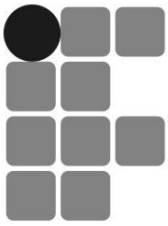


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE
Campus Campos-Centro

Secretaria de Educação Profissional e Tecnológica

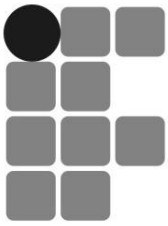
Ministério da Educação

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.

Ela foi fundamental para criação do *Cloud computing*, em português computação nas nuvens, que consiste em permitir ao usuário acesso a diversas aplicações e ao máximo de funcionalidades que esta nova forma de interação entre maquina e usuário possa disponibilizar, independente da plataforma em que o mesmo se encontre, tornando-se cada vez mais ampla a medida que o avanço tecnológico permite maior 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 para disseminação de 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. 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 esses é possível citar perda de conexão, *deadlock* no OpenOffice.org utilizado pela aplicação, estouro de memória entre outros.

Assim dada a necessidade de corrigir estes erros a fim de obter estabilidade, através da parceria do NSI com a Nexedi,foi realizada a analise desta ferramenta.

Entretanto a análise provou que era inviável corrigir tal ferramenta em função de novas demandas também percebidas.

Definiu-se assim uma nova meta, a proposta da criação de uma ferramenta que fosse capaz de manter as conversões 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 apresentada. Seguindo em parte a sigla 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 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 vem sendo desenvolvido, e sobre sua necessidade para seu funcionament.

No quarto capítulo é apresentado um breve estudo de caso do desenvolvimento desta ferramenta, avaliando seu desenvolvimento; falando sobre aplicações que o utilizam e demonstrando sobre sua atual estabilidade com base em estudos anteriores.

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

2 TECNOLOGIAS EMPREGADAS

Este capítulo apresenta um conceito simplificado sobre cada ferramenta utilizada para o desenvolvimento desta aplicação, ou que esteja diretamente relacionada a esta.

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 interpretável, tornam-na uma linguagem ideal para *scripts* e desenvolvimento rápido de aplicações em muitas áreas, em diversas plataformas (ROSSUM, 2003).

Por todas estas vantagens hoje é uma das linguagens mais utilizadas no mundo.

2.1.1 Buildout

Buildout é uma ferramenta desenvolvida em *python*, capaz de possibilitar o desenvolvimento de novas aplicações em um ambiente isolado. Ele fornece suporte para criação diversas aplicações, especialmente em *python*. A partir de um conjunto de características e variáveis de ambiente previamente definidas em uma 'receita' toda a estrutura da aplicação é gerada independente das dependências de sistema. Uma única aplicação pode conter ainda outras aplicações, processos e definições de configuração em suas próprias receitas (BRANDOM, 2008).

Levando em consideração o ideal do ambiente isolado, também chamado de puro, a instalação do *buildout* é extremamente simples, podendo ser realizada a partir de um *script python*, (PYTHON SOFTWARE FOUNDATION, 2012), que após executado gerará um *script* 'bin/buildout', a partir do qual a aplicação sera gerada com base nos arquivos de configuração(receitas), como o exemplo 2.1:

```
1 [buildout]
2 develop = .
3 parts =
4   xprompt
5   test
6
```

```
[xprompt]
recipe = zc.recipe.egg:scripts
eggs = xanalogica.tumbler
interpreter = xprompt

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

Código 2.1: Exemplo de um *script buildout*

O *script* inicia com a declaração *[buildout]* que é a única parte realmente obrigatória para identificá-lo como uma receita. Por *develop* defini-se que existe um local da aplicação que encontra-se em desenvolvimento, ideal para testes. Por *parts* subentende-se o percurso do *script*, a ordem que deve ser executado independente da ordem que estejam descritos ao longo da receita.

Em cada *part* existe um *recipe* que se trata do componente responsável pela instalação da parte a qual se encontra, *recipe* pode ser também a parte em desenvolvimento. *Eggs* se trata da opção referente as dependências necessárias. O *interpreter* é utilizado para gerar um interpretador *python* contendo as dependências relacionadas em *eggs* e que seja renomeado *xprompt*.

Assim com poucas especificações quando o *script* 'bin/buildout' for executado seguirá esses passos e gerará uma nova aplicação com essas especificações.

Ainda que incomum, vale ressaltar que aplicações não desenvolvidas em *python* podem ter seu ambiente gerado pelo *Buildout*, com base na vasta disponibilidade de *recipes*, ele possibilita a instalação até mesmo de dependências de sistemas apenas dentro do ambiente escolhido.

2.1.2 Subprocess

O *subprocess* é uma biblioteca *python* que permite a criação de processos no sistema. Similar a *Thread*, porém mais eficiente no quesito escalabilidade, pois diferentemente desta, o *subprocess* permite que seus processos utilizem diferentes processadores. Esta biblioteca foi criada na intenção de substituir outros módulos e funções obsoletas, e permitir também maior iterabilidade entre a aplicação e o sistema no qual se encontra.

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

Código 2.2: Exemplo de uso do *subprocess*

No código 2.2 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 aplicações que utilizam ambientes isolados, como citado na sessão 2.1.1, 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 por exemplo no código 2.3.

Nesses casos também é preferível o uso da variável *env* que representa as *environment* do ambiente, isto é, direciona o processo a utilizar os binários que estão neste ambiente, 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 2.3: Exemplo de uso do *subprocess*com *PIPE*

No exemplo, código 2.3 também utiliza-se da opção *PIPE* que permite ao comando retornar as mensagens de saída e erro respectivamente através das variáveis *stdout* e *stderr*, para que possam ser utilizadas de informação após seu uso.

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 responsável pela popularização do *python*, não existiu ferramenta em Perl que o levasse as pessoas como o Zope levou o *python* (UDELL, 2000), no entanto por ter se tornado muito extenso e complexo, requirindo um alto nível de taxa de aprendizagem, acabou sendo ofuscado pela ferramenta Djando em questão de popularidade.

Dado seu nível de complexidade e todas as extensões disponíveis muitos continuaram utilizando o Zope, indiferentes a sua queda de popularidade, assim muitas extensões continuam surgindo, e muitas delas se mantém 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 criadas com intuito de auxiliar na criação de outras aplicações *python*, de forma a não deixá-las muito extensas ou pesadas, entre elas 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 qualquer componentes externos que possuíssem uma *Application Programming Interface* (API). A ideia de criar uma interface para os componentes de uma aplicação é uma forma quase elegante de re-

solver 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

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. Foi derivada de um formato antigo chamado SGML, 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 este formato 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 tanto implementável em software proprietário quanto software livre, usando as licenças típicas de cada um. Em contraste com o formato proprietário que 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 diretas legais ou à especificação técnica. Um objetivo secundário dos formatos abertos é permitir a competição de mercado, 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 abertos 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 qualquer plataforma operacional.

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), em cima de uma base XML criada por desenvolvedores do OpenOffice.org, responsáveis pela aplicação que na época era uma das poucas capazes de utilizar sua estrutura, senão a única.

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.

> 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 variação de extensões, no que desrespeita a documentos são elas:

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

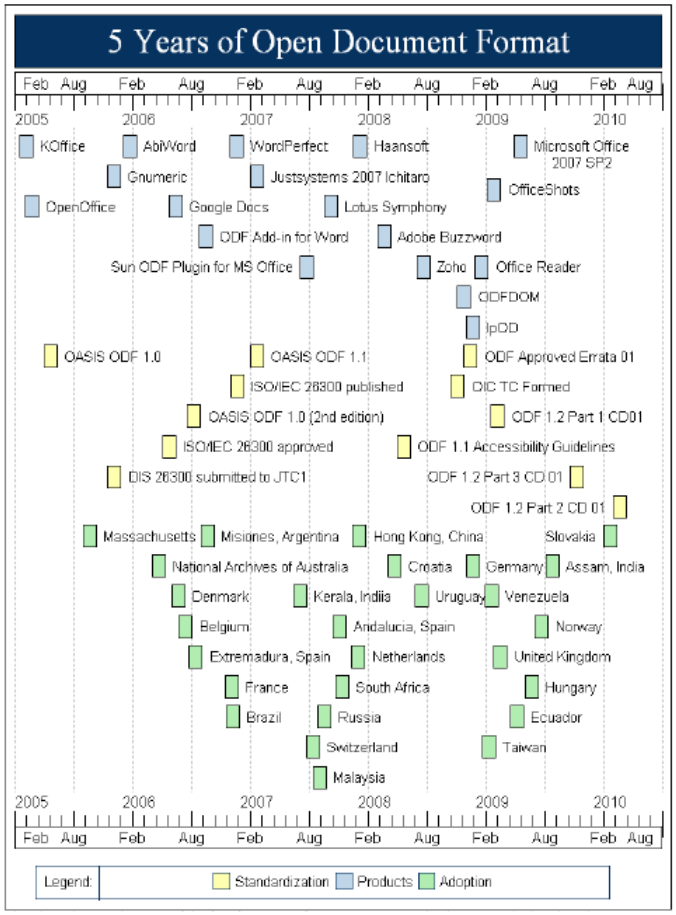


Figura 2.1: Crescimento do ODF nos primeiros cinco anos. Adaptado de (SILVA, 2010)

- Para fórmulas: .odf.

E ainda existe um padrão a parte para modelos, 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 intensão 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 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 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 contentores, ou *containers* de áudio e vídeo digitais integrados, grosseiramente chamados de formato de vídeo (WIKIPéDIA, 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 2.4 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 2.4: 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 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 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 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 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 se-

gundo 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 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, entretanto muito utilizado atualmente, principalmente por projetos livres.

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

3 CloudOoo

Em 2006, em função da necessidade da conversão de documentos a empresa francesa Nexedi SA, em parceria com Núcleo de Pesquisa em Sistemas de Informação (NSI), originou a construção da ferramenta *OpenOffice.Org Daemon* (OOOD), com base no uso da ferramenta *LibreOffice*, antiga suíte *OpenOffice.Org*, 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 *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, tais como arquivos de áudio, vídeo, imagem e PDF. A princípio essas funcionalidades seriam as mesmas aplicadas a documentos, conversões e manipulações gerais, vindo a criação de funcionalidades especificas quando a ferramente fosse considerada estável.

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 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 as interfaces:

- IApplication: representa os métodos de controles as aplicações externas do servidor;
- IFile: representa métodos para manipulação dos arquivos recebidos;
- IOdfDocument: representa métodos de manipulação específica de documentos ODF;
- 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 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 instalação em um ambiente isolado onde tanto o CloudOoo, quanto suas ferramentas 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 iniciado para possibilitar a manipulação dos documentos.

Além dos métodos citados em 3.1.1 esta classe é capaz de apresentar erros ocorridos durante processos e também de retornar o *pid* utilizado pela aplicação. E ainda um método responsável pelo 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 guardado durante sua instanciação de forma no acontecimento de erros não previstos este co

4 ESTUDO DE CASO

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

Para este estudo foram escolhidas duas formas de instalação: a primeira a partir do uso do SlapOS, subseção 4.1.3; a segunda a partir do uso direto do *buildout*, forma que é principalmente utilizada para desenvolvimento e para serviços do NSI.

Não existe praticamente diferença entre ambas as instalações, exceto pelo tempo que levam, e configuração final.

Além disso este capítulo também apresentará uma avaliação quanto a forma de desenvolvimento do CloudOoo em comparação a sua versão anterior e do uso de técnicas ágeis.

4.1 Aplicações relacionadas ao CloudOoo

Com seus quase três anos de produzido o CloudOoo já vem sendo utilizado a muito por três principais aplicações, sendo uma delas diretamente responsável por sua instalação, como citado na seção 4.4.1.1 .

A subseções a seguir apresentam sobre essas aplicações.

4.1.1 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 20 bolsistas e 20 pesquisadores.

4.1.2 ERP5

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

Muito embora 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.

4.1.3 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 processos.

Para (SMETS-SOLANES; CERIN; COURTEAUD, 2011) a computação em nuvem é dividida em três camadas, infraestrutura 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 construído 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.

4.2 Ambiente de desenvolvimento

O estudo apresentado neste trabalho foi desenvolvido 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; bem como de um notebook com processador Intel Core 2 Duo CPU 2.13 Hz, 8GB de memória 50GB de HD disponíveis para o sistema; além de dois servidores do NSI um de processador Xeon CPU E5335 2.00 x8, 14 GB de memória e 20GB de HD e um segundo com processador Xeon CPU E5335 2.00 x4, 4GB de memória e 20GB de HD, em uma máquina virtual.

As duas primeiras máquinas contavam com o sistema operacional Ubuntu 12.04 Lt, enquanto os servidores utilizavam o sistema operacional Debian 6.0.

4.3 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.

A técnica TDD(*Test-Driven Development*), ou desenvolvimento orientado a testes, defende o desenvolvimento dos testes antes do desenvolvimento da parte funcional da aplicação, dado que os testes também fazem parte da mesma. A eficácia dela garanti que todas as expectativas da aplicação sejam testadas e portanto garantidas ao usuário.

No inicio da parceria do NSI e Nexedi no desenvolvimento do CloudOoo quase não se utilizavam testes, entretanto com seu crescimento como produto e apresentada a dificuldade em mantê-lo estável foi dado o começo ao desenvolvimento de teste, a técnica TDD ainda não era empregada neste primeiro momento em função da porcentagem ja desenvolvida do produto, atualmente porém vêm sendo empregada diretamente.

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 utilização de rede por este.

Para a realização de testes unitários no CloudOoo foi preciso antes o desenvolvimento de *scripts* que pudessem interligar todas as bibliotecas envolvidas para o funcionamento do mesmo. Também foram desenvolvido *scripts* que “ligam” a aplicação e realizam testes na rede

local, a fim de garantir que as respostas estejam corretas quando este serviço estive ativo e responder a conexões distantes.

Além dos testes, outra ferramenta que garante o desenvolvimento do CloudOoo é o uso de um sistema de controle de versão, como o *subversion*, que era utilizado na versão 2.0, e que atualmente foi trocado pelo *git* em função de vantagens como, por exemplo, o controle de versões distribuído. A importância desta ferramenta esta no controle do crescimento da aplicação, que por momentos contou com uma equipe de desenvolvimento sem contato constante e em diferentes períodos.

Assim por meio de ferramentas que podiam acompanhar as modificações da aplicação, bem como reverter determinadas alterações em casos de erros posteriores na mesma e dar um detalhamento de quando essas modificações ocorreram, foi possível seguir com este desenvolvimento.

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.

4.4 Processo de instalação

Para instalação via SlapOS, foi definido como pré-requisito a instalação do próprio SlapOS, sendo este dependente apenas da existência da instalação do *python*, em qualquer versão uma vez que o SlapOS instala todos seus requisitos de funcionamento, inclusive o próprio *python*. Variando de acordo com o sistema escolhido podem ser necessários demais pacotes de dependências de sistema.

Da mesma forma, para instalação via *buildout*, forma um pouco mais simples, é necessário igualmente possuir a instalação do *python* e do *git* como pré requisitos, uma vez que através deles e do uso da biblioteca *bootstrap*, disponível em (PYTHON SOFTWARE FOUNDATION,

2012), é possível gerar o *script bin/buildout*, bem como utilizá-lo para viabilizar a instalação do CloudOoo por seus arquivos de configuração, ou receitas.

Nas próximas subseções serão apresentadas as devidas instalações.

4.4.1 Instalação via SlapOS

Esta instalação foi realizada a partir da modificação do tutorial de instalação do ERP5 no SlapOS, (NEXEDI SA, 2012), o qual passa por modificações ocasionalmente em função das atualizações frequentes desta ferramenta.

Nestas subseções será apresentado o tutorial já modificado em sequência de instalação:

4.4.1.1 Instalação do SlapOS

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

É aconselhado que seja realizada a instalação na pasta raiz do sistema, assim requerendo privilégios para instalação. É possível para o usuário optar por instalá-lo em sua pasta de usuário com algumas modificações no tutorial, entretanto em determinados momentos será podem ocorrer erros inesperados, e a exigência do uso de privilégio de qualquer forma.

Como é possível notar no código 4.1, cria-se uma pasta para instalação em */opt/slapos*, e também um arquivo de configuração *buildout*.

Após a criação deste arquivo, através do uso do *python*, é realizada a execução de um *bootstrap* próprio da Nexedi.

```
1  $ sudo mkdir /opt/slapos
2
3  $ cd /opt/slapos
4  $ touch buildout.cfg
5  $ vi buildout.cfg
6
7  [buildout]
8  extends =
9    http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.57:/
         component/slapos/buildout.cfg
10
11 $ sudo python -S -c 'import urllib2; print urllib2.urlopen("http://www.
       nexedi.org/static/\
12 packages/source/slapos.buildout/bootstrap-1.5.3-dev-SlapOS-002.py").read()'
       | python -S -
13
14 $ sudo bin/buildout -v
```

Código 4.1: Primeira parte da instalação do SlapOS

Com o termino do *bootstrap* é executado o *script bin/buildout*, que realiza a instalação dos componentes necessários ao SlapOS.

Após a instalação dos componentes e dependências é preciso configurar o mesmo, no código 4.2, existe um exemplo de arquivo de configuração e logo em seguida na figura 4.3, existe a configuração de rede para uso do SlapOS.

É necessário que o *slapproxy* seja iniciado e mantido rodando em *background* sempre que esta máquina estiver ligada e/ou em uso, ele representa funcionalidades de conectividade do SlapOS *master*.

```
$ 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 = 5
bridge_name = br1331
partition_base_name = slappart
user_base_name = slapuser
tap_base_name = slaptap
ipv4_local_network = 10.0.0.0/16

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

Código 4.2: Arquivo de configuração do SlapOS

```

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

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

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

Código 4.3: Configurações de rede do SlapOS

É possível reparar que o SlapOS é uma ferramenta adaptada ao IPV6, e necessariamente

é preciso configurá-lo através da interface *br1331*.

O *script /etc/network/interfaces* do sistema também é configurado para habilitar as funcionalidades de IPV6, a fim de que não seja preciso iniciá-las sempre que reinicie o sistema.

Por fim nas duas ultimas linhas do código 4.3 são executados os *scripts slapformat* e *slapgrid*, eles são responsáveis por registrar seu computador na nuvem e seus devidos *slots* além de habilitar o cliente SlapOS em seu computador, respectivamente.

4.4.1.2 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* citando o arquivo de configuração citado na subseção 4.4.1.1, assim com o uso de uma instância do SlapOS(*slap*), referenciada na terceira linha da figura 4.4, informa-se a esta instância qual será sua função, isto é, seu produto específico.

A partir do uso do *slapgrid* a parte referente aos componentes deste produto será instalado de forma compilada pelo *Buildout*.

```
1  $ bin/slapconsole /opt/slapos/slapos.cfg
2
3  import slapos.slap.slap
4  slap = slapos.slap.slap()
5  slap.initializeConnection('http://127.0.0.1:5000/')
6
7  slap.registerSupply.supply(
8      "http://git.rp5.org/gitweb/slapos.git/blob_plain/cloudooo:/software/
           cloudooo/software.cfg",
9      computer_guid="vifibnode")
10 #teste que retorna id da instancia
11 computer_partition.getId()
12
13 $ sudo bin/slapgrid-sr -c /opt/slapos/slapos.cfg
```

Código 4.4: Requisição de instalação do CloudOoo no SlapOS

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

Após a instalação dos componentes é necessário requerer uma instância, como em 4.5.

Este processo também requer o uso do *slapconsole*, de forma semelhante a instalação de componentes, no entanto é necessário fornecer um título a sua instância e é possível verificar se a mesma foi criada requirindo, por exemplo, seu *id*.

```
1  $ bin/slapconsole /opt/slapos/slapos.cfg
2
```

```
import slapos.slap.slap
slap = slapos.slap.slap()
slap.initializeConnection('http://127.0.0.1:5000/')

computer_partition = slap.registerOpenOrder(
    "http://git.rp5.org/gitweb/slapos.git/blob_plain/cloudooo:/software/
        cloudooo/software.cfg",
    "Meu CloudOoo")
#teste que retorna id da instancia
computer_partition.getId()

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

Código 4.5: Requisição de uma instância do CloudOoo via SlapOS

Por fim é utilizado novamente o *slapgrid*, mas desta vez com novos parâmetros para que a criação da instância seja finalizada.

Por se tratar de um serviço web em um sistema de nuvens, esta instalação compreende que após terminada deverá iniciar um servidor do CloudOoo pelas configurações estabelecidas no software.

4.4.2 Instalação do CloudOoo.git

Esta segunda instalação foi criada para ser mais flexível aos projetos do NSI que também utilizam o CloudOoo, bem como aos que tenham intenção de colaborar com esta ferramenta.

Esta instalação esta disponível no Github do NSI, (NSI, 2012a), 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 (NSI, 2012b), que se mantém sempre na última versão de desenvolvimento, a mais atual.

Após o download do (NSI, 2012a), são necessárias apenas duas instruções para esta instalação, 4.6:

```

$ python -S -c 'import urllib2; exec urllib2.urlopen("http://python-
    distribute.org/boostrap.py").read()'

$ bin/buildout -vv
```

Código 4.6: Modo de instalação do cloudOoo (NSI, 2012a)

Na primeira etapa utiliza-se o *python* para fazer *download* e rodar o *script* do *boostrap*, este irá fazer o download e disponibilizar o *Buildout*.

Ao termino desta fase, com o *script buildout* acrescido de argumentos no intuito de torná-lo verboso e do arquivo padrão de configuração (*buildout.cfg*), ocorre a instalação do

CloudOoo e todos os componentes necessários, entre eles 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* para dar inicio ao servidor.

Ainda caso o usuário desejar é possível trocar a porta de funcionamento do servidor no arquivo *cloudooo.cfg*, bem como outras configurações padrão, como o limite do uso de memória.

4.5 Processos de requisições

Para estabelecer conexão entre um cliente qualquer e o CloudOoo é necessário o uso de uma biblioteca compatível ao XMLRPC. No código 4.7 foi realizado um exemplo de cada requisição básica disponível para todos *handlers*, através de uma conexão estabelecida pela biblioteca *xmlrpclib*:

```
1  from base64 import encodestring
2  from xmlrpclib import Server
3
4  conexao = Server("http://localhost:23000")
5
6  arquivo = open("test.ogv").read()
7
8  novo-arquivo = conexao.convertFile(encodestring(arquivo), 'ogv', 'mpeg')
9
10 info = conexao.getFileMetadataItemList(encodestring(arquivo), 'ogv')
11
12 nova-info = conexao.updateFileMetadata(encodestring(arquivo), 'ogv', dict(
       Titulo="Arquivo teste"))
```

Código 4.7: Exemplo prático de uso do CloudOoo

Na linha 6 do código a variável *arquivo* recebe o conteúdo do arquivo *test.ogv*.

Para conversão deste, na linha 8, é preciso codificá-lo para que durante sua 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*, linha 10, e *updateFileMetadata*, linha 12, nos quais serão realizados respectivamente a extração e inserção de *metadados* no arquivo.

Na linha 12 é possível notar que para inserção de *metadados*, os novos dados a serem passados devem estar em um dicionário.

Observe também que nesta inserção de *metadados*, foi utilizado o *nova-info*, como resposta da requisição de inserção de *metadados*. Isto porque esta requisição do CloudOoo retorna um arquivo com os dados inseridos no mesmo, e que já se encontra codificado para transporte.

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

4.6 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 4.1, existe um exemplo de submissão de arquivos, do tipo Relatório, o qual passará pela conversão de DOC para ODT através de uma requisição ao CloudOoo, após a aprovação 4.2:

Figura 4.1: Pagina de submissão de documentos da Biblioteca Digital

Nas figuras não é implícito o uso da ferramenta, entretanto, uma vez que essa funcionalidade requer o uso de documentos em formato livre para realização de suas tarefas, é requerido o uso do CloudOoo para converter o arquivo em questão para um padrão livre e posteriormente realizar a funcionalidade de “granularização”.

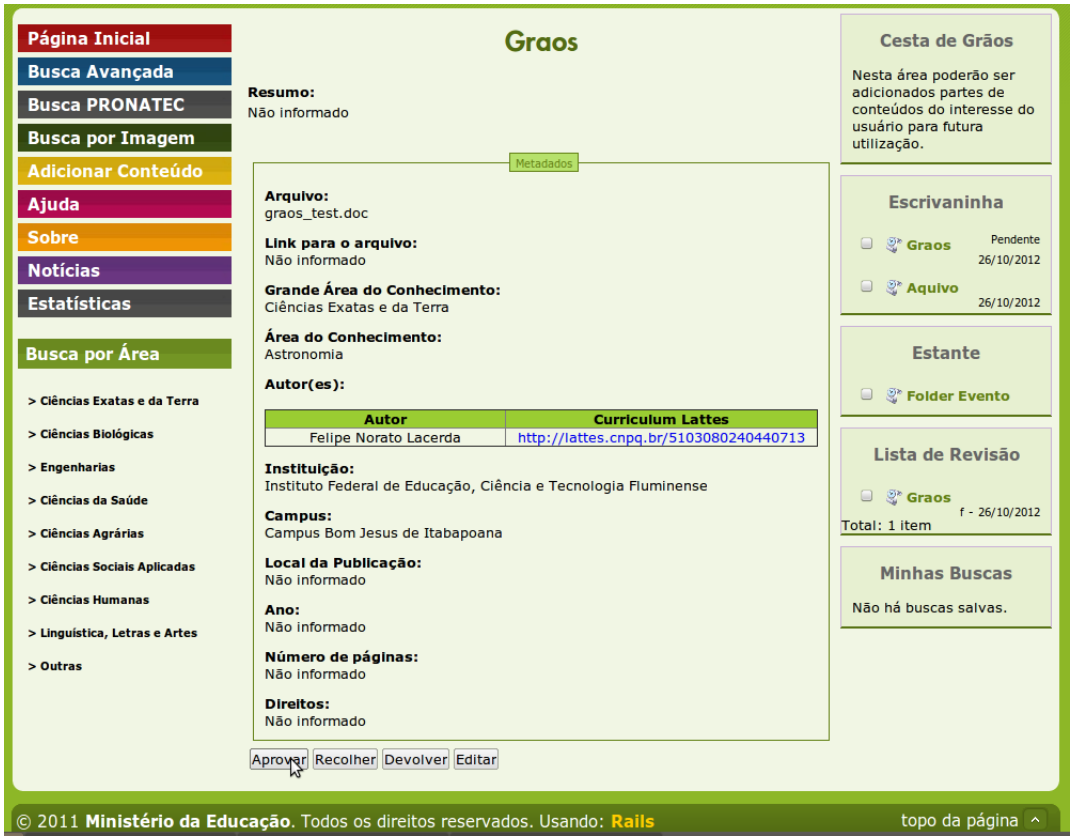


Figura 4.2: Pagina de aprovação de documentos da Biblioteca Digital

Na figura 4.3 há uma demonstração da funcionalidade de extração de “metadados”, enquanto os grãos encontram-se nas figuras 4.4:

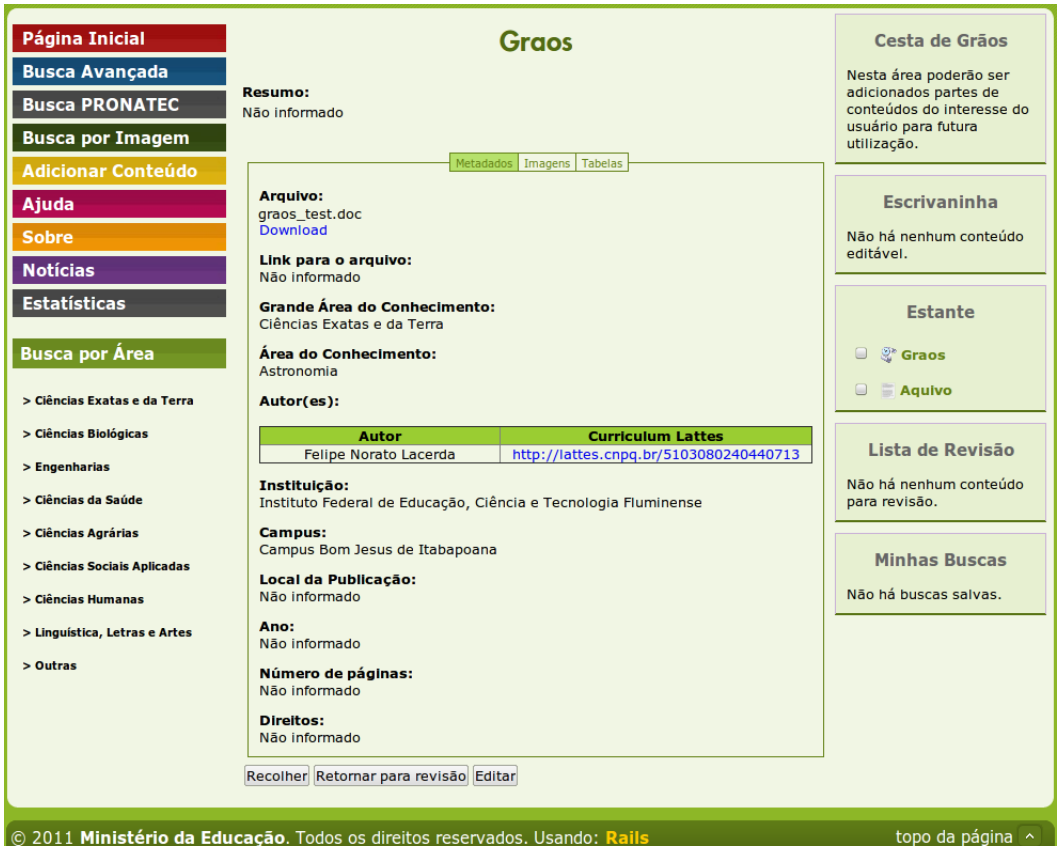


Figura 4.3: Pagina de exibição de metadados de documentos da Biblioteca Digital

Os grãos extraídos pelo processo de “granularização” são separados em imagens(4.4) e tabelas, que não existem nesse caso, e dispostos para visualização do usuário.

Figura 4.4: Pagina de exibição de grãos do tipo imagem de documentos da Biblioteca Digital

4.7 Performance

Para estabelecer a performance frente aos antecessores do CloudOoo foi realizado um teste, por (MONNERAT, 2010), 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 foi escolhidos um teste distinto representado no código 4.8 onde foram selecionados 3 diferentes documentos DOC para serem convertidos para ODT; 3 documentos com PDF para serem convertidos para TXT, 1 arquivo de vídeo AVI de aproximadamente 100 MB que seria convertido para *THEORA*; um arquivo de áudio MP3 de aproximadamente 3 MB para ser convertido para *VORBIS*; e por fim uma imagem PNG de aproximadamente 30 KB para ser convertida para JPG.

```
from random import randint
from base64 import encodestring, decodestring
from datetime import datetime
from xmlrpclib import ServerProxy
import magic


mime_decoder = magic.Magic(mime=True)

documents = ['ISSCWR6PaperTemplate.doc','SiteLeaderAppWinter2012.doc', '
    Estonia.doc']
pdfs = ['Bankruptcies.pdf', 'WIA-IM_tfw_studentguide.pdf', '2012
    FinancialRpt.pdf']

type_convert_choices = { 0: [documents, "doc", "odt", 'application/vnd.
    oasis.opendocument.text'],
                  1: ["test.png", "png", "jpg", 'image/jpeg'],
                  2: ["test.mp3", "mp3", "ogg", 'application/ogg'],
                  3: ["test.avi", "avi", "ogv", 'application/ogg'],
                  4: [pdfs, "pdf", "txt", 'text/plain']
                }


proxy = ServerProxy("http://localhost:23000/RPC2")
id = 0
process = []

while True:

  name = 'test'
  file, source_format, destination_format, mime = type_convert_choices[
      randint(0,4)]
  data = ''
  if type(file) == str:
    data = open(file).read()
  else:
    name =file[randint(0,2)]
    data =  open(name).read()
  done, mimetype = False, False
  try:
    file = proxy.convertFile(encodestring(data), source_format,
        destination_format)
    mimetype = mime_decoder.from_buffer(decodestring(file))
    if mimetype == mime:
      done = True
  except:
    done = "Error"
  process.append([id, name, source_format, mime, destination_format,
      mimetype, done, '\n'])
  id += 1

content = "  id     |          file        |     source          |   dest
              |    done          \n" + str(process)
log = open('log.txt', 'w')
log.write(content)
log.close()
```

Código 4.8: *Script* de teste

Neste teste as conversões seriam escolhidas de forma aleatória a fim de simular requisições que o CloudOoo terá de enfrentar num ambiente de produção, e a verificação de seu sucesso seria em torno do *mimetype* resultante de sua conversão.

Este teste foi realizado em duas maquinas diferentes de desenvolvimento citadas em 4.2, no notebook e no servidor de 4GB de memória.

No notebook a performasse foi consideravelmente estável em 10 horas o CloudOoo realizou 2034 conversões, entre elas 405 conversões foram de DOC para ODT e apenas 3 retornaram erro; 260 foram de PDF para TXT e nenhuma retornou erro; 307 foram de AVI para OGV *Theora* e nenhuma retornou erro; 384 foram de MP3 para OGG *Vorbis*; e 678 foram de PNG para JPG sem retorno de erro.

Resultando assim em apenas 3 erros em 10 horas.

No servidor, entretanto o resultado foi menos promissor, o teste durou apenas 3 horas pois ocorreu um erro de estouro de memória no servidor, que despunha de aproximadamente 3 GB livres para o teste.

Durante essas 3 horas foram realizados 611 conversões, entre elas 122 conversões foram de DOC para ODT e apenas 1 retornou erro; 77 foram de PDF para TXT e nenhuma retornou erro; 91 foram de AVI para OGV *Theora* e nenhuma retornou erro; 114 foram de MP3 para OGG *Vorbis*; e 207 foram de PNG para JPG sem retorno de erro.

De forma positiva o CloudOoo se manteve estável durante essas horas tanto no notebook quanto no servidor, pois mesmo com o estoura de memória no segundo, ele se manteve ativo. Entretanto com os resultados dos teste foi constatado que com as novas funcionalidades adicionadas será preciso rever seus testes de escalabilidade e uso a fim de que não volte a ocorrer estouros de memória independente da máquina em uso.

No que respeito aos erros de documentos, foram erros relativos aos documentos em conversão os quais foram registrado em seu *log* e serão verificados a fim de garantir que também não voltem a ocorrer.

5 CONCLUSÕES

5.1 Objetivos alcançados

Neste trabalho foi possível apresentar de forma simplificada porém descritiva os processos e tecnologias empregadas por trás do CloudOoo, uma aplicação livre e de código aberto, que se encontra disponível a 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 visando sua melhoria continua.

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, que se demonstraram poucos estáveis.

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 de vídeo, que é um processo já disponível e implantado no projeto da Biblioteca Digital.

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-.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.

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

MONNERAT, G. *Trabalho de Conclusão de Curso*. 2010. Disponível em: <https://github-.com/gmonnerat/monografia/blob/master/Monografia.pdf>. Acesso em: 23/09/2012.

NEXEDI SA. *User install ERP5 with SlapOS*. 2012. Disponível em: <http://www-.erp5.com/user-Install.ERP5.With.SlapOS/user-Install.ERP5.With.SlapOS>. Acesso em: 20/08/2012.

NSI. *Buildout for Cloudooo*. 2012. Disponível em: <http://github.com/nsi-iff-/cloudooo_buildout>. Acesso em: 20/08/2012.

NSI. *Cloudooo*. 2012. Disponível em: <http://github.com/nsi-iff/cloudooo>. Acesso em: 20/08/2012.

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.

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

PYTHON SOFTWARE FOUNDATION. *buildout.bootstrap*. 2012. Disponível em: <http://pypi.python.org/pypi/buildout.bootstrap/1.4.1>. Acesso em: 23/09/2012.

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.

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.

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.