ele é constituído de uma estrutura única a fim de que toda e qualquer aplicações que o manipule obedeça seu padrão e permita que futuramente ele possa ser reutilizado por outra aplicações ou mesmo para permitir a extração de seus dados.

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 para compor o arquivo final, assim pode-se concluir que estes formatos representam uma diferente linha de arquivos comprimidos.

2.3.2 Formatos de Imagens

No que desrespeito a imagens a questão livre trata 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.

O PNG é um formato livre, criado desde o início para ser utilizado em qualquer aplicação sem necessidade de pagamentos de licenças ou afins.

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 é apoiado pela *World Wide Web Consortium*(W3C) e o único formato realmente livre existente no momento, e em 2003, mesmo ano em que a patente do formato GIF expirou, tornou-se 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.

2.3.3 Formatos de Áudio

Os formatos livres de áudio tratam sobre codecs disponíveis sem uma patente aplicada aos mesmos. Nesta categoria conta-se com os formatos de *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*.

OGG Vorbis é um novo formato comprimível de áudio É grosseiramente comparado 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 proposito 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 contentores, ou *containers* de áudio e vídeo digitais (WIKIPéDIA, 2012a).

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 de produtividade compatível com a maioria dos softwares semelhantes, e esta disponível para varias plataformas. Ele é um software de código aberto e, por isso, é livre para baixar, utilizar e distribuir (PARKER; JR, 2010).

Este pacote teve inicio em 2000, quando a Sun Microsystems liberou o código de seu produto StarOffice, e se chamava OpenOffice.org. Em 2010 a comunidade que desenvolvia o projeto anunciou a fundação independente *The Document Foundation*, a fim de cumprir com a independência explicita da carta de 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 OpenOffice.org possui uma característica muito útil e pouco utilizada que é a capacidade de integrar seu funcionamento com outros aplicativos. Isto é possível através do UNO (Universal Network Objects), que é um modelo de componentes do OpenOffice.org.

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 por qualquer linguagem de programação que possua acesso aos *bindings* do UNO (PYTHON. . . ,).

Atualmente existem *bindings* para as linguagens C, C++, Java e Python. Desde a versão 1.1 o OpenOffice.org dispõe do pyUNO em sua instalação 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*, e assim dependendo em grande parte de coloboração por parte da comunidade que utiliza a mesma.

Na Figura 2.3 é possível ver um exemplo pratico do uso do PyUNO em um código python:

```
import uno
import unohelper
import os

# Abre o OpenOffice.org usando os parametros para que fique ouvindo na porta 2002 por novas
# conexões. O parâmetro accept é usado para que clientes tenham acesso a API do
# OO.org através da rede, seja interna ou internet.
os.system('soffice "-accept=socket,host=localhost,port=2002;urp;"')

# Retorna o componente context do PyUNO runtime
localContext = uno.getComponentContext()

# Cria o UnoUrlResolver
resolver = localContext.ServiceManager.createInstanceWithContext(
    'com.sun.star.bridge.UnoUrlResolver', localContext)

# Conecta ao OO.org em execução
ctx = resolver.resolve(
    'uno:socket,host=localhost,port=2002;'
    'urp;StarOffice.ComponentContext')
smgr = ctx.ServiceManager

# Retorna o objeto central do desktop
desktop = smgr.createInstanceWithContext(
    'com.sun.star.frame.Desktop', ctx)

# Carrega o documento
cwd = os.getcwd()
path = os.path.join(cwd, 'modelo.sxw')
url = unohelper.systemPathToFileUrl(path)
doc = desktop.loadComponentFromURL(url, '_blank', 0, ())
```

Figura 2.3: Exemplo de uso do UNO com python(PyUNO). Adaptado de (PYTHON...,)

Nela é possível observar passo a passo o código python, atentando aos comentários, neste exemplo ele utiliza o LibreOffice em *localhost*, ou seja, suas modificações são apenas realizadas na própria maquina, no entanto, no caso de um serviço web é possível passar o endereço do mesmo e alterar a porta utilizada para estabelecer comunicação.

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 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.

2.6 XPDF

Xpdf é uma ferramente de código aberto para visualização de arquivos Portable Document Format(PDF) (GLYPH COG, 2011).

Além de permitir a visualização de arquivos PDF o *xpdf* é uma ferramenta que permite a extração de textos dentro destes formatos de arquivos e a conversão dos mesmos para *postscript*, formato de arquivos especialmente compostos 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 arquivo PDF e passa para aplicação que o utilizou em forma 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 o PDF é um trabalho eletrônico, então o pdftk é o removedor de grampo, furador, pasta, anel decodificador secreto e o óculos de raio-X. Pdftk é a ferramenta simples para fazer as tarefas de todos os dias com documentos PDF (STEWARD, 2011).

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

Na realidade de simples o *pdftk* tem apenas seu uso, é uma ferramenta de fácil utilização, que no entanto permite a livre manipulação de documentos PDF, criado pelo autor do livro “PDF

Hacks", Sid Steward, é considerada uma ferramenta profissional.

2.8 FFMPEG

Para (ZHANG, 2011), a melhoria constante do uso do processamento de multimédia, requerida e obtida pela expansão multifuncional 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 ideias para projetos escaláveis é o FFMPEG.

Ffmpeg é um rápido conversor de vídeo e áudio que também consegue tratar informações momentâneas de ambos. Ele também pode 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 possui poucas maquinas, 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 da maquinas próprias 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 encodificação. 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 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.

2.11 WSGI

O *Web Service Gateway Interface*(WSGI) é uma Interface de entrada para serviços web. É uma especificação para serviços e aplicações web para que haja comunicação com outras aplicações web(embora possa 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

O *Paster* se trata de um servidor 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 versão distribuída 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 seu criador Linus Torvalds, 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 experiencia 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, ainda muito utilizado atualmente, principalmente por projetos livres.

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

2.12.2 Biblioteca Digital

A Biblioteca Digital da Rede Nacional de Pesquisa e Inovação(RENAPI), é um projeto que visa disponibilizar um acervo bibliográfico digital para contribuir com a disseminação de material científico e tecnológico produzido na rede de Educação Profissional Científica e Tecnológica(EPCT), sendo esse material periódicos, teses, monografias, artigos entre outros. Assim esse disseminação visa colaborar na qualificação do material humano digitalizado e na disseminação de conhecimento.

Este projeto é um dos principais desenvolvidos no Núcleo de Pesquisa em Sistemas de Informação(NSI), que conta atualmente com ??? bolsistas e ??? pesquisadores.

2.12.3 ERP5

Um ERP é capaz de integrar processos e dados de uma organização, através de recursos tecnológicos que padronizam e automatizam os mesmos.

Muito seja um sistema propriamente dito, ele foca mais em processos do que em funcionalidades, ele mascará informações em funcionalidades transparentes((DESAI; PITRE, 2010)).

No entanto, apesar de trazer muitas vantagens a organização, esse processo de automatização, de forma geral, é longo, de alto custo e complexidade, e até mesmo difícil de implementar.

De acordo com (SMETS-SOLANES; CARVALHO, 2003), esta situação que motivou

a criação do ERP5, cujas ferramentas são de código aberto, permitindo que a organização modifique-o a fim de torná-lo mais flexível aos seus processos.

Ele também incorpora conceitos avançados como o de banco de dados orientados a objetos, um sistema de gerenciamento, de sincronização, variação, *workflows*, e possibilita a implementação de *Business Templates*.

Compreende-se assim o ERP5, como sendo um ERP de baixo custo de implantação e alta tecnologia para pequenas e médias empresas.

2.12.4 SlapOS

O SlapOS é um sistema operacional de código aberto para o uso de redes distribuídas em computação em nuvem, que se baseia em que tudo se trata de um processo.

Para (SMETS-SOLANES; CERIN; COURTEAUD, 2011) a computação em nuvem é dividida em três camadas, infra-estrutura como serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). Na IaaS esta o funcionamento virtual da maquina e seu armazenamento, sob o ele é construido o Paas, que funciona como coração dos serviços, como servidor e bancos de dados. Finalmente sobre Paas estão as aplicações de uso do usuário.

Através de uma API unificada e simples, que requer poucos minutos para aprendizagem, este sistema combina computação em grade e o conceito de ERP para fornecer estas categorias previstas na compução em nuvem.

Dada sua abordagem unificada e arquitetura modular, ele tem sido usado como uma ferramenta de testes para benchmark de bancos de dados NoSQL e para otimização do processo de alocação em nuvem.

3 CloudOoo

Em 2006, em função da necessidade da conversão de documentos a empresa francesa Nexedi SA originou a construção da ferramenta OpenOffice.Org Daemon (OOOD 1.0), com base no uso da ferramenta LibreOffice, para conversão de documentos, no entanto com o uso prolongado desta ferramenta, em ambientes de produção, foram identificados erros relacionados com a aplicação e com a sua atuação com o LibreOffice. Entre eles estavam erros como perdas de requisições, deadlock de processo, e no LibreOffice e memory leak.

Assim foi ressaltada a necessidade de modificações a fim de que a aplicação se tornasse mais estável, além de adicionar novas funcionalidades, como de exportar documentos para outras extensões além de ODF, PDF por exemplo, e também manipular informações destes, conhecidas como *metadados*.

O resultado da continuação do desenvolvimento foi a ferramenta Web Service OOOD 2.0, posteriormente nomeada por CloudOoo, apresentada em 2010. Esta nova versão da ferramenta se provou bem mais estável no uso a longo prazo, embora fosse considerada mais lenta em processos individuais, devido aos tratamentos adicionados para os erros conhecidos, e suas modificações de novas funcionalidades.

Ao final do processo de melhoria da ferramenta novas funcionalidades foram idealizadas para diferentes tipos de arquivos, como converter e manipular documentos, bem como arquivos de áudio, vídeo, imagem e PDF.

Assim o CloudOoo apresentado neste capítulo, é um serviço Web livre e de código aberto, sob a licença LGPL, que foi desenvolvido através parceria da Nexedi SA e do NSI, na linguagem de programação Python e que utiliza o protocolo XML-RPC para troca de mensagens, que pode ser utilizado inteiramente ou em partes separadas.

3.1 Estrutura

Desde de sua estrutura anterior, o CloudOoo foi desenvolvido para trabalhar de forma genérica prevendo futuras mudanças. Sua estrutura contém com estas interfaces:

- IApplication: representa os métodos de controles as aplicações externas do servidor;

- IFile: que representa métodos para manipulação de arquivos recebidos pelo servidor;
- IOdfDocument: representa métodos de manipulação de documentos ODF recebidos pelo servidor;
- IHandler: representa os objetos que irão realizar a requisição emitida pelo cliente;
- IMonitor: representa métodos de controle e manuseio dos processos estabelecidos no servidor;
- IMimemapper: representa métodos utilizados para trabalhar com filtros;
- IFilter: representa métodos de tratamento de filtros;
- ILockable: representa os métodos de controles criados para região crítica do servidor;
- ITableGranulator: representa métodos para extrair tabelas de documentos;
- IImageGranulator: representa métodos para extrair imagens de documentos;
- ITextGranulator: representa os métodos para extrair o conteúdo de um documento em capítulos e parágrafos;
- IERP5Compability: representa os métodos de compatibilidade com o ERP5;
- IManager: representa os métodos utilizáveis entre cliente e servidor.

As próximas subseções apresentam detalhadamente sobre cada interface e as principais classes a implementá-las.

3.1.1 IApplication

Por possuir a opção de estar instalado num ambiente própria onde tanto o CloudOoo, assim como as ferramentas utilizadas pelo mesmo, podem possuir instalação própria a partir do uso do *buildout*, foi preciso construir uma interface para controlar as funções dos processos utilizados pela aplicação, ou seja, uma classe que fosse capaz de carregar a configurações das aplicações, controlar a inicialização e finalização de cada processo, e que fosse capaz de verificar se continuavam rodando no sistema operacional, a partir de um identificador e/ou da porta que cada uma utilizasse.

3.1.1.1 Application

Esta classe implementa a interface IApplication e tem por objetivo controlar aplicações que estejam dentro do processo do cloudooo, como o LibreOffice que precisa estar executando para possibilitar a manipulação dos documentos.

Ela possui os métodos responsáveis por carregar as configurações, iniciar e parar o processo, igualmente o método de reiniciar a instancia, utilizado principalmente caso esta apresente algum erro durante processos; um método para verificar o status dessa aplicação através do método de pid que retorna o pid utilizado pela aplicação.

Além desses métodos existe um responsável por endereço dessa aplicação, ou seja, onde esta estabelecida e em qual porta.

3.1.2 IFile

Esta interface propõe um contrato de tratamento de arquivos, a fim de assegurar uma resposta eficiente e consistente ao cliente. Nela são contidos métodos para que o conteúdo do arquivo seja guardada durante sua instanciação de forma no acontecimento de erros não previstos este possa ter seu conteúdo recuperado, ou mesmo restaurado a forma original.

3.1.2.1 File

Com base na implementação da interface IFile, esta classe possui métodos para manter qualquer arquivo recebido do cliente no sistema apenas durante o uso do mesmo. Ao receber um arquivo ela escreve o mesmo no disco, podendo assim recuperar seus dados, e obter informações do mesmo, como seu caminho por exemplo.

Após o uso e manipulação deste ela é instituída a removê-lo do sistema, como demais métodos de tratamento de arquivos temporários.

3.1.3 IOdfDocument

Embora muito similar a interface IFile, descrita na seção 3.1.2, neste caso o tratamento é especifica para arquivos do tipo ODF, dada sua complexidade de armazenamento e manipulação.

3.1.3.1 Document

Por se trata de uma aplicação livre o foco dos tipos de arquivos do CloudOoo é igualmente livre, assim sua estrutura é relativamente planejada para estes, e no caso de documentos ODF, sua estrutura complexa e compacta exigiu que diversas classes fossem criadas especificamente para estes. É o caso desta classe, em seus métodos existe por exemplo o *getContentXml* responsável pela leitura da estrutura XML desses tipos de documentos.

Outro razão para existência dessa estrutura vem a ser o tempo maior de estudo de manipulação de documentos em função dos outros tipos de arquivos.

3.1.4 IHandler

Esta interface foi especificamente criada para estabelecer o contrato entre as aplicações externas utilizadas pelo servidor em função dos pedidos do cliente no que desrespeito a manipulação direta do arquivo, como no caso da conversão por exemplo, bem como a extração e inserção de *metadados* do mesmo.

Para as classe que implementam esta interface é recomendado o igual uso de objetos do tipo File, apresentado na seção 3.1.2.1, para manipulação dos arquivos utilizados nas mesmas.

3.1.4.1 OOHandler

Dado o nome o OOHandler é a implementação especifica de comunicação com o LibreOffice. Ao receber uma requisição que trata de documentos é gerada uma instancia desta classe para manipulá-lo.

3.1.4.2 PDFHandler

Da mesma forma ocorre entre a classe PDFHandler e os arquivos do tipo PDF. Ao ser recebido pelo Manager, este arquivo será tratado diretamente nesta classe.

3.1.4.3 IMAGEMAGICKHandler

No que se trata de ferramentas para imagens o ImageMagick é uma das melhores disponível a nível de comando, ela consegue inclusive manipular dados do tipo *Exif*, que são *metadados* referentes a imagens.

Assim com base na aplicação foi escolhido o nome para esta classe responsável pelos arquivos de imagem.

3.1.4.4 FFMPEGHandler

No que se trata de arquivos de vídeo e áudio existe a ferramenta FFMPEG que é capaz de manipular ambos e portanto representa a classe responsável por estas instancias.

Assim como as demais classes que implementam o IHandler, esta classe é capaz de manipular os *metadados* desses arquivos, bem como convertê-los para determinadas extensões do mesmo tipo.

3.1.5 IMonitor

Esta interface foi desenvolvida principalmente com base nos erros anteriormente obtidos com o uso do LibreOffice, no entanto é importante para o sistema como um todo, seu uso estabelece controles sobre princípios básicos do sistema, como uso de memória, tempo de requisições, tempo de uso do processo, entre outros.

3.1.5.1 Monitor

Basicamente a classe Monitor funciona como uma simples implementação da IMonitor a fim de estabelecer os atributos principais, como por exemplo o tempo minimo entre a monitorações do sistema. As próximas subseções explicam detalhadamente sobre estes controles através da herança desta classe.

3.1.5.2 MonitorMemory

Nas configurações do CloudOoo existem definições que podem ser modificadas de acordo com o sistema em que vai ser instalado, entre elas existe uma variável responsável pelo uso máximo de memória pelo LibreOffice.

A partir dessa definição, dada em *megabytes*, essa classe monitora o uso da memória do sistema, assim caso esta chegue em seu limite máximo de uso, a aplicação é reinicia com intuito de limpar da memória mensagens trocadas e que não liberaram o uso da mesma, evitando assim o evento chamado *memory leak*, o qual consiste no uso de toda memória do sistema.

3.1.5.3 MonitorTimeout

Também entre as definições da subseção 3.1.5.2 existe uma variável responsável pelo tempo limite de execução de um determinado processo, caso este tempo seja excedido ocorre o que chamamos de *timeout* e a aplicação é forçada a parar, sendo reiniciada posteriormente.

A utilidade desta limitação é dada pela ideia de que caso este tempo tenha excedido ocorreu algum erro durante o processo, provavelmente em função da resposta da aplicação, assim reiniciá-la pode resolvê-lo.

3.1.5.4 MonitorSleepingTime

Com intuito de poupar uso do sistema em momentos desnecessários esta classe foi criada para observar o momentos de inutilização da aplicação e após determinado tempo parar a mesma. Assim é possível economizar no uso de recursos, e disponibilizá-los para outras aplicações que possam vir a utilizar o mesmo.

3.1.5.5 MonitorRequest

A fim de conservar a estabilidade do CloudOoo, esta classe implementa um controle em função do valor máximo de requisições que podem ser respondidas por cada instancia da aplicação, bem como nas subseções 3.1.5.2 e 3.1.5.3 a variável de reaquisições limite é estabelecido nas configurações.

Caso o valor informado seja excedido a instancia é parada e encerrada, em seguido uma nova instancia da mesma aplicação é iniciada.

3.1.6 IMimemaper

Em casos de aplicações como o LibreOffice que representam uma suíte de menores utilitários é preciso reconhecer a extensão de arquivo especifica para cada utilitário. De forma geral estas extensões são explicitas no nome do arquivo, entretanto no caso de que esta não esteja explicita é preciso reconhecer o tipo de arquivo de alguma outra forma. Neste caso existem o *mimetypes*, que são identificações presentes no conteúdo do arquivo, que permitem decidir sua extensão. Esta interface propõe métodos ara lidar com a identificação desses *mimetypes*.

Seu exemplo de uso é a classe Mimemapper a qual será apresentada na próxima subseção.

3.1.6.1 Mimemapper

A despeito de suas aplicações internas, o CloudOoo possui seus próprio filtros para identificar e renderizar arquivos, como vista na seção 3.1.4, no entanto dada a necessidade dessas aplicações também tornou-se necessário identificá-los de forma a torná-los igualmente reconhecível para a aplicação especifica.

No caso do LibreOffice é possível, através do uso do UNO, extrair os *mimetypes* e demais informações sobre cada um deles, caso necessário durante sua manipulação.

3.1.7 IFilter

Conforme explicado na seção anterior, 3.1.6.1, cada filtro pode ter demais propriedades que ao serem requisitadas precisam estar disponível de forma facilmente utilizável, com base neste principio esta interface propõe um contrato para trabalhar com os filtros mais complexos da melhor forma possível e ,se possível, da forma igualmente mais simples.

3.1.7.1 Filter

Se comparada as outras classes e suas devidas interfaces, a classe Filter pode ser considerada a que representa o métodos mais simples. Ao ser iniciada ela guarda todos os dados dos filtro em diversos atributos que foram selecionados prevendo seu uso posterior na aplicação.

3.1.8 ILockable

Quando se possui-se um recurso compartilhado, isto é, um recurso, que pode ser outra aplicação, rodando em paralelo a aplicação principal, admiti-se também existir uma região crítica.

Esta visão ocorre devido ao fato que determinadas aplicações não conseguem atender a mais de um processo simultaneamente, assim é necessário controlar essa região crítica prevendo travá-la caso um processo já esteja em uso da mesma, para isto foi implementada a interface ILockable.

3.1.8.1 OpenOffice

A classe OpenOffice foi exclusivamente criada visando controlar a aplicação LibreOffice, anteriormente conhecida por OpenOffice.org, a qual foi responsável por maior parte dos erros respondidos do CloudOoo. Esta classe estende o uso da interface IApplication, através da classe descrita na seção 3.1.1.1 e ainda implementa a interface anterior, ILockable, para que assim seja possível controlar a aplicação e ainda trancá-la e destrancá-la durante seu processo de uso.

A partir do método *isLocked* é possível verificar se esta aplicação está trancada ou disponível para uso, evitando assim erros, como o *deadlock* por exemplo.

3.1.9 ITableGranulator, IImageGranulator, ITextGranulator

Uma das principais funções desenvolvidas para documentos foi a *granularização*, ela trata da extração de partes importantes do documento que não sejam especificamente texto, como por exemplo tabelas e imagens. Dada a complexidade dessa tarefa foi necessário a implantação das novas interfaces, para que que os processos fossem realizados.

A interface ITableGranulator é a interface responsável pelo processo de granularização específico de tabelas presentes nos apenas em documentos. Ela implementa funções respectivas a uma tabela comum, baseado nas linhas e colunas que a mesma possua.

Na interface IImageGranulator ocorre a granularização das imagens presentes nestes documentos, que são extraídas em seu formato original. Por fim através da interface ITextGranulator é possível partir o documento em textos menores dividindo o mesmo a partir de seus parágrafos, ou mesmo em capítulos.

Este processo ocorre com base em namespaces definidos anteriormente com base no estudo desses documentos.

3.1.10 IManager

A IManager trabalha como a interface entre o cliente e servidor. Detém um padrão genérico para troca de informações entre diferentes tipos de arquivos, inclusive vídeos. Ele possui os principais métodos para funcionalidades do CloudOoo, no que desrespeito a todos seus *handlers*, ou seja, módulos para tratar diversos tipos de arquivos.

3.1.11 IERP5Compability

Esta interface estabelece as funcionalidades entre cliente e servidor especificamente para o uso da aplicação ERP5. Ela rescreve a forma dos métodos da aplicação OpenOffice.org Daemon para utilizarem os novos métodos sem interferir nas requisições feitas pelo uso do ERP5, e outros possíveis clientes que utilizassem a antiga plataforma.

3.1.11.1 Manager

A classe Manager implementa as classes IManager, IERP5Compability, ITableGranulator, ITextGranulator e IImageGranulator com o proposito de interligar suas funcionalidades ao cliente que venha a requiri-las, em outras palavras esta classe é a principal responsável pela conectividade entre cliente, servidor, aplicação e funcionalidades.

Nela são absorvidos os dados importantes para o funcionamento do CloudOoo, como

por exemplo a base de *mimetypes*, os *handlers* disponíveis na mesma, a pasta de trabalho da aplicação, entre outros dados.

Assim a partir desta classe é possível iniciar o servidor do CloudOoo.

4 ESTUDO DE CASO

Este capítulo apresenta um estudo de caso do CloudOoo e sua implantação num ambiente limpo.

Para este estudo foram escolhidas duas formas de instalação: a primeira a partir do uso do SlapOs, e a segunda a partir do uso direto do *buildout*. Não existe praticamente diferença entre ambas as instalações, exceto pelo tempo que levam, e configuração final.

Além disso este caso de uso também apresenta uma avaliação quanto a forma de desenvolvimento do CloudOoo em comparação a sua versão anterior e do uso de metodologias ágeis.

4.1 Ambiente de desenvolvimento

A parte de estudo apresentada neste trabalho foi desenvolvida no NSI e contou com a disponibilidade de uma maquina com processador Intel Core I5 CPU 650 3.20 x4, 4GB de memória RAM, 320GB de HD.

Como sistema operacional foi selecionado o Ubuntu 12.04, utilizado praticamente em todas as maquinas do ambiente.

Para instalação via SlapOs, definimos a instalação do próprio como pré-requisito, sendo este dependente de que o sistema disponha do Python instalado, em qualquer versão pois o SlapOs instala seus requisitos de funcionamento.

Da mesma maneira o Python 2.6 e o git são os únicos requisitos para instalação pelo uso do *buildout*, pois através dele e do uso da biblioteca *boostrap* é possível gerar o *script bin/buildout*.

4.2 Processo de instalação

Como citado anteriormente toda a estrutura é gerada pelo *Buildout*, e depende da existência de uma inalação do Python no sistema, neste caso foi utilizado o Python 2.6 em ambas.

4.2.1 Instalação via SlapOs

Esta instalação foi realizada a partir da modificação do tutorial de instalação do ERP5 no SlapOs, disponível em `http://www.erp5.com/user-Install.ERP5.With.SlapOS/user-Install.ERP5.With.SlapOS`, o qual passa por modificações frequentes em função da atulização desta ferramenta.

Nas próximas subseções será representado o tutorial encontrado nesta pagina, modificando porém que invés da instalação do ERP5, será modificado para instalar o CloudOoo.

4.2.1.1 Instalação do SlapOs

Está instalação é padrão para qualquer ferramenta que utilize o SlapOS.

O SlapOS é instalado na pasta raiz do sistema, assim requer privilégios para instalação. Assim como é possível notar na figura 4.1, cria-se uma pasta para instalação em */opt/slapos*, e também um arquivo de configuração do **??**, após a criação com o uso do python é feita a execução do *bootstrap* próprio da Nexedi.

```
$ sudo mkdir /opt/slapos

$ cd /opt/slapos
$ touch buildout.cfg
$ vi buildout.cfg

[buildout]
extends =
    http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.57:/component/slapo

$ sudo python -S -c 'import urllib2;print urllib2.urlopen("http://www.nexedi.org/static/\
packages/source/slapos.buildout/bootstrap-1.5.3-dev-SlapOS-002.py").read()' | python -S -
$ sudo bin/buildout -v
```

Figura 4.1: Primeira parta da instalação do SlapOS

Com o termino do *bootstrap* e o *script bin/buildout* é feita a instalação do SlapOS.

Após a instalação basica dos componentes e dependências é preciso configurar o mesmo, na figura 4.2, existe um exemplo de arquivo de configuração e logo em seguida na figura 4.3, é feita a configuração de rede para uso do SlapOS, é necessário que o *slapproxy* seja iniciado e mantido rodando, ele representa funcionalidades do *master* do SlapOS.

É valido recordar que o SlapOS é uma ferramenta já adaptada ao IPV6, e é necessário adicionar uma configuração extra ao *script /etc/network/interfaces* para habilitar as funcionalidades IPv6 do SlapOS. Por fim nas duas ultimas linhas da figura 4.3 são executados os *scripts slapformat e slapgrid*, eles são responsáveis por registrar seu computador e devidos *slots* e por habilitar o cliente SlapOS em seu computador, respectivamente.

```
$ touch slapos.cfg

  [slapos]
  software_root = /opt/slapgrid
  instance_root = /srv/slapgrid
  master_url = http://127.0.0.1:5000/
  computer_id = vifibnode

  [slapformat]
  computer_xml = /opt/slapos/slapos.xml
  log_file = /opt/slapos/slapformat.log
  partition_amount = 10
  bridge_name = br1331
  partition_base_name = slappart
  user_base_name = slapuser
  tap_base_name = slaptap
  # You can choose any other local network which does not conflict with your
  # current machine configuration
  ipv4_local_network = 10.0.0.0/16

  [slapproxy]
  host = 127.0.0.1
  port = 5000
  database_uri = /opt/slapos/proxy.db
```

Figura 4.2: arquivo de configuração do SlapOS

4.2.1.2 Instalação do CloudOoo

Após realizada a instalação do SlapOS, o passo de instalação de ferramentas através do mesmo é consideravelmente simples.

Para requerer a instalação do CloudOoo é necessário utilizar o *slapconsole*, assim através do mesmo e uma instância do SlapOS(*slap*), referenciada na terceira linha da figura 4.4, assim como no uso do *bootstrap*, esta instalação também aponta para um link que é carregado e compilado.

Nesta referência em particular o ponteiro esta voltado para o *branch* do CloudOoo, em função de suas configurações próprias que não foram encorporadas ao projeto do SlapOS.

Por fim para que ocorra a instalação dos componentes e dependências é utilizado o *script slapgrid-sr*.

Após a instalação é necessário requerer uma instância, o que também ocorre pelo uso do *slapconsole* de forma praticamente identica a instalação, no entanto é necessário fornecer um título a sua instância e é possível verificar se a mesma foi criada requerindo seu *id*.

Por fim é utilizado o *script slapgrid-cp* para que a criação da instâcia ocorra e seja finalizada. Após o uso da mesma será iniciado um servidor do CloudOoo pelas configurações estabelecidas no software.

```
$ sudo bin/slapproxy -vc /opt/slapos/slapos.cfg
$ sudo brctl addbr br1331
$ sudo ip l s dev br1331 up
$ sudo ip a a dev br1331 fd00::1/64
$ sudo vi /etc/network/interfaces

  auto br1331
  iface br1331 inet6 static
      address fd00::1
      netmask 64
      bridge_ports none

$ sudo bin/slapformat -c /opt/slapos/slapos.cfg
$ sudo bin/slapgrid -c /opt/slapos/slapos.cfg
```

Figura 4.3: Configurações de rede do SlapOS

```
1  $ bin/slapconsole /opt/slapos/slapos.cfg
2  ..import slapos.slap.slap
3  ..slap = slapos.slap.slap()
4  ..# Conecta com slapproxy
5  ..slap.initializeConnection('http://127.0.0.1:5000/')
6  ..# Requisição para aplicação cloudooo
7  ..slap.registerSupply().supply(
8  ......'http://git.erp5.org/gitweb/slapos.git/blob_plain/cloudooo:/software/cloudooo/softw
9  ......computer_guid='vifibnode')
10
11 $ sudo bin/slapgrid-sr -c /opt/slapos/slapos.cfg
```

Figura 4.4: Requisição de instalação do CloudOoo no SlapOS

4.2.2 Instalação do CloudOoo.git

Esta segunda instalação foi criada para ser mais flexível aos projetos do NSI que também utilizam o CloudOoo. Ela esta disponível no Github do NSI, em `http://github.com/nsi-iff/cloudooo_buildout.git`, e possui um README com instruções para instalação. Além disso esta instalação esta orientada a fazer download do repositório igualmente disponível no Github do NSI, em `http://github.com/nsi-iff/cloudooo`.

São necessárias apenas duas instruções para esta instalação, presentes na figura 4.6:

Na primeira etapa utiliza-se o Python para fazer download e rodar o *script* do *boostrap.py*, este irá fazer o download e instalação dos componentes necessários para gerar o buildout.

Ao termino da primeira, com o *script bin/buildout* acrescido de argumentos para torná-lo verboso e do arquivo padrão de instalação(*buildout.cfg*), ocorre a instalação do CloudOoo e todos os componentes necessários, entre ele o LibreOffice, FFMPEG, ImageMagick e demais.

Diferentemente da instalação via SlapOs o servidor do CloudOoo não fica disponível automaticamente para uso, é preciso ainda utilizar o *script bin/supervisord*.

```
1  $ bin/slapconsole /opt/slapos/slapos.cfg
2  ..import slapos.slap.slap
3  ..slap = slapos.slap.slap()
4  ..slap.initializeConnection('http://127.0.0.1:5000/')
5
6  ..cloudooo_computer_partition = slap.registerOpenOrder().request(
7  ......'http://git.erp5.org/gitweb/slapos.git/blob_plain/cloudooo:/software/cloudooo/soft
8  ......'Minha instancia do CloudOoo')
9  ..# Como teste requisite o id da aplicação
10 ..cloudooo_computer_partition.getId()
11
12 .$ sudo bin/slapgrid-cp -c /opt/slapos/slapos.cfg
```

Figura 4.5: Requisição de uma instância do CloudOoo via SlapOS

```
1 Buildout for cloudooo
2 ======================
3
4 How to Install
5 --------------
6
7 python -S -c 'import urllib2; exec urllib2.urlopen("http://python-distribute.org/bootstra
8
9 bin/buildout -vv
```

Figura 4.6: README do buildout do CloudOoo, disponível em `http://github.com/nsi-iff/cloudooo_buildout`

4.3 Processos de requisições

Para estabelecer conexão entre um cliente qualquer e o CloudOoo é necessário o uso de uma biblioteca XMLRPC. Na figura 4.7 foi realizado um exemplo de cada requisição básica disponível para todos *handlers*, através de uma conexão estabelecida pela biblioteca *xmlrpclib*:

Na linha 6 da figura a variável *arquivo* recebe o conteúdo do arquivo *test.ogv*, no entanto para conversão do mesmo, na linha 8, é preciso codificá-lo para que durante a passagem do cliente para o servidor esse arquivo não seja danificado, esta codificação realizada pelo uso da biblioteca *base64*.

No momento da conversão o servidor vai receber os dados do cliente, vai decodificar o arquivo e identificá-lo como um arquivo de vídeo, sendo assim o mesmo será encaminhado para o *FFMPEGHandler*, que por sua vez irá convertê-lo para o formato *mpeg*.

o *FFMPEGHandler* também será o responsável pelos métodos de *getFileMetadataItemList* e *updateFileMetadata*, nos quais serão realizados respectivamente a extração e inserção de *metadados* no arquivo.

Observe também que na segunda extração de *metadados*, linha 17, foi utilizado o $arquivo_atualizado$,

Além destas requisições comuns a todos os handlers existem também requisições que

```
>>> from base64 import encodestring
>>> from xmlrpclib import Server

>>> conexao_CloudOoo = Server("http://localhost:23000")

>>> arquivo = open("test.ogv").read()

>>> convercao = conexao_CloudOoo.convertFile(encodestring(arquivo), 'ogv', 'mpeg')

>>> info = conexao_CloudOoo.getFileMetadataItemList(encodestring(arquivo), 'ogv')

>>> info
...{'Encoder': 'Lavf52.64.2'}

>>> arquivo_atualizado = conexao_CloudOoo.updateFileMetadata(encodestring(arquivo), 'ogv

>>> info = conexao_CloudOoo.getFileMetadataItemList(arquivo_atualizado, 'ogv')

>>> info
...{'Encoder': 'Lavf53.4.0', 'Titulo': 'Test setMetadata'}

```

Figura 4.7: Exemplo prático de uso do CloudOoo

foram previamente implantadas apenas para documentos:

- *getAllowedExtensionList: retorna extensões permitidas para determinado arquivo;*
- *getChapterItem: retorna o capítulo selecionado do documento;*
- *getChapterItemList: retorna todos capítulos do documento;*
- *getColumnItemList: retorna colunas da tabela selecionada;*
- *getImage: retorna imagem selecionada;*
- *getImageItemList: retorna lista de imagens em documento;*
- *getLineItemList: retorna lista de linhas da tabela selecionada;*
- *getParagraph: retorna parágrafos selecionado;*
- *getParagraphItemList: retorna lista de parágrafos de um documento;*
- *getTable: retorna tabela selecionada;*
- *getTableItemList: retorna lista de tabelas no documento;*
- *system.listMethod: retorna lista de métodos disponíveis no servidor.*

4.4 Uso do CloudOoo na Biblioteca Digital

Para disponibilizar seu acervo, a Biblioteca Digital tem como regra converter os arquivos recebidos seu formato livre compatível, para isso utiliza de requisições ao CloudOoo. Na figura 4.8, existe um exemplo de submissão de arquivos, do tipo ???, o qual passára pela conversão através de uma requisição ao CloudOoo:

Figura 4.8: Pagina de submissão de documentos da Biblioteca Digital

Na figura não é implicito o uso da ferramenta, uma vez que essa requisição só é realizada por formatos não livres. Outra funcionalidade implicita é a "granularização" realizada igualmente pelo CloudOoo. Desmonstrada na figura 4.9:

Figura 4.9: Pagina de submissão de documentos da Biblioteca Digital

Os grãos extraidos pelo processo de "granularização" são separados em imagens e tabelas ???, e dispostos para visualização do usuário.

4.5 Performasse

Para estabelecer a performasse o os antecessores do CloudOoo foi realizado um teste em ambos, neste teste 3699 documentos no formato odt foram selecionados, entre eles alguns documentos inválidos e outros em formatos desconhecidos. Por ser a conversão mais utilizada, foi decido que neste teste os documentos seriam convertidos para PDF.O teste foi realizado por meio do uso de script, neles além de prover a conversão também era provido o armazenamento do tempo de cada conversão, bem como o tempo total que todas as conversões levaram em ambos.

Ao final do teste notou-se que o OOOD 2.0 levará mais tempo para realizar as conversões separadamente, no entanto com o uso excessivo ele se mostrou mais estável e mais rápido, levou 10 horas para realizar o teste e apresentou 12 erros, enquanto o OOOD 1.0 apresentou 531 erros e levou 11 horas para realizar o teste.

Para testar a performasse do CloudOoo 1.24 foram escolhidos dois testes distintos, primeiramente a repetição do teste anterior com 3699 documentos odt convertidos para PDF, para reanalisar sua performasse após suas mudanças e por segundo um novo teste com 3500 arquivos distintos sendo todos convertidos para formatos abertos equivalentes aos seus tipos específicos.

[resultado dos testes] ???

4.6 Processo de Desenvolvimento

Para (PRESSMAN, 1995), a garantia da qualidade de software de esta diretamente ligada ao emprego de testes sobre o mesmo, onde esses testes representam a expectativa do usuário sobre a aplicação, e podem ser empregados de forma automatizada ou manual. Embora testes automatizados tendam a gastar mais tempo em relação a programação dos mesmos, sua cobertura sobre o produto garante menor porcentagem de erros quando executada a aplicação.

Desde o inicio de parceria de desenvolvimento do CloudOoo tem sido aplicada a técnica chamada TDD(Test-Driven Development), ou desenvolvimento orientado a testes, nela é defendido o desenvolvimento dos testes antes do desenvolvimento da parte funcional da aplicação, dado que os testes também fazem parte da mesma.

Segundo (ASTELS, 2003), projetos que utilizam TDD devem possuir uma suíte exaustiva de testes que por sua vez determinam o código que deve ser escrito. No entanto para uma aplicação de serviço web, existe certo grau de dificuldade de começar do zero apenas com testes, tendo por justificativa suas dependências como outras aplicações bem como a rede utilizada por este. Mesmo assim desde a parceria implementada ao CloudOoo anteriormente, até seu estado atual seu desenvolvimento tem partido do principio de realizar testes antes de implementar suas funcionalidades. Isto porque, além da garantia de qualidade, o uso de testes nele serviu de auxilio ao desenvolvimento dado todos as configurações que devem ser estabelecidas para funcionamento deste. Entre elas o próprio padrão das aplicações externas que não estão ligadas diretamente ao sistema operacional do servidor e que precisam ser apontadas por meio de scripts.

Além do testes, outra ferramenta que garante o desenvolvimento do CloudOoo é o uso de sistemas de controle de versão, como o subversion, que era utilizado na versão 2.0, e que atualmente foi trocado pelo git em função de suas diversas vantagens como o controle de versões distribuído por exemplo. A importância do controle de versão esta no controle do crescimento desta aplicação, que por momentos contou com uma equipe de desenvolvimento sem contato constante, assim por meio de ferramentas como estas podiam acompanhar as modificações da aplicação, bem como reverter determinadas alterações em casos de erros posteriores na mesma.

Sob certo ponto de vista o uso destas práticas sugere sobre o desenvolvimento do CloudOoo severas semelhanças com os métodos ágeis, muito embora não exista a definição propriamente dita de nenhuma delas aplicadas diretamente sobre o projeto.

Em seu artigo (SILVA; MONNERAT; CARVALHO, 2010), comprovam o uso do TDD no CloudOoo, e de suas complicações de emprego, uma vez que inicialmente não existia total proficiência por parte dos desenvolvedores. Além disso foi verificado sobre o estabilidade do desenvolvimento do projeto, tomando por base a lista de mudanças armazenadas pelo controle

de versão. Observou-se através deste artigo que embora no incio do projeto nenhum teste falhasse, conforme o mesmo foi acrescido de mudanças e novas funcionalidades testes que antes passavam passaram a apresentar erros antes não conhecidos, precisando assim de correções.

Estas implicações trazem ao meio de software uma compreensão muito similar ao do ramo de indústria, no que se desrespeita a aplicação de novas técnicas para garantir que quando um produto chega ao meio de produção possa continuar estável ao uso.

5 CONCLUSÕES

5.1 Objetivos alcançados

Neste trabalho foi possível apresentar de forma simplificada porém descritiva os processos por trás do CloudOoo, uma aplicação livre e de código aberto, que encontra-se a disponível acesso.

Também foi descrito sobre sua instalação e uso como ferramenta de conversão e manipulação de arquivos comuns aos tipos de documentos, imagens, vídeos, áudio e PDF.

Foi possível ainda demonstrar mais sobre as ferramentas que possibilitaram este trabalho, e explicitar sobre suas facilidades de instalação e uso, bem como da facilidade de associá-las e utilizar diversas funcionalidades das mesmas através da linguagem Python.

5.2 Trabalhos futuros

Apesar deste trabalho representar em grande parte a realização de objetivos propostos por um trabalho anterior com a aplicação CloudOoo, nota-se a necessidade de modificações futuras ao mesmo.

Entre elas visar maior estabilidade do projeto não só por meio dos testes implementados e seus acréscimos, bem como pela revisão da escrita do projeto em função das novas funcionalidades adquiridas.

Umas vez que os tipos de arquivos atendidos pelo mesmo foram expandidos também há o interesse de estender funcionalidades mais complexas já aplicadas aos documentos, como por exemplo a granularização de arquivos PDF, bom como de vídeo, ambos processos que já estão disponível e implantados no projeto da Biblioteca Digital.

Além destas mudanças vale citar o desejo pela melhoria contínua deste projeto que continua em crescimento.

REFERÊNCIAS BIBLIOGRÁFICAS

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

ASTELS, D. Test-Driven Development: A Practical Guide. *New Jersey: Prentice Hall, 2003.*

BRANDOM, R. buildout tutorial. buildout howto. buildout review. *2008. Disponível em: <http://renesd.blogspot.com.br/2008/05/buildout-tutorial-buildout-howto.html, 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.*

DESAI, M. S.; PITRE, R. Developing a curriculum for on-line internacional business degree: an integrated approach using systems and erp concepts. Education*, 2010.*

FFMPEG.ORG. Ffmpeg Documetation. *2012. Disponível em: <http://ffmpeg.org/ffmpeg-.htmlDescription, 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>.*

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

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

PIRNAU, M. C. M. 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.*

PRESSMAN, R. S. Engenharia de Software. *São Paulo: MAKRON Books, 1995.*

PYTHON Uno.

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: O'REILLY', 1999.*

ROSSUM, G. V. An introduction to Python. *United Kingdom: Network Theory Limited, 2003.*

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

SILVA, F. L. de Carvalho e; MONNERAT, G. M.; CARVALHO, R. A. de. AutonomaÇÃo na produÇÃo de software. ENEGEP*, 2010.*

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>.*

SMETS-SOLANES, J.; CARVALHO, R. de. Erp5: A next-generation, open-source erp architecture. IT Professional*, august 2003.*

SMETS-SOLANES, J.; CERIN, C.; COURTEAUD, R. Slapos: a multi-purpose distributed cloud operating system based on an erp billing model. IEEE*, 2011.*

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>.*

WIKIPéDIA. Matroska. *2012. Disponível em: <http://pt.wikipedia.org/wiki/Matroska, acesso em 15/05/2012>.*

WIKIPéDIA. OpenDocument. *2012. Disponível em: <http://pt.wikipedia.org/wiki-/OpenDocument, acesso em 15/05/2012>.*

XIPH.ORG FOUNDATION. What is Vorbis? *2012. Disponível em: <http://www.vorbis.com-/faq/what, 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-/Zope2introduction, acesso em 15/05/2012>.*