UNIVERSITÀ DEGLI STUDI DI FIRENZE

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

CURSO DE CIÊNCIA DA COMPUTAÇÃO

Redes federadas eventualmente conectadas

Arquitetura e protótipo para a Rede Mocambos

de

Vincenzo Tozzi

Orientador:
Prof. Rocco De Nicola

Co-orientador:
Dott. Alberto Mancini

Ano Académico 2010-2011

"Vamos fazer um mundo mais do nosso jeito..."

Zumbi dos Palmares

UNIVERSITÀ DEGLI STUDI DI FIRENZE

Abstract

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in Informatica

Eventually connected federated network

Architecture and prototype for Mocambos Network

by Vincenzo Tozzi

This work researches on technologies for a federated network of afrodescendants communities looking at autonomy and freedom. The work propose an architecture and prototype for an eventually connected federated network where the main limit and technical bond is the connection availability and bandwidth. The prototype is a media sharing system for local web portals, based on Django and Git-Annex, authenticated on LDAP servers.

Agradecimentos

Um agradecimento especial para as minhas famílias...

Sumário

Lista de Figuras

Siglas

GESAC	**G**overno **E**letrônico **S**erviço de **A**tendimento ao **C**idadão
RM	**R**ede **M**ocambos
SP	**S**ervice **P**rovider
IdP	**Id**entity **P**rovider
API	**A**pplication **P**rogramming **I**nterface
RFC	**R**equest **F**or **C**omments
TOS	**T**erms **O**f **S**ervice
P2P	**P**eer To **P**eer
LDAP	**L**ightweight **D**irectory **A**ccess **P**rotocol
DIT	**D**irectory **I**nformation **T**ree
XMPP	e**X**tensible **M**essaging and **P**resence **P**rotocol
SSO	**S**ingle **S**ign **O**n
VSAT	**V**ery **S**mall **A**perture **T**erminal
DRY	**D**on't **R**epeat **Y**ourself
MVC	**M**odel **V**iew **C**ontroller
ORM	**O**bject **R**elational **M**apper
NPDD	**N**úcleo de **P**esquisa e **D**esenvolvimento **D**igital
NFC	**N**úcleo de **F**ormação **C**ontinuada

Para Giusy, minha mãe...

Capítulo 1

Introdução

Este trabalho é baseado numa experiencia direta que teve origem em 2004 em Florença para depois se desenvolver e continuar no Brasil desde 2005 ate hoje. Em maio de 2004, com o Coletivo de Informática "Ada Byron"[1], e com a associação Hipatia[2], organizamos uma serie de encontros e conferencias sobre o tema da liberdade do conhecimento aplicada a vários ámbitos. Parteciparam a "Modelli liberi"[3], assim se chamavam os encontros, relatores de relevança internacional, como Richard Stallman[4], Diego Saraiva[5], Juan Carlos Gentile[6] e, representando o governo brasileiro, Sergio Amadeu[7] e Elaine da Silva[8]. Em seguida, retomei os contatos com Elaine, e em 2005 me mudei para Brasília onde comecei a trabalhar no programa de inclusão digital federal GESAC, *Governo Eletrônico Serviço de Atendimento ao Cidadão.* O GESAC, era um dos mais ambiciosos projetos inclusão digital da América Latina. O programa previa um numero inicial de 3200[9] conexões por satélite, uma plataforma de serviços online e uma equipe no território. O Ministério das Comunicações tinha entregue a direção do programa para Antonio Albuquerque, um dos fundadores do Sindicado dos Trabalhadores nas Telecomunicações brasileiro, que apoiou e espalhou o uso do Software Livre em todos os níveis do programa,

[1] O Coletivo de Informática "Ada Byron" é um grupo de estudantes do Curso de Informática da Universidade de Florença. O sitio internet do coletivo é `http://informada.wordpress.com`

[2] Hipatia nasceu como coordenamento espontaneo de pessoas do mundo inteiro que compartilham a mesma visão e o mesmo objetivo: uma sociedade do conhecimento global baseada na liberdade, igualdade e soliedaridade", traduzido desde `http://www.hipatia.net`.

[3] "Modelos livres". Um artigo sobre o evento é disponível, em italiano, em `http://www.apogeonline.com/webzine/2004/04/16/01/200404160103`

[4] Richard Stallman é presidente e fundador da *Free Software Foundation*

[5] Professor da Universidade Nacional de Salta em Argentina e fundador do projeto Ututo, `http://ututo.org`

[6] Juan Carlos Gentile é coordenador e um dos fundadores da ONG Hipatia, `http://www.hipatia.net`.

[7] Sergio Amadeu é professor da Universidade Federal do ABC (UFABC), e na epoca era presidente do Instituto Nacional de Tecnologia da Informação do Brasil.

[8] Elaine da Silva Tozzi é diretora de cultura do Municipio de Hortolandia/SP, na epoca era na coordenação do GESAC.

[9] Hoje as conexões são mais de 11000.

remarcando a sua importância estratégica. O nosso trabalho consistia na pesquisa e experimentação de soluções informáticas aplicadas aos mais variados habitat e contextos sociais e culturais. Eramos colocados continuamente em contato com realidades extremamente diferentes. O GESAC atende escolas, associações, quarteis, comunidades rurais, aldeias indígenas, quilombos[10], caiçaras, etc. Na maioria dos casos eram pessoas na primeira experiência com informática. A abordagem escolhida favorecia o dialogo e a troca, como por exemplo, no lugar de aulas, se recorria a rodas de conversa. Na equipe do GESAC foram convidados a trabalhar pessoas com experiencia na área social, como por exemplo ativistas de Indymedia[11] e Intervozes[12]. O dialogo aberto e paritário, a partir das especificidades do contexto, abriam novos espaços e olhares para a revolução digital, recolocando ao centro da discussão o fim, além do meio. Além de não chegar com soluções pré-definidas, frequentemente as ferramentas e as praticas precisavam ser reinterpretadas e possivelmente adaptadas as novas necessidades.

No bojo desta experiência seguiu a colaboração com algumas das comunidades que eram atendidas pelo GESAC, em especial com a Casa de Cultura Tainã[13] que naqueles anos colocava as bases do primeiro núcleo da Rede Mocambos (RM) (ver 2.2).

1.1 Neutralidade tecnológica

Essa monografia pesquisa uma solução tecnológica para facilitar a comunicação entre comunidades quilombolas. Essas comunidades afrodescendentes compartilham muitos aspetos sociais, culturais, históricos e geográficos que as diferenciam das realidades para quais, e pelas quais, são normalmente desenvolvidos os meios de comunicação digital. A neutralidade do meio é muitas vezes tida como certa, assim fica difícil perceber quanto este na realidade influi e condiciona nossas ações e nossos objetivos. Somos levados a pensar o meio como neutro, mas se considerarmos, como exemplo, os principais e

[10] Os quilombos são comunidades fundadas por africanos deportados e afrodescendentes que resistiram a escravidão perpetrada por colonizadores portugueses e europeus em toda América Latina. Muitas das comunidades chegaram ate os dias de hoje e, só no Brasil, se contam cerca de cinco mil. A Constituição brasileira estabelece o direito a terra para os quilombolas, também no sentido de reparação histórica.

[11] "O CMI Brasil é uma rede de produtores e produtoras independentes de mídia que busca oferecer ao público informação alternativa e crítica de qualidade que contribua para a construção de uma sociedade livre, igualitária e que respeite o meio ambiente.", retirado de `http://www.midiaindependente.org/pt/blue/static/about.shtml`.

[12] A missão do Intervozes é: "promover o direito humano à comunicação, trabalhando para que este seja apropriado e exercido pelo conjunto da sociedade na luta por uma sociedade democrática, justa e libertária, construída por meio da autonomia, dignidade e participação de todos e todas., retirado de `http://www.intervozes.org.br/o-intervozes`.

[13] A Casa de Cultura Tainã é uma entidade cultural e social sem fins lucrativos fundada por moradores da periferia de Campinas em 1989. O objetivo da entidade é possibilitar o acesso à informação, fortalecendo a prática da cidadania e a formação da identidade cultural, visando contribuir para a formação de indivíduos conscientes e atuantes na comunidade. O endereço do sitio internet é: `http://www.taina.org.br`.

mais comuns meios de comunicação, as línguas, podemos perceber como essas não são intercambiáveis sendo a expressão das culturas e das sociedades que as usam e as vivem. Na era digital é importante considerar o peso e a influencia das novas linguagens e meios de comunicação.

1.2 Internet, autonomia e liberdade tecnológica

Nos últimos anos a difusão da banda larga, mas sobretudo estratégias como aquela adotada pela Google, transformaram o próprio conceito da internet que de rede global de redes heterogêneas, vira principalmente uma rede para a globalização de serviços fortemente centralizados e uniformados. Até poucos anos atras era normal para uma organização prover a gestão, manutenção e as vezes ao desenvolvimento de serviços além da infraestrutura tecnológica para os próprios usuários, desde as bases de dados até aos serviços de alto nível, como sistemas para mensagens e armazenamento de arquivos. Nesses contextos nasceram e se desenvolveram boa parte das tecnologias de comunicação hoje disponíveis mas que vem sendo em parte abandonadas por escolhas principalmente de mercado. Internet nasceu da troca aberta entre os responsáveis de varias redes, unidos pela vontade de conectar as suas diferentes realidades. A criação de novos serviços, para essas redes heterogêneas, era baseada na discussão e na confrontação. O primeiro uso documentado do termo "internet", seguindo essa pratica, apareceu de fato num RFC [1]. Os "Request for Comments (RFC)"[14] são documentos, normalmente escritos em uma linguagem simples e informal, orientados para difusões e discussões de novas tecnologias no ambiente das telecomunicações. São também a base principal para a definição de novos padrões e protocolos. Atualmente os RFC são o canal oficial de publicação da *The Internet Engineering Task Force (IETF)*, da *The Internet Architecture Board (IAB)*, entre outros, e, em geral, da comunidade mundial de pesquisadores da área das redes de comunicação.

Internet então, rede de redes heterogêneas, nascida da pratica da discussão, da confrontação, do "pedido de comentários", esta ultimamente vivendo a era dos *Terms of Service (ToS)*[15] sobre infraestruturas sempre mais homologadas e centralizadas, as ditas "nuvens", ou, mais geralmente e comercialmente, *cloud*[16]. Aos protocolos padrão se

[14] Pedido de comentários.

[15] Termo de Uso

[16] "Em informática, com o termo inglês, cloud computing se indica um conjunto de tecnologias que permitem, normalmente sob forma de serviço oferecido por um provedor ao cliente, de memorizar/armazenar e/ou elaborar dados (através de CPU ou software) graças ao uso de recursos hardware/software disponíveis e virtualizados na Rede. O uso correto do termo é contestado por vários especialistas: se essas tecnologias são vistas por alguns como uma maior evolução tecnológica oferecida pela rede Internet, por outros, como Richard Stallman, são considerados uma armadilha de marketing.", traduzido de Wikipedia: `http://it.wikipedia.org/wiki/Cloud_computing`.

substituem API proprietárias e suscetíveis de alterações continuas e unilaterais. Uma empresa oferece serviços através API, decidindo os termos pelos quais os usuários, ou outras empresas, podem acessar e usar estes serviços.

Do site do Google:

> "*Google has been pushing the technological bounds of cloud computing for more than ten years. Today, feedback and usage statistics from hundreds of millions of users in the real world help us bring stress-tested innovation to business customers at an unprecedented pace. From our consumer user base, we quickly learn which new features would be useful in the business context, refine those features, and make them available to Google Apps customers with minimal delay.*"[17]

Como declarado ainda, "o Cloud computing esta no DNA do Google" que têm mais de 350 milhões de usuários ativos na sua nuvem[18].

Segundo a sociedade de consultoria Gartner, em 2016, todas as empresas contempladas pelo *Forbes Global 2000* usaram soluções *cloud* [2].

Em essência um tempo o desenvolvimento seguia um modelo *bottom up*[19], no qual os mestres informáticos desenvolviam sistemas ad hoc para as necessidades locais, para depois abrir uma discussão em rede com seus homólogos, para definir protocolos padrão e colocar em comunicação o todo.

Hoje passamos a um modelo de desenvolvimento *top down*[20], no qual novos serviços são lançados se baseando em pesquisas de mercado e testes sobre amostras de usuários. Pocas empresas dominam em traçar o desenvolvimento da rede que assume horizontes sempre mais determinados pelo mercado. Além disso o mercado de referência, e amostra escolhida, são principalmente ligados a contexto econômico, social e cultural norte-americano e europeu.

Mesmo sendo uma analise de interesse filosófico/antropológico vira território de fronteira com a informática estudar a cultura digital e os efeitos que as tecnologias digitais podem

[17] "Google tem empurrado os limites tecnológicos do cloud computing desde mais de dez anos. Hoje, os feedback e as estatísticas de uso de centenas de milhões de usuários reais no mundo nos ajudam a trazer inovações bem experimentadas para os nossos clientes comerciais com um ritmo sem precedentes. Desde a base de usuários, aprendemos rapidamente quais características podem ser uteis para o mercado, melhorando-as e deixando-as disponíveis para os usuários de Google Apps com um atraso minimo.", Retirado desde `http://www.google.com/apps/intl/en/business/cloud.html`.

[18] "Declaração dada no "earning call" do 19 janeiro de 2012, ver: `http://thenextweb.com/google/2012/01/19/gmail-closes-in-on-hotmail-with-350-mm-active-users/`.

[19] De baixo para cima.

[20] De cima pra baixo.

ter sobre as culturas de diversos países e povos. Uma analise estruturada e analítica foi deixada pelo Vilém Flusser, recentemente redescoberto e considerado entre os primeiros filósofos da comunicação dos nossos tempos, de quem cito dois pensamentos muito atuais, mesmo publicados em 1983 no livro *Für eine Philosophie der Fotografie*:

> "*Both those taking snaps and documentary photographers, however, have not understood 'information.' What they produce are camera memories, not information, and the better they do it, the more they prove the victory of the camera over the human being.*"[21], Flusser [3].

e

> "*Our thoughts, feelings, desires and actions are being robotized; 'life' is coming to mean feeding apparatuses and being fed by them. In short: everything is becoming absurd. So where is there room for human freedom?*"[22], Flusser [3].

Uma abordagem diametralmente oposta a centralização da rede são as tecnologias *Peer To Peer (P2P)*[23]. Mesmo sendo as tecnologias P2P uma alternativa disponível e já implementada a vários níveis, apresentam algumas caraterísticas pelas quais não podem ser amplamente aplicadas ao contexto da Rede Mocambos. As conexões das comunidades da RM são quase todas via satélite, com banda muito limitada. Os protocolos P2P, pesando na banda de muitos nós, para operações efetuadas por um único nó, penalizariam exatamente o recurso mais escasso.

[21] "Seja quem se limita a tirar uma fotografia, seja os fotógrafos documentaristas, não compreenderam a 'informação'. Aquilo que produzem são memorias fotográficas, não informação, e melhor o fazem, mais confirmam a vitoria da maquina fotográfica sobre o ser humano."

[22] "Nossos pensamentos, sentimentos, desejos e ações vem vendo robotizados; a 'vida' começa a significar nutrir aparelhos e ser nutridos por eles. Em breve: tudo esta tornando-se absurdo. Assim onde esta o espaço para a liberdade humana?"

[23] O principio base do P2P prevê uma infraestrutura de comunicação onde os nós comunicam diretamente entre eles, sem nós centrais pré-configurados. Existem também protocolos P2P que prevêem super-nós, de alguma forma replicando o modelo cliente/servidor, normalmente usados para otimizar as prestações e superar problemas de atravessamento de redes mascaradas. Geralmente os sistemas P2P não precisam de configuração e necessitam um conhecimento mínimo da arquitetura subjacente, dado que se baseiam em protocolos para roteamento automático dos pacotes. Essas tecnologias funcionam bem quando um numero suficiente de nós é ativo e participam no funcionamento da rede. Um exemplo de protocolo P2P muito usado é *bittorrent*, através do qual é possível transferir dados em alta velocidade quando os conteúdos requeridos são presentes em nós com banda a disposição.

Capítulo 2

Redes federadas eventualmente conectadas

Com rede federada pode se entender um universo muito amplo de situações. Neste trabalho se considera principalmente uma rede federada que tenha as seguintes caraterísticas:

- administração descentralizada
- acesso a gestão lógica (e física) da rede
- conhecimentos técnicos locais das tecnologias usadas na rede
- serviços internos a rede federada
- interações entre redes locais inteligentes

Uma rede federada, então, entendida como um conjunto de soluções tecnicamente viáveis e adaptáveis a usos, práticas e contextos diferentes, que permita uma gestão flexível da rede, com sub-redes heterogêneas, aproveitando diversas tecnologias, como por exemplo P2P onde necessário. Uma rede federada se adapta particularmente num contexto onde já existe uma estrutura organizacional que pode se espelhar na estrutura da rede e que pode acompanhar sua gestão.

2.1 Redes federadas eventualmente conectadas

A introdução sobre o contexto sugere o âmbito de estudo mas necessita uma restrição maior. Para "redes federadas eventualmente conectadas" consideramos, uma rede federada baseada em conexões não sempre disponíveis, como as conexões por satélite,

com a exigência de manter os serviços federados ativos, mesmo em ausência de comunicação. Os serviços federados são, ainda, otimizados para a resiliência do sistema e a redução do trafego de rede externo, através de estrategias de replicação, sincronização e memorização dos dados na infraestrutura logico/física local.

2.2 A Rede Mocambos

> "*É uma rede solidária de comunidades, no qual o objetivo principal é compartilhar ideias e oferecer apoio recíproco.*" ..."*A tecnologia é uma frente de trabalho da Rede Mocambos, sendo ao mesmo tempo ideia e meio para transferir ideias.*" [4].

A Rede Mocambos (RM), atualmente envolve diretamente mais de duzentas comunidades quilombolas, coletivos, aldeias indígenas, pontos de cultura e terreiros (ver figura 2.1). Existem dois termos de cooperação[1] entre Rede Mocambos e o Ministério das Comunicações, precisamente com os programas GESAC e Telecentros.BR.

O GESAC garante conectividade por satélite a todas as comunidades, com banda limitada devido a tipologia de tecnologia, que devido as distancias representa a única alternativa, pelo menos no curto e médio prazo. Através do programa Telecentros.BR estão sendo instalados telecentros, salas equipadas com computadores para acesso público, e garantidas bolsas para monitores, para a gestão desses espaços. É exatamente em contextos tão específicos que nasce a necessidade de adaptar a tecnologia as exigências locais, também pelas limitações técnicas impostas. A escassez de banda leva a reconsiderar a rede não somente como o meio de conexão para os grandes *data centers*. A rede pode, e neste caso deve, ser estruturada no território com logicas de desenvolvimento e gestão local determinadas pelas comunidades. Neste sentido é fundamental a formação e o acesso as tecnologias. A Casa de Cultura Tainã, núcleo fundador da Rede Mocambos, e entre as primeiras realidades populares que perceberam a necessidade do Software Livre, como expressão de liberdade de poder criar as próprias ferramentas tecnológicas digitais. A exclusão digital não diz respeito somente ao acesso ao mundo digital mas também da impossibilidade de participar a sua criação e desenvolvimento.

Para a RM é importante, por vários aspectos, poder construir e gerir os meios de comunicação e adaptá-los a seu contexto. No Brasil são muitas as comunidades indígenas, quilombolas e tradicionais que conhecem bem as potencialidades das tecnologias digitais e estão, cada vez mais, dominando-as. Isso não teria sido possível sem a existência e

[1] Os termos de cooperação foram assinados, como Rede Mocambos, mas formalmente pela Casa de Cultura Tainã.

FIGURA 2.1: Mapa das comunidades da Rede Mocambos, retirado desde `http://mapa.mocambos.net`

ampla difusão de Software Livre. Tecnologias digitais sob forma de produtos comerciais seriam o enésimo passo para uma dependência econômica e cultural. Esta consciência esta atras de escolhas bem ponderadas. Alguns anos atras, em Brasília, numa reunião do programa Luz Para Todos, programa federal para levar a eletricidade a população da área rural, um cacique disse que, mesmo querendo a energia elétrica, esta deveria ser limitada aos espaços comunitários e não residenciais. Não é difícil entender o porque de tal condição. Além dos aspectos culturais, ter um contador e uma conta para cada casa significaria introduzir custos em dinheiro, que não são compatíveis com a economia deles.

FIGURA 2.2: Fotos da Rede Mocambos

2.3 Tecnologias

Antes de mergulhar nos requisitos específicos, e nas escolhas feitas, pode ser útil uma breve descrição das ferramentas tecnológicas que de várias formas foram usadas para estruturar o protótipo, para a Rede Mocambos, de rede federada eventualmente conectada, em particular para os seus serviços de base, como a identificação, autenticação e comunicação.

2.3.1 LDAP

Lightweight Directory Access Protocol (LDAP)[2] é um conjunto de protocolos abertos para acessar as informações mantidas centralmente através de uma rede. LDAP organiza as informações através de uma hierarquia a árvore chamada *Directory Information Tree (DIT)*[3]. LDAP é um sistema cliente/servidor. O servidor pode usar uma variedade de banco de dados para armazenar um DIT, normalmente otimizados para operações de leitura. Quando uma aplicação cliente se conecta a um servidor LDAP, pode consultar o diretório ou tentar alterá-lo. No caso de uma consulta, o servidor pode responder de maneira local, ou pode encaminhar o pedido para um servidor LDAP em condição de responder. Se a aplicação cliente está tentando modificar informações do diretório LDAP, o servidor verifica se o usuário possui permissão de efetuar a mudança, para em seguida adicionar e atualizar as informações. LDAP suporta a delegação de parte do DIT para servidores específicos, a replicação só em leitura e a replicação em leitura/escritura (chamada *multi-master*). LDAP é um protocolo solido, muito comum e suportado, e desde muito tempo é o padrão de fato para gerir base de dados de usuários. OpenLDAP é uma implementação aberta e livre do protocolo LDAP, que inclui cliente, servidor e uma série de ferramentas para facilitar a administração.

2.3.2 XMPP

Extensible Messaging and Presence Protocol (XMPP)[4] é um conjunto de protocolos abertos para as mensagens e a presença em rede baseado no XML. XMPP é um sistema cliente/servidor. As especificações para a comunicação entre servidores permitem que os usuários de um servidor interajam de maneira transparente com usuários de outros servidores federados. A *XMPP Standard Foundation (XSF)* coordena o desenvolvimento das extensões do padrão por meio das *XMPP Extensions Protocols (XEPs)*, que até hoje são já 311. XMPP e as XEPs constituem um ambiente flexível e completo para o

[2] Protocolo Leve de Acesso a Diretório.
[3] Arvore do diretório de informações
[4] Protocolo extensível para mensagens e presencia.

desenvolvimento de serviços federados. Estes protocolos são já usáveis graças as muitas implementações de servidores, clientes e bibliotecas livres. A história do XMPP, uma vez conhecido como Jabber, também é interessante. Jabber foi inicialmente desenvolvido por Jeremie Miller na sua fazenda no Iowa. É um exemplo concreto de como a pesquisa e o desenvolvimento de tecnologias da comunicação, fora de ambientes acadêmicos e empresariais, além de possível pode ser revolucionária. De fato, hoje, XMPP é a tecnologia mais usada para mensagens instantâneas também pelo grandes atores da *new economy*.

Para as necessidade específicas de uma rede é possível então estender e customizar as funcionalidades do próprio servidor XMPP a ao mesmo tempo usufruir dos serviços base, mensagens e presencia, implementados pelos servidores já existentes na rede.

2.3.3 OpenID

OpenID é um sistema de identificação descentralizada no qual a própria identidade é uma URL que pode ser verificada por qualquer serviço que suporte o protocolo. É um protocolo aberto e são disponíveis várias implementações livres. Além disso o protocolo foi adotada pelos principais provedores de serviços web. Com OpenID é possível usar a mesma identidade em mais serviços e é uma ótima base para um sistema *Single Sign On (SSO)*. O protocolo utiliza HTTP e *Cookies* para manter uma sessão ativa. No primeiro tentativo de autenticação em um serviço compatível com OpenID, o usuário é redirecionado para o próprio provedor OpenID para efetuar o acesso e confirmar a autorização a proceder com o serviço inicial. Por todo o período da sessão, é possível acessar serviços OpenID sem reinserir as credenciais.

2.3.4 OAuth

OAuth é um protocolo aberto para autorização de serviços através API. Por exemplo, permite um usuário autorizar o acesso a informações específicas armazenadas num site, chamado *service provider*, para outro site, chamado *consumer*, sem necessidade de compartilhar a sua identidade. É uma maneira de publicar e interagir com dados protegidos. Existem muitos outros protocolos e API parecidos como Google AuthSub, AOL OpenAuth, Yahoo BBAuth, Upcoming API, Flickr API, Amazon Web Services e cada um fornece maneiras proprietárias para troca de credenciais e para acesso através de *tokens*. OAuth é uma padronização aberta das práticas mais comuns. Além disso foi pensado para suportar vários tipos de aplicações, não somente para serviços web.

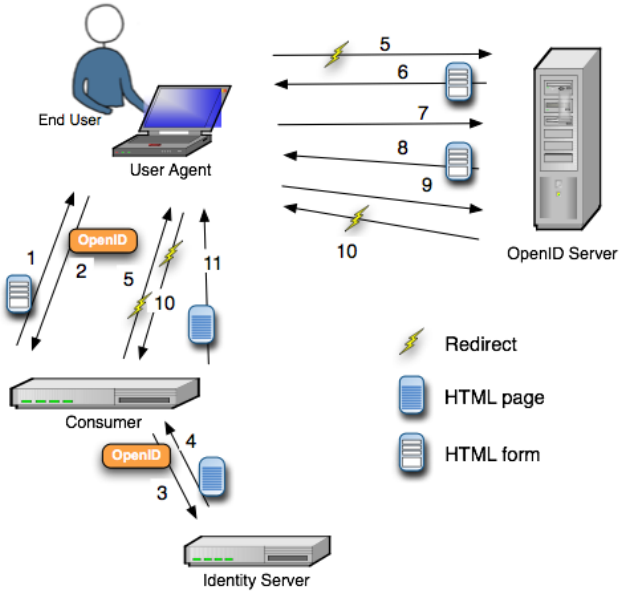

FIGURA 2.3: Diagrama de autenticação de OpenID

2.3.5 Shibboleth

Shibboleth é uma arquitetura e uma implementação aberta para autenticação e autorização de identidades federadas baseada no *Security Assertion Markup Language (SAML)*[5]. As identidades federadas permitem que as informações de um usuário sob um certo domínio possam ser compartilhadas com um outro domínio federado. Isso permite o SSO através de mais domínios sem a troca de nomes de usuários e senhas. Os IdP mantêm as informações do usuário enquanto os SP fazem uso dessas informações para o acesso seguro aos conteúdos.

2.3.6 Kerberos

Kerberos é um protocolo aberto para autenticação forte em redes de computadores. É um protocolo cliente/servidor e permite a autenticação reciproca, ou seja ambos verificam suas identidades. Kerberos é baseado no protocolo de Needham-Schroeder a chaves simétricas e prevê uma entidade terceira de confiança chamada *Key Distribution Center (KDC)*.

[5]Linguagem de marcação para asserções de segurança.

2.3.7 Django

Django é um framework, escrito em Python, para o desenvolvimento rápido de aplicações web, também conhecido como "*The web framework for perfectionists with deadlines*"[6], pois foi criado com o objetivo de respeitar o máximo possível os princípios DRY[7], e então oferece todas as ferramentas para escrever código limpo de maneira pragmática e eficaz.

As caraterísticas do Django são:

- é um framework *Model, View, Template*, (MVT), correspondente ao difundido *pattern Model, View, Controller*, (MVC). O *Template* do Django corresponde ao *view* de qualquer framework MVC, enquanto a *View* corresponde ao *controller* (mesmo se as funcionalidades do *controller*, no caso do Django, não são limitadas a componente *View*).

- permite modelar os dados diretamente em python e o *Object Relational Mapper* (ORM), se ocupa de transformá-los em código SQL. Também não precisa escrever *query* diretamente em SQL; é só usar o ORM para obter os resultados diretamente sob forma de objetos python. O ORM de Django suporta PostgreSQL, MySQL, sqlite, Microsoft SQL Server e Oracle.

- possui uma biblioteca para *form* realmente poderosa e expressiva que permite realizar logicas complexas com poucas linhas de código, deixando o máximo controle ao desenvolvedor para todas as situações.

- é um framework "*batteries included*", ou seja já vêm com algumas aplicações, chamadas *apps*, para as funcionalidade mais comuns, que reduzem o tempo de desenvolvimento, por exemplo um sistema de autenticação, uma interface de administração, um framework para gerar *sitemaps XML* e muito mais. Além disso a comunidade é bem ativa e existem muitas *apps* adaptáveis e reusáveis.

- é entre os melhores framework quanto a documentação oficial. Além de alguns tutoriais, uteis para aprender as bases, todas as funcionalidades e características do framework são bem documentadas.

[6] "O framework web para perfeccionistas com prazos".

[7] "Don't Repeat Yourself, não se repita, (DRY, também conhecido como "Single Point of Truth", ponto único de verdade) é um principio segundo o qual a informação não tem que ser repetida e redundante e não tem que expressar o mesmo conceito mais de uma vez.", traduzido do Wikipedia: `http://it.wikipedia.org/wiki/Don%27t_Repeat_Yourself`.

2.3.8 Git

Git é um sistema multiplataforma para o controle de versão distribuído, projetado para ser rápido e usável mesmo em grande projetos.

As características principais incluem:

- é totalmente distribuído e cada clone de um repositório contem o histórico inteiro das versões e no qual podem ser efetuadas operações independentemente de conexões de rede o de servidores centrais. As mudanças podem ser copiadas entre um clone e o outro e são mantidos em *branch* (ramos) diferentes, facilitando as operações de *merge* (fusão). Os repositórios são facilmente acessíveis através do eficiente protocolo do Git, que além de suportar HTTP, pode funcionar junto com SSH, para obter conexões seguras e um sistema de autenticação solido e bem comum.
- suporta o *branching* (ramificação), e o *merging* (fusão), de maneira rápida e conveniente, incluindo uma serie de ferramentas para visualizar e navegar o histórico não linear das versões.
- é muito rápido e escala mesmo em projetos muito grandes e com muitas mudanças, graças a um eficiente sistema de empacotamento e memorização do histórico (é considerado o mais eficiente entre os sistemas atualmente disponíveis).
- associa um nome de versão, para cada *commit*, que é função do histórico inteiro, por isso, uma vez publicada uma versão, não é possível alterar as velhas sem ser notado. As versões podem também ser etiquetadas e assinadas digitalmente com GPG.

Git é um sistema completo que, em bom estilo Unix, é organizado em programas e comandos independentes, pensados para ser facilmente usáveis, seja automaticamente através de *scripting* seja de maneira interativa pelo usuário final. Git é, então, uma base solida para o desenvolvimento de aplicações orientadas a sincronização, a portabilidade e a gestão autônoma e descentralizada.

"A força da rede está nos nós"

TC

Capítulo 3

Uma arquitetura para a Rede Mocambos

3.1 Especificação dos requisitos

A Rede Mocambos atualmente envolve cerca de 200 comunidades localizadas em todo o território nacional brasileiro e algumas comunidades na Africa e na Europa. A conectividade no Brasil é por satélite, garantida pelo GESAC que disponibiliza para cada comunidade uma *Very Small Aperture Terminal* (VSAT) com banda de 512 kbit/s em download e 128 kbit/s em upload. A topologia da rede é a estrela então todos os nós comunicam por satélite concentrando o trafego num *hub* terrestre, onde a rede via satélite é interligada a Internet. Cada comunidade tem uma sala com 10 computadores com acesso público, recém instalados (ou sendo instalados) pelo Telecentros.BR[1], outro programa do governo federal. A população de casa comunidade varia desde as centenas as milhares de pessoas. A maioria das comunidade se encontra em área rural, geralmente de difícil acesso e sem outros meios de comunicação. Além dos espaços comunitários, normalmente concentrados na zona central, a população é dividida em pequenos núcleos familiares espalhados no território e as vezes muito distantes um dos outros.

Vejamos agora alguns requisitos essenciais para essa arquitetura de rede federada.

[1] "O Programa Nacional de Apoio à Inclusão Digital nas Comunidades – Telecentros.BR é uma ação do Governo Federal de apoio à implantação de novos espaços públicos e comunitários de inclusão digital e o fortalecimento dos que já estão em funcionamento em todo o território. São disponibilizados equipamentos de informática e mobiliário necessários ao funcionamento dos telecentros, serviços de conexão em banda larga à internet, assim como a formação e bolsas de auxílio financeiro para monitores atuarem como agentes de inclusão digital. Esses monitores bolsistas participam de um curso de formação e atendem as comunidades dos telecentros.", retirado de `http://www.inclusaodigital.gov.br/telecentros`.

3.1.1 Identidade de rede

Os usuários das comunidades tem que ter identidade digitais com as quais acessar e usar os serviços existentes. Além de acessar localmente, é importante ter acesso a alguns serviços também fora da própria comunidade. Por exemplo, no caso uma pessoa se encontra na cidade, ela deve poder acessar, através da internet, aos serviços de base, como o e-mail, o aos portais e serviços web da RM. É necessária então uma identidade digital de rede univoca.

3.1.2 Autenticação descentralizada

Um requisito essencial para cada comunidade é ter autonomia em ausência de conexão internet, e a presença então de um sistema local de autenticação e autorização. Além de garantir a continuidade do serviço (em relação a problemas da conexão por satélite), é um requisito importante para um bom desempenho de todos os serviços autenticados.

3.1.3 Sincronização

A troca de informações entre as comunidades é o objetivo primário para a RM. É então necessário um sistema para a sincronização seletiva dos conteúdos de interesse geral. Para otimizar o uso da banda, as operações de sincronização dos dados tem que influenciar o mínimo possível o uso quotidiano da internet, programando essas operações durante a noite ou quando a conexão por satélite não está sendo usada.

3.1.4 Replicabilidade

Cada comunidade se organiza autonomamente para a gestão da própria infraestrutura tecnológica e as soluções adotadas tem então que ser facilmente implementáveis e adaptáveis localmente. As ferramentas precisam ser então soluções estáveis e possivelmente de uso comum, também ao fim de encontrar mais facilmente assistência em loco.

3.1.5 Manutenção

A manutenção do sistema tem que prever seja intervenções a distância seja locais. O sistema é baseado em tecnologias *standard* e abertas para garantir o acesso aos dados mesmo em caso de problemas.

3.1.6 Desenvolvimento

Cada comunidade tem características e necessidades próprias que levam a serviços diferenciados. A arquitetura geral da RM tem que ser uma base estável em cima da qual poder desenvolver sem demais restrições. As tecnologias e as linguagens adotadas têm que facilitar a interação, a formação e o reuso dos conhecimentos.

3.2 Ferramentas e praticas para o desenvolvimento

Pela natureza da RM, e em particular pela necessidade de autonomia tecnológica, a formação é fundamental então é importante o uso de ferramentas que facilitem a documentação e o desenvolvimento colaborativo. A RM já utiliza a anos um wiki[2], disponível no endereço `http://wiki.mocambos.net`, onde se encontra parte da documentação sobre o sistema e o código desenvolvido para este trabalho. Para o versionamento do código foi usado o sistema descentralizado GIT (ver 2.3.8) e a plataforma GITHUB[3]. O código do protótipo é disponível no endereço `https://github.com/RedeMocambos`.

3.2.1 Sistema operacional

A escolha de um sistema operacional comum facilita a documentação, a automação e a assistência a distância. Muitas comunidades já usam distribuições GNU/Linux baseadas no Debian[4]. Para o protótipo foram escolhidas as ultimas versões estáveis do Debian, a 6.0, e de Ubuntu[5], a 11.10.

[2] "Un Wiki é uma página (ou uma coleção de hipertextos) que é atualizada pelos seus usuários e cujos conteúdos são desenvolvidos colaborativamente por todos que tenham acesso. A alteração dos conteúdos é aberta, no sentido que o texto pode ser modificado por todos os usuários (as vezes mesmo se anônimos) contribuindo não somente adicionando, como acontece normalmente nos fóruns, mas também mudando e excluindo o que outros escreveram. Cada mudança fica gravada em uma cronologia que permite, quando necessário, voltar a versão antecedente; o objetivo é compartilhar, trocar, armazenar e otimizar os conhecimentos de maneira colaborativa. O termo wiki indica também o software utilizado para criar o site web num servidor.", traduzido de `http://it.wikipedia.org/wiki/Wiki`.

[3] GITHUB è uma rede social baseada no sistema GIT 2.3.8.

[4] "O Projeto Debian é uma associação de indivíduos que têm como causa comum criar um sistema operacional livre. O sistema operacional que criamos é chamado Debian GNU/Linux, ou simplesmente Debian.", retirado de `http://www.debian.org`

[5] "Ubuntu é um sistema operacional GNU/Linux criado em 2004, baseado no Debian, focado no usuário e na facilidade de uso. Ubuntu é orientado para uso em desktop e dando muita atenção ao suporte do hardware. A cada seis meses é lançada uma nova versão. Financiado pela empresa Canonical Ltd (com registro na Ilha de Man), este sistema é compartilhado como Software Livre sob licença GNU GPL.", traduzido de `http://it.wikipedia.org/wiki/Ubuntu`.

3.2.2 Linguagens de programação

Para o sistema desenvolvido foi escolhida a linguagem de programação Python, uma linguagem muito flexível e provavelmente o mais versátil para conectar componentes heterogêneas. Várias comunidades já usam Software Livres como: *GIMP*[6], *Blender*[7], *Inkscape*[8] que também fazem uso de *scripting* Python para criação de filtros e para outras funcionalidades avançadas.

3.2.3 Virtualização

O protótipo foi desenvolvido e testado em um ambiente virtualizado, para garantir maior controle e verificar a exatidão dos procedimentos em todos seus passos. Os *script* para automatizar as instalações, e os procedimentos passo a passo na documentação, foram feitos a partir da configuração de base do sistema. Foram criadas máquinas virtuais, com a ajuda do software livre *VirtualBox*[9], simulando os servidores comunitários/locais e o servidor central.

3.3 Arquitetura base

3.3.1 Gestão das identidades de rede

Analisando a especificação dos requisitos e algumas das tecnologias existentes que vimos no capítulo precedente, foi escolhido o uso de LDAP para gerenciar as credenciais dos usuários da RM. O protótipo proposto prevê um servidor central em conectividade internet garantida e servidores locais em replicação. É possível uma configuração onde todos os servidores da rede sejam *master* (modalidade *N-Way-Multimaster*). Esta configuração, se por um lado permite a atualização em escritura das bases dos usuários em cada servidor, por outro lado pode gerar mais facilmente situações de inconsistência dos dados e problemas de sincronização entre os servidores LDAP. Por esta razão para o protótipo realizado se escolheu relaxar o requisito 3.1.2, assumindo um síngulo servidor *master* para toda a rede onde efetuar as operações de escritura (criação, eliminação, e atualização dos usuários) e servidores locais habilitados somente para operações em

[6] *GNU Image Manipulation Program (GIMP)* é um programa para edição, montagem e criação de imagens. Disponível em `http://www.gimp.org/`.

[7] *Blender* é um programa para a criação de gráfica e ambientes tridimensionais. Disponível em `http://www.blender.org/`.

[8] *Inkscape* é um programa de gráfica vectorial conforme ao formado SVG. Disponível em `http://inkscape.org`.

[9] *VirtualBox* é um virtualizador completo para arquiteturas x86 para uso em servidores, desktop e sistemas embutidos. Disponível em `http://www.virtualbox.org/`.

leitura. Cada comunidade então teria a disposição um servidor LDAP replicado para autenticação e autorização. Contudo, na falta da conexão esterna, os serviços locais permanecem ativos para os usuários habilitados até a ultima sincronização.

3.3.2 Mocambos_LDAP

O código esta disponível em `https://github.com/RedeMocambos/Mocambos_LDAP`

Para facilitar a implementação dos servidores LDAP por parte dos administradores locais, foi desenvolvido um *script* de instalação, que pode se tornar útil também para um administrador remoto. O *script* instala os pacotes necessários e prepara o servidor LDAP *slapd* em configuração *master* ou replica. Também cria um DIT pré-configurado para a RM com uma simples estrutura (ver figura 3.1).

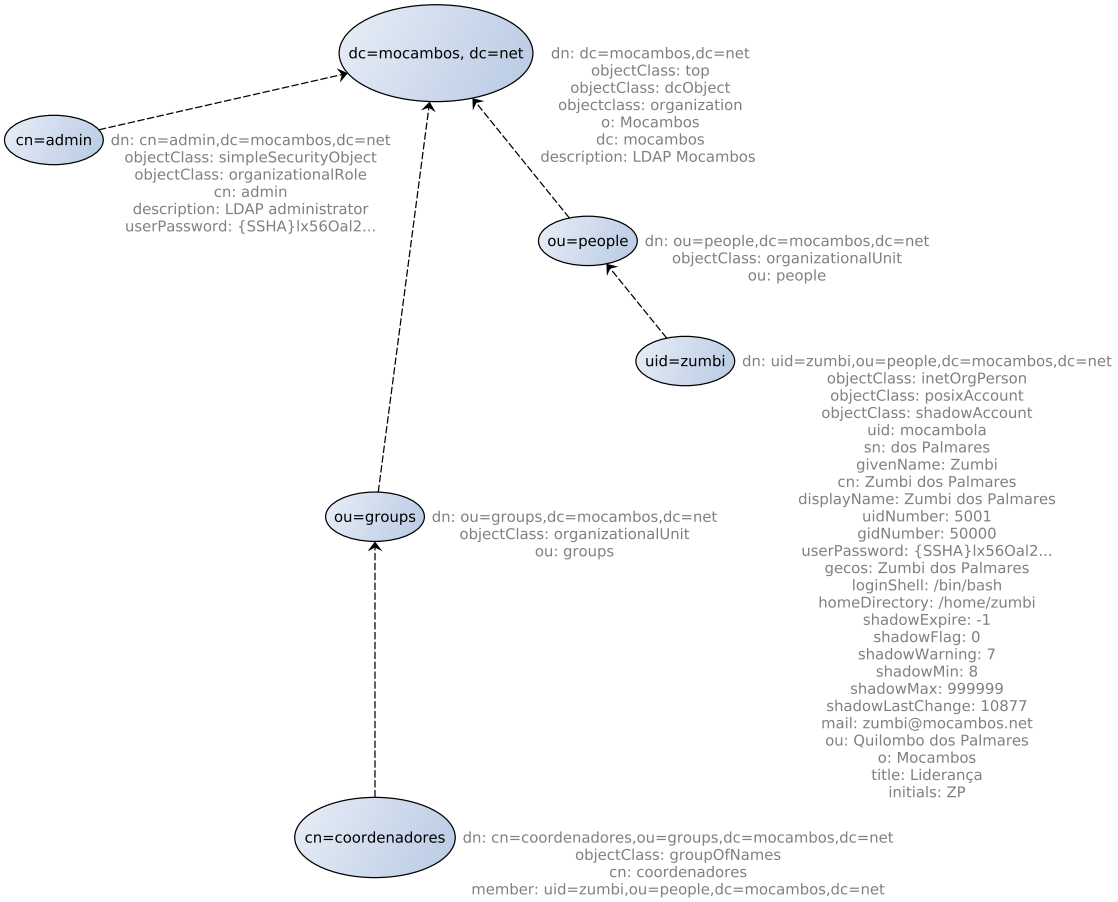

FIGURA 3.1: DIT base do servidor LDAP da RM.

Capítulo 4

Um protótipo de serviço federado

O uso de tecnologias digitais oferece novas possibilidades para o ensino e a formação. Muitas comunidades, aferentes a RM, vem produzindo material pedagógico audiovisual, que com a ajuda da rede, poderiam contribuir para enriquecer os programas educacionais públicos, geralmente deficientes e pouco integrados com a cultura quilombola. O *Núcleo de Pesquisa e Desenvolvimento Digital (NPDD)*[1] da RM, com o projeto *Tambor e Comunicação*[2], propus a pesquisa e o desenvolvimento de uma solução para a publicação e compartilhamento em rede de imagens, áudios e vídeos de interesse comum e geralmente produzidos nas comunidades.

4.1 Sistema de publicação e sincronização de conteúdos multimídia

O sistema desenvolvido prevê a instalação de um portal no servidor local das comunidades através do qual é possível visualizar e publicar conteúdos multimídia aproveitando a rapidez da rede local. O sistema cuida de memorizar os conteúdos em um acervo multimídia local e compartilhar, aqueles etiquetados como de interesse comum, com os servidores das outras comunidade (ver figura 4.1). O protótipo tenta resolver as limitações de banda da conexão respeitando a especificação dos requisitos.

[1] O *Núcleo de Pesquisa e Desenvolvimento Digital (NPDD)* da RM pesquisa e desenvolve tecnologias para a comunicação, a produção de energias renováveis e sustentáveis, e a melhoria das condições de vida em simbiose com o ambiente. Mais informações em `http://wiki.mocambos.net/wiki/NPDD`.

[2] O projeto *Tambor e Comunicação* tenta fortificar a rede de comunicação digital seguindo as necessidades das comunidades. Ver `http://wiki.mocambos.net/wiki/Projeto_Tambor_e_Comunicacao`.

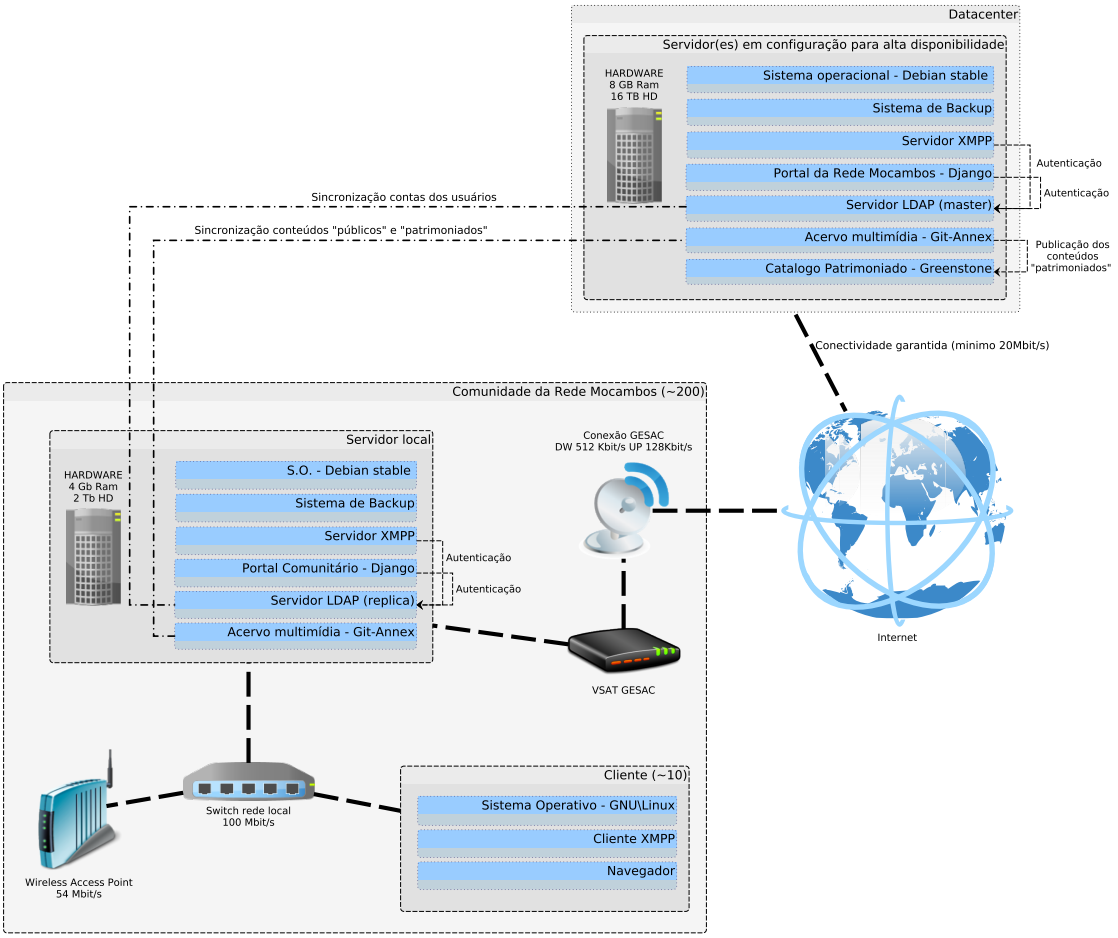

FIGURA 4.1: Esquema da infraestrutura da RM.

4.2 Acervo multimídia

O acervo multimídia local da comunidade é um repositório *git-annex* (ver 4.2.1) que vem gerenciado através de um portal comunitário, seja pela publicação seja pela visualização dos conteúdos. O acesso direto aos dados é todavia garantido sendo esses arquivos acessíveis no disco. Também é possível interagir com os dados através do amplo universo de aplicações e bibliotecas do *git*. Em especial o protótipo usa a estrutura de metadados do *git* para manter o controle do usuário que publicou um conteúdo. A operação de *commit* coincide de fato com a publicação de um conteúdo, e o *committer* com o usuário que o publica. Os metadados, em vez, relativos ao conteúdo multimídia em si, como autor, tipo de licenciamento, data de criação, podem ser memorizados internamente ao arquivo seguindo os padrões existentes como o *Dublin Core*[3].

[3]"O Dublin Core é um sistema de metadados que contempla um núcleo de elementos essenciais para a descrição de qualquer conteúdo digital acessível via rede.", traduzido de Wikipedia: `http://it.wikipedia.org/wiki/Dublin_Core`.

Um elemento importante para o acervo multimídia são as operações de sincronização. As ferramentas baseadas no *git* herdam a sua natureza descentralizada e a capacidade de comunicar de forma transparente usando vários protocolos. Em particular é interessante a possibilidade de executar sincronizações com sistemas de armazenamento massivo, característica essencial na fase de criação de um novo nó, onde a primeira sincronização via rede poderia levar dias (ver requisito 3.1.3). As transferências contudo, no caso do *git-annex*, são executadas através do protocolo *rsync*[4], que gerencia eventuais interrupções, evitando retransmissões onerosas.

4.2.1 git-annex

git-annex[5] permite a gestão de arquivos com *git*, sem a necessidade de adicionar os arquivos dentro *git*. Mesmo se pode parecer paradoxal, é útil quando se trabalha com arquivos muito grandes que *git* atualmente não gerencia facilmente por limitações devidas a memoria, tempo ou espaço no disco.

Mesmo sem manter o histórico das mudanças do conteúdo do arquivo, ter a possibilidade de gerenciar arquivos com *git*, de movê-los, e exclui-los, numa árvore de pasta versionada, com uso de *branches* e de clones distribuídos, são todos bons motivos para usar *git*. E os arquivos anexos (por isso o nome *git-annex*) podem coexistir no mesmo repositório *git* com os arquivos regularmente versionados.

git-annex transforma os arquivos anexos em *link* simbólicos, que são normalmente versionados por *git*.

O conteúdo dos arquivos é mantido por *git-annex* em um acervo chave/valor distribuído que corresponde aos clones de um dado repositório *git*. Praticamente *git-annex* memoriza o conteúdo do arquivo em uma subpasta de `.git/annex/`.

A primeira vez que um arquivo é adicionado no *git-annex*, é calculada uma chave, normalmente fazendo um *hash* do seu conteúdo. *git-annex* todavia suporta vários *backend* que podem produzir diferentes tipos de chaves. O arquivo que é adicionado no *git* nada mais é que um *link* simbólico para a chave memorizada no `.git/annex/`. Se o conteúdo do arquivo for modificado, é gerada uma outra chave, e o *link* é alterado.

O conteúdo do arquivo pode ser transferido de um repositório para outro por *git-annex*, que além de manter controle de quem mantem o que, permite criar um mapa das copias disponíveis e impor um número mínimo de cópias. Essas informações são mantidas

[4] *rsync* é um Software Livre para a transfêrencia rápida e incremental de arquivos disponível no `http://rsync.samba.org/`.

[5] *git-annex* é um programa que estende as funcionalidades do *git* em gerir arquivos de grande tamanho disponível no `http://git-annex.branchable.com`.

em um *branch* separado, chamado "*git-annex*", e as operações de sincronização, são simplesmente *push* e *pull* entre os vários clones dos repositórios.

git-annex suporta:

- localização das cópias (*location tracking*)
- download seletivo dos conteúdos
- gestão da confiança dos repositórios
- gestão do número minimo de cópias
- vários *backend* para as chaves (SHA[6], WORM[7])
- vários *backend* para os conteúdos/valores (BUP[8], rsync, web, S3[9])

4.3 Portal Comunitário

O portal local tem que oferecer acesso aos principais serviços locais da comunidade. Para o desenvolvimento foi escolhido o uso de um framework baseado no Python, *Django* (ver 2.3.7), que possibilita uma integração flexível e avançada com outros sistemas graças as numerosas bibliotecas disponíveis como, por exemplo, para a autenticação LDAP.

Para a instalação do framework e do modelo de base do protótipo de portal comunitário foi criado um *script* enquanto, para gerir o acervo multimídia *git-annex* foram desenvolvidos duas aplicações para *Django*, que definem o modelo dos dados e cuidam de adicionar os conteúdos no repositório executando as operações de *commit*, *push* e *pull*.

4.3.1 Mocambos_Portal_Local

O código esta disponível no `https://github.com/RedeMocambos/Mocambos_Portal_Local`

O *script bash*, `script/install-django-env.sh`, instala os pacotes de sistema necessários, e também cria o ambiente virtual, usando o programa *virtualenv*[10], que permite instalar facilmente versões específicas de bibliotecas e do interprete Pyhton, além do

[6] Secure Hash Algoritm, (SHA), é um algoritmo usado em sistemas chave/valor onde as chaves são calculadas através de uma função criptográfica dos valores.

[7] O algoritmo WORM identifica os arquivos em base ao nome, dimensão e data de alteração.

[8] BUP é um sistema para *backup* a alta eficiência disponível no: `https://github.com/apenwarr/bup`.

[9] Amazon Simple Storage Service, (S3) é uma infraestrutura para a memorização dos dados totalmente redundante, disponível no: `aws.amazon.com/`.

[10] *Virtualenv* è um Software Livre para criar ambientes Python isolados disponível no `http://www.virtualenv.org`.

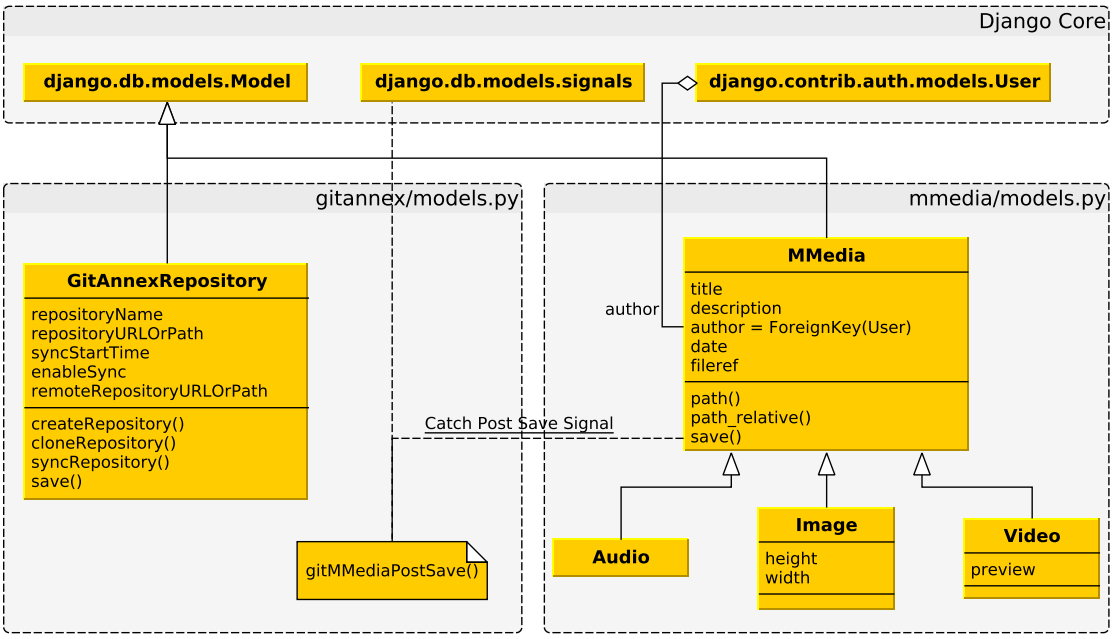

FIGURA 4.2: Esquema UML das aplicações Django.

Django, sem alterar as versões já instaladas no sistema. Na pasta `exemplos/` se encontram alguns arquivos de configuração de exemplo, pré-configurados para a conexão ao servidor LDAP da RM (ver 3.3.2).

4.3.2 Django mmedia

O código esta disponível no `https://github.com/RedeMocambos/mmedia`

A aplicação *mmedia* implementa o modelo de dados multimídia com suporte par o armazenamento em repositório *git-annex*.

Os modelos, em *Django*, precisam ser criados no arquivo `model.py`, onde, primeiramente, definimos a estrutura de base da classe *MMedia*, com os atributos comuns para todos os objetos multimídia. As classes *Audio*, *Image* e *Video*, herdam os atributos da classe abstrata "MMedia", acrescentando outros atributos específicos para o tipo de objeto.

Os objetos são salvos no *database*, e serializados no disco através o *overriding* da função *save()*:

```
def save(self, *args, **kwargs):
    logger.debug(type(self))
    serializeTo = os.path.join(settings.MEDIA_ROOT,\
                               settings.GITANNEX_DIR,\
                               settings.PORTAL_NAME,\
                               settings.SERIALIZED_DIR,\
                               os.path.basename(self.fileref.path)+ '.xml')
    logger.info('>>>> Serialize to: ' + serializeTo)
    out = open(serializeTo, "w")
    XMLSerializer = serializers.get_serializer("xml")
    xml_serializer = XMLSerializer()
    xml_serializer.serialize((self, ), stream=out)
    super(MMedia, self).save(*args, **kwargs)
```

4.3.3 Django gitannex

O código esta disponível no https://github.com/RedeMocambos/gitannex

A aplicação *gitannex* implementa parte do modelo de dados de um repositório *git-annex* no *Django*, acrescentando os atributos e as funcionalidades necessárias para a planejamento da sincronização.

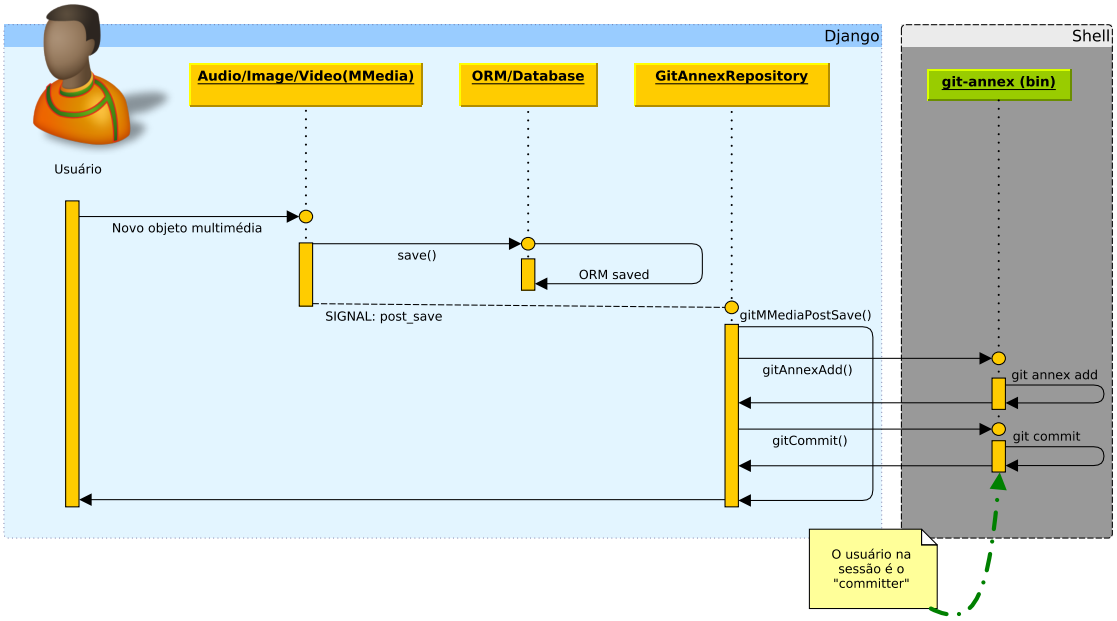

FIGURA 4.3: Diagrama de sequencia da criação de um novo objeto multimídia.

Seguindo a especificação, é definido, através do atributo *syncStartTime*, um horário para o início da sincronização, que é inicializada pela função *runScheduledJobs()*. Para interagir mais facilmente com o framework através do terminal é possível definir comandos

que precisam ser criados na pasta `management/commands/`, onde se encontra, por exemplo, o comando `run_scheduled_jobs` que chama a função homônima. Desta maneira é possível executar operações planejando-as através do *cron*[11] diretamente no terminal com:

```
zumbi@palmares:~$ python manage.py run_scheduled_jobs
```

Django providência um sistema de sinais, enviados em concomitância de operações como a *save()* de um objeto, que podem ser interceptados em outras partes do sistema. A aplicação *gitannex* intercepta o sinal padrão do Django, *post-save* (ver figura 4.3), em objetos que herdam da classe *MMedia*[12], através da função *gitMMediaPostSave()*:

```
@receiver_subclasses(post_save, MMedia, ``mmedia_post_save'')
def gitMMediaPostSave(instance, **kwargs):
    logger.debug(instance.mediatype)
    logger.debug(type(instance))
    logger.debug(instance.path_relative())

    path = instance.path_relative().split(os.sep)
    if gitannex_dir in path:
        repositoryName = path[path.index(gitannex_dir) + 1]
        gitAnnexRep = GitAnnexRepository.objects.get(\
                      repositoryName__iexact=repositoryName)
        gitAnnexAdd(os.path.basename(instance.fileref.name),\
                    os.path.dirname(instance.fileref.path))
        gitCommit(instance.title, instance.author.username,\
                  instance.author.email, os.path.dirname(instance.fileref.path))
```

[11] "Em sistemas operacionais Unix e Unix-like, o comando crontab permite o planejamento de comandos, ou seja permite de agendá-los no sistema para serem executados periodicamente.", traduzido de Wikipedia: `http://it.wikipedia.org/wiki/Crontab`.

[12] Django não suporta a "filtragem" de sinais enviados por subclasses de uma classe dada, neste caso *MMedia*. Para isso, foi usado um truque encontrado na rede (ver o código no arquivo `gitannex/signals.py`).

Capítulo 5

Conclusões e desenvolvimentos futuros

A arquitetura e o protótipo desenvolvidos são em curso de implementação, em escada reduzida, a partir das comunidades mais estruturadas que tem já um papel de polos regionais, chamados *Núcleos de Formação Continuada, (NFC)*, que são atualmente dez, geograficamente bem espalhados no território brasileiro.

Seguindo a filosofia de desenvolvimento Agile [5], pela qual:

> "*Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.*"[1]

é importante, em um contexto assim amplo e heterogêneo, ter uma estrutura geral em cima da qual pode-se desenvolver código funcionante, que possa ser melhorado e evoluído no tempo.

Atualmente o código suporta:

- autenticação LDAP (com gestão básica dos grupos)
- criação e upload de conteúdos áudios, imagens e vídeos
- distribuição através do *git-annex*

[1] "O desenvolvimento de software Agile é um conjunto de metodologias baseadas no desenvolvimento iterativo e incremental, onde os requisitos e as soluções evolvem através a colaboração entre equipe auto-organizadas e multifuncionais.", traduzido de Wikipedia: `http://en.wikipedia.org/wiki/Agile_software_development`.

- sincronização dos objetos em portais Django (recriando os objetos relativos aos conteúdos distribuídos via *git-annex*)

Do ponto de vista estrutural a componente mais importante é o sistema de gestão das identidades federadas digitais. Portanto é essencial uma experimentação aprofundada da solução LDAP adotada que, contudo, pode ser estendida e integrada com outros mecanismos. Para mensagens instantâneas e VOIP, seria interessante explorar as possibilidades oferecidas pelo protocolo XMPP, que se destaca pela facilidade de integração em sistemas heterogêneos. Por exemplo a integração entre XMPP e OpenID, onde, uma vez autenticados, se recebem os pedidos de autorização a acessar a serviços terceiros, através de mensagens instantâneas[2]. Neste sentido o trio LDAP, XMPP e OpenID poderia oferecer uma boa estrutura para serviços federados respetivamente "desktop", "client" e "web oriented".

No que diz respeito ao protótipo, a parte implementada constitui um primeiro passo, e a prova no campo é uma passagem fundamental para testar as capacidades reais e os limites que ainda existem. A distribuição de arquivos de grande tamanho, além dos limites de banda, implica um considerável uso de espaço no disco. É necessário então impor limites, por exemplo, nas dimensões máximas dos arquivos aceitos pelo sistema.

A ser implementados ou melhorado nas próximas versões:

- transferência seletiva dos conteúdos/valores baseado no uso estatístico o sob pedido
- desenvolvimento de uma interface de visualização e publicação para o usuário final
- gestão do DIT do LDAP através de *script* e/ou portal

[2]Uma implementação pública deste sistema é disponível no site: `http://openid.xmpp.za.net/`.

Apêndice A

Listagem do código

A.1 gitannex

```
          /
     _ \ /
      \_| |/__
 <==    | /  ()
 ==>    |/   NPDD/Rede Mocambos
 nnn    |
        |    http://wiki.mocambos.net/wiki/NPDD
        |    "Vamos fazer um mundo digital mais do nosso jeito..."
______/~~~\_____________________________________________________________ - - -

       /
      GITANNEX                                    Software LIVRE! GPLv3

Aplicacao django para gerir repositorios Git Annex (git-annex.branchable.com)


Pode usar com o django mmedia:
https://github.com/RedeMocambos/mmedia
```

A.1.1 gitannex/admin.py

```
from gitannex.models import GitAnnexRepository
from django.contrib import admin

"""
Arquivo de configuracao e customizacao da interface administrativa do Django.
"""

admin.site.register(GitAnnexRepository)
```

A.1.2 gitannex/models.py

```
from django.db import models
from django.db.models.base import ModelBase
from django.contrib.auth.models import User
from django.conf import settings

from django.db.models.signals import post_save
from django.dispatch import receiver

from gitannex.signals import receiver_subclasses, filesync_done
from mmedia.models import MMedia, Audio

import os
import datetime
import subprocess
import logging

"""
Modelos da aplicacao Django.

Neste arquivo sao definidos os modelos de dados da aplicacao *gitannex*.
"""

logger = logging.getLogger(__name__)
gitannex_dir = settings.GITANNEX_DIR

def _createRepository(repositoryName, remoteRepositoryURLOrPath):
    """Cria e inicializa um repositorio *git-annex*."""
    logger.info('git config --global user.name "admin"')
    cmd = 'git config --global user.name "admin"'
    pipe = subprocess.Popen(cmd, shell=True, cwd=os.path.join(settings.MEDIA_ROOT, gitannex_dir,
     repositoryName))
    pipe.wait()
    logger.info('git config --global user.email "admin@mocambos.net"')
    cmd = 'git config --global user.email "admin@mocambos.net"'
    pipe = subprocess.Popen(cmd, shell=True, cwd=os.path.join(settings.MEDIA_ROOT, gitannex_dir,
     repositoryName))
    pipe.wait()
    logger.info('git init')
    cmd = 'git init'
    pipe = subprocess.Popen(cmd, shell=True, cwd=os.path.join(settings.MEDIA_ROOT, gitannex_dir,
     repositoryName))
    pipe.wait()
    logger.info('git annex init ' + settings.PORTAL_NAME)
    cmd = 'git annex init ' + settings.PORTAL_NAME
    pipe = subprocess.Popen(cmd, shell=True, cwd=os.path.join(settings.MEDIA_ROOT, gitannex_dir,
     repositoryName))
    pipe.wait()
    # TODO: Manage repositories dinamically
    logger.info('git remote add baoba ' + remoteRepositoryURLOrPath)
    cmd = 'git remote add baoba ' + remoteRepositoryURLOrPath
    pipe = subprocess.Popen(cmd, shell=True, cwd=os.path.join(settings.MEDIA_ROOT, gitannex_dir,
     repositoryName))
    pipe.wait()
```

```
def _cloneRepository(repositoryURLOrPath, repositoryName):
    """Clona e inicializa um repositorio *git-annex*."""
    cmd = 'git config --global user.name "admin"'
    pipe = subprocess.Popen(cmd, shell=True, cwd=os.path.join(settings.MEDIA_ROOT, gitannex_dir))
    pipe.wait()
    cmd = 'git config --global user.email "admin@mocambos.net"'
    pipe = subprocess.Popen(cmd, shell=True, cwd=os.path.join(settings.MEDIA_ROOT, gitannex_dir))
    pipe.wait()
    cmd = 'git clone ' + repositoryURLOrPath + repositoryName
    pipe = subprocess.Popen(cmd, shell=True, cwd=os.path.join(settings.MEDIA_ROOT, gitannex_dir))
    pipe.wait()
    cmd = 'git annex init ' + settings.PORTAL_NAME
    pipe = subprocess.Popen(cmd, shell=True, cwd=os.path.join(settings.MEDIA_ROOT, gitannex_dir,
     repositoryName))
    pipe.wait()

def _selectRepositoryByPath():
    # Controlla il path del file ed estrai il nome del Repository
    return

def _getAvailableFolders(path):
    """Procura as pastas que podem ser inicializada como repositorio, retorna a lista das
     pastas."""
    folderList = [( name , name ) for name in os.listdir(os.path.join(path, gitannex_dir)) \
                      if os.path.isdir(os.path.join(path, gitannex_dir, name))]
    return folderList

def gitCommit(fileTitle, authorName, authorEmail, repoDir):
    """Executa o *commit* no repositorio impostando os dados do author."""
    logger.info('git commit --author="' + authorName + ' <' + authorEmail +'>" -m "' + fileTitle
     + '"')
    cmd = 'git commit --author="' + authorName + ' <' + authorEmail +'>" -m "' + fileTitle + '"'
    pipe = subprocess.Popen(cmd, shell=True, cwd=repoDir)
    pipe.wait()

def gitPush(repoDir):
    """Executa o *push* do repositorio, atualizando o repositorio de origem."""
    logger.info('git push ')
    cmd = 'git push '
    pipe = subprocess.Popen(cmd, shell=True, cwd=repoDir)
    pipe.wait()

def gitPull(repoDir):
    """Executa o *pull* do repositorio, atualizando o repositorio local."""
    logger.info('git pull ')
    cmd = 'git pull '
    pipe = subprocess.Popen(cmd, shell=True, cwd=repoDir)
    pipe.wait()

def gitStatus(fileName, repoDir):
    """Verifica o estado atual do repositorio"""
    # Dovrebbe restituire oltre allo status un flag per avviare o no il sync
    cmd = 'git status'
```

```
def gitGetSHA(repoDir):
    """Resgata o codigo identificativo (SHA) da ultima revisao do repositorio, retorna o
     codigo."""
    logger.info('git rev-parse HEAD')
    cmd = 'git rev-parse HEAD'
    pipe = subprocess.Popen(cmd, shell=True, cwd=repoDir)
    output,error = pipe.communicate()
    logger.debug('>>> Revision is: ' + output)
    return output

def gitAnnexAdd(fileName, repoDir):
    """Adiciona um arquivo no repositorio *git-annex*."""
    logger.info('git annex add ' + fileName)
    cmd = 'git annex add ' + fileName
    pipe = subprocess.Popen(cmd, shell=True, cwd=repoDir)
    pipe.wait()

def gitAnnexMerge(repoDir):
    """Executa o *merge* do repositorio, reunindo eventuais diferencias entre o repositorio local
     e remoto."""
    logger.info('git annex merge ')
    cmd = 'git annex merge '
    pipe = subprocess.Popen(cmd, shell=True, cwd=repoDir)
    pipe.wait()

def gitAnnexCopyTo(repoDir):
    """Envia os conteudos binarios para o repositorio remoto."""
    # TODO: Next release with dynamic "origin"
    logger.info('git annex copy --fast --to origin ')
    cmd = 'git annex copy --fast --to origin'
    pipe = subprocess.Popen(cmd, shell=True, cwd=repoDir)
    pipe.wait()

def gitAnnexGet(repoDir):
    """Baixa os conteudos binarios desde o repositorio remoto."""
    # TODO: Next release with possibility to choice what to get
    logger.info('git annex get .')
    cmd = 'git annex get .'
    pipe = subprocess.Popen(cmd, shell=True, cwd=repoDir)
    pipe.wait()

# Connecting to MMedia signal
@receiver_subclasses(post_save, MMedia, "mmedia_post_save")
def gitMMediaPostSave(instance, **kwargs):
    """Intercepta o sinal de *post_save* de objetos multimedia (*mmedia*) e adiciona o objeto ao
     repositorio."""
    logger.debug(instance.mediatype)
    logger.debug(type(instance))
    logger.debug(instance.path_relative())

    path = instance.path_relative().split(os.sep)
    if gitannex_dir in path:
        repositoryName = path[path.index(gitannex_dir) + 1]
        gitAnnexRep = GitAnnexRepository.objects.get(repositoryName__iexact=repositoryName)
```

```
        gitAnnexAdd(os.path.basename(instance.fileref.name),
     os.path.dirname(instance.fileref.path))
        gitCommit(instance.title, instance.author.username, instance.author.email,
     os.path.dirname(instance.fileref.path))

def runScheduledJobs():
    """Executa as operacoes programadas em todos os repositorios. """
    allRep = GitAnnexRepository.objects.all()
    for rep in allRep:
        if rep.enableSync:
            # TODO: Manage time of syncing
            # if rep.syncStartTime >= datetime.datetime.now():
            rep.syncRepository()


class GitAnnexRepository(models.Model):
    """Classe de implementacao do modelo de repositorio *git-annex*.

    Atributos:
        repositoryName: nome do repositorio (campo preenchido por *_getAvailableFolders()*)
        repositoryURLOrPath: apontador ao repositorio no disco ou em rede
        syncStartTime = orario de inicio da sincronizacao
        enableSync = flag booleano para abilitar ou disabilitar a sincronizacao
        remoteRepositoryURLOrPath = apontador ao repositorio de origem
    """

    # Forse dovrei mettere qualcosa nella view. Esattamente.. Quando
    # creo un repository questo puo' essere locale o remoto.  Quindi
    # devo poter scegliere tra una cartella locale (eventualmente
    # crearla), o inserite un URL per effetuare il clone (via ssh).
    # Nella view va messo un if che a seconda chiama create o
    # cloneRepository a seconda della scelta.

    repositoryName = models.CharField(max_length=60,
     choices=_getAvailableFolders(settings.MEDIA_ROOT))
    repositoryURLOrPath = models.CharField(max_length=200)
    syncStartTime = models.DateField()
    enableSync = models.BooleanField()
    remoteRepositoryURLOrPath = models.CharField(max_length=200)
#    lastSyncSHA = models.CharField(max_length=100)

    def createRepository(self):
        """Cria e inicializa o repositorio."""
        # Dovrebbe scegliere tra remoto e locale?
        _createRepository(self.repositoryName, self.remoteRepositoryURLOrPath)

    def cloneRepository(self):
        """Clona e inicializa o repositorio."""
        _cloneRepository(self.repositoryURLOrPath, self.repositoryName)

    def syncRepository(self):
        """Sincroniza o repositorio com sua origem."""
        gitPull(self.repositoryURLOrPath)
        gitAnnexMerge(self.repositoryURLOrPath)
        gitPush(self.repositoryURLOrPath)
```

```
        gitAnnexCopyTo(self.repositoryURLOrPath)
        # TODO: Next release with possibility to choice what to get
        gitAnnexGet(self.repositoryURLOrPath)
        # TODO: Next release with selective sync since a given revision (using git SHA)
        # self.lastSyncSHA = gitGetSHA(self.repositoryURLOrPath)
        # Signal to all that files are (should be) synced
        logger.debug(">>> BEFORE filesync_done")
        filesync_done.send(sender=self, repositoryName=self.repositoryName, \
                           repositoryDir=self.repositoryURLOrPath)
        logger.debug(">>> AFTER filesync_done")

    def save(self, *args, **kwargs):
        self.createRepository()
        super(GitAnnexRepository, self).save(*args, **kwargs)
```

A.1.3 gitannex/signals.py

```
import logging
import django.dispatch

"""
Arquivo de definicao dos sinais.

Os sinais sao usados para interligar diferentes *apps* do Django.
"""

def get_subclasses(classes, level=0):
    """
    Procura as subclasses da uma classe dada, retorna a lista de subclasses.

    Return the list of all subclasses given class (or list of classes) has.
    Inspired by this question:

     http://stackoverflow.com/questions/3862310/how-can-i-find-all-subclasses-of-a-given-class-in-python
    """
    # for convenience, only one class can can be accepted as argument
    # converting to list if this is the case
    if not isinstance(classes, list):
        classes = [classes]

    if level < len(classes):
        classes += classes[level].__subclasses__()
        return get_subclasses(classes, level+1)
    else:
        return classes

def receiver_subclasses(signal, sender, dispatch_uid_prefix, **kwargs):
    """
    Decorador para conectar todos os sinais do *receiver* e das subclasses do *receiver*.

    A decorator for connecting receivers and all receiver's subclasses to signals. Used by
     passing in the
```

```
    signal and keyword arguments to connect::

        @receiver_subclasses(post_save, sender=MyModel)
        def signal_receiver(sender, **kwargs):
            ...
    """
    def _decorator(func):
        all_senders = get_subclasses(sender)
        logging.info(all_senders)
        for snd in all_senders:
            signal.connect(func, sender=snd, dispatch_uid=dispatch_uid_prefix+'_'+snd.__name__,
     **kwargs)
        return func
    return _decorator


## Novo sinal para alertar que os repositorios sao sincronizados
filesync_done = django.dispatch.Signal(providing_args=["repositoryName", "repositoryDir"])
```

A.1.4 gitannex/management/commands/run scheduled jobs.py

```
from django.core.management.base import NoArgsCommand, CommandError

from gitannex.models import GitAnnexRepository, runScheduledJobs

"""
Definicoes do comando para executar as operacoes planejadas.
"""

class Command(NoArgsCommand):
    """Executa as operacoes planejadas."""
    help = 'Run scheduled jobs related to git repositories'

    def handle_noargs(self, **options):
        runScheduledJobs()
```

A.2 mmedia

```
          /
     _ \ /
      \_| |/__
 <==    | /  ()
 ==>    |/   NPDD/Rede Mocambos
 nnn    |
        |    http://wiki.mocambos.net/wiki/NPDD
        |    "Vamos fazer um mundo digital mais do nosso jeito..."
______/~~~\_____________________________________________________________ - - -
       /
```

```
      MMEDIA                                    Software Livre! GPLv3

Aplicacao Django para gerir arquivos multimediais.

Trabalha junto a aplicacao gitannex (https://github.com/RedeMocambos/gitannex)
para o sync dos arquivos entre instalacoes django.
```

A.2.1 mmedia/admin.py

```
from mmedia.models import Video, Audio, Image
from django.contrib import admin

"""
Arquivo de configuracao e customizacao da interface administrativa do Django.
"""

admin.site.register(Audio)
admin.site.register(Video)
admin.site.register(Image)
```

A.2.2 mmedia/models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.conf import settings
from django.utils.translation import ugettext_lazy as _
from django.core import serializers

import os
import logging

"""
Modelos da aplicacao Django.

Neste arquivo sao definidos os modelos de dados da aplicacao *mmedia*.
"""


logger = logging.getLogger(__name__)

def _path(instance):
    """Constroe o caminho (path) do arquivo relativo a um objeto *mmedia*, retorna o caminho
     (path) do arquivo."""
    return os.path.join(settings.MEDIA_ROOT, instance.user.username, \
                            instance.filename)

def _path_to_upload(instance, filename):
    """Constroe o caminho (path) para armazenar o arquivo relativo a um objeto *mmedia*, retorna
     o caminho (path) para armazenar o arquivo."""
    # TODO: Next release should manage gin annex directory dynamically
```

```
    return os.path.join(settings.GITANNEX_DIR, settings.PORTAL_NAME, instance.author.username,
     instance.mediatype, filename)

def createObjectFromFiles():
    """Recria os objetos em Django a partir dos objetos serializados"""
    pass


class MMedia(models.Model):
    """Classe abstrata com o modelo geral dos objetos multimediais.

    Atributos:
        title: nome do conteudo multimedial
        description: descricao do conteudo multimedial
        author: chave externa (agregacao) para objeto *User*
        date: data de publicacao
        fileref: apontador para o arquivo no disco (objeto FileField)
        mediatype: etiqueta para definir o tipo de conteudo
    """

    def __init__(self, *args, **kwargs):
        super(MMedia, self).__init__(*args, **kwargs)

    title = models.CharField(_('title'), max_length=120)
    description = models.TextField(_('description'), blank=True)
    author = models.ForeignKey(User)
    date = models.DateTimeField(_('release date'), blank=True, null=True)
    fileref = models.FileField(upload_to=_path_to_upload)
    mediatype = "mmedia"
#    tags = TagField(verbose_name=_('tags'), help_text=tagfield_help_text)

    def path(self):
        """Constroe o caminho (path) absoluto do arquivo do objeto, retorna o caminho (path)
     absoluto do arquivo em disco do objeto."""
        return os.path.join(settings.MEDIA_ROOT, settings.GITANNEX_DIR, settings.PORTAL_NAME,
     self.author.username, self.mediatype, \
                                self.fileref.path)

    def path_relative(self):
        """Constroe o caminho (path) relativo do arquivo do objeto, retorna o caminho (path)
     relativo do arquivem disco do objeto."""
        return os.path.join(settings.GITANNEX_DIR, settings.PORTAL_NAME, self.author.username,
     self.mediatype, \
                                self.fileref.path)

    def __unicode__(self):
        return self.title

    class Meta:
        abstract = True

    def save(self, *args, **kwargs):
        """Sobrescreve (override) a funcao generica *save()* incluindo a serializacao do objeto.

        Serializa o objeto em XML na pasta /MEDIA_ROOT/GITANNEX_DIR/PORTAL_NAME/SERIALIZED_DIR/
```

```
        """
        logger.debug(type(self))
        serializeTo = os.path.join(settings.MEDIA_ROOT, settings.GITANNEX_DIR,\
                                   settings.PORTAL_NAME, settings.SERIALIZED_DIR,\
                                   os.path.basename(self.fileref.path) + '.xml')
        logger.info('>>>> Serialize to: ' + serializeTo)
        out = open(serializeTo, "w")
        XMLSerializer = serializers.get_serializer("xml")
        xml_serializer = XMLSerializer()
        xml_serializer.serialize((self, ), stream=out)
        super(MMedia, self).save(*args, **kwargs)

class Audio(MMedia):
    """Implementa o modelo de dados de um arquivo de audio.

    Implementa e especializa a classe abstrata *MMedia* para conteudos de tipo audio.

    Atributos:
        mediatype: etiqueta para definir o tipo de conteudo
    """
    mediatype = "audio"

class Image(MMedia):
    """Implementa o modelo de dados de um arquivo de imagem.

    Implementa e especializa a classe abstrata *MMedia* para conteudos de tipo imagem.

    Atributos:
        mediatype: etiqueta para definir o tipo de conteudo
        height: altura da imagem
        width: largura da imagem
    """

    mediatype = "image"
    height = models.IntegerField(max_length=4)
    width = models.IntegerField(max_length=4)

    def get_tiny_object(self):
        return self.mediatype

    class Meta:
        verbose_name = _('image')
        verbose_name_plural = _('images')

class Video(MMedia):
    """Implementa o modelo de dados de um arquivo de video.

    Implementa e especializa a classe abstrata *MMedia* para conteudos de tipo video.

    Atributos:
        mediatype: etiqueta para definir o tipo de conteudo
        preview: apontador para uma imagem de anteprima (objeto ImageField)
    """

    mediatype = "video"
```

```
    preview = models.ImageField(upload_to="video_thumbnails")

    def get_tiny_object(self):
        return self.mediatype

    class Meta:
        verbose_name = _('video')
        verbose_name_plural = _('videos')
```

A.2.3 mmedia/signals.py

```
from django.dispatch import receiver
from django.core import serializers
from django.conf import settings

from gitannex.signals import filesync_done
from gitannex.models import GitAnnexRepository

import os

"""
Arquivo de definicao dos sinais.

Os sinais sao usados para interligar diferentes *apps* do Django.
"""


@receiver(filesync_done, sender=GitAnnexRepository)
def syncGitAnnexRepository(sender, **kwargs):
    """Inicia a sincronizacao do repositorio git annex."""
    createObjectsFromFiles(os.path.join(settings.MEDIA_ROOT, settings.GITANNEX_DIR,
     sender.repositoryURLOrPath))

def createObjectsFromFiles(pathToFiles):
    """Recria os objetos no Django a partir dos objetos serializados em XML."""
    print ">>> DESERIALIZING"
    for root, dirs, files in os.walk(pathToFiles):
        for file in files:
            if file.endswith('.xml'):
                xmlIn = open(os.path.join(root, file), "r")

                for obj in serializers.deserialize("xml", xmlIn):
                    obj.id = None
                    obj.pk = None
                    obj.save()
                os.remove(os.path.join(root, file))
```

A.2.4 mmedia/forms.py

```
form django import forms
```

```
from mmedia.models import MMedia

"""
Arquivo com as definicoes dos forms da aplicacao Django.
"""

class MMediaForm(forms.Form):
    def __init__(self, author, *args, **kwargs):
        super(MMediaForm, self).__init__(*args, **kwargs):
        self.author = author
```

A.2.5 mmedia/management/commands/create objects from files.py

```
from django.core.management.base import BaseCommand, CommandError

from mmedia.signals import createObjectsFromFiles

"""
Definicoes do comando para recriar objetos no Django a partir de objetos serializados em XML.
"""

class Command(BaseCommand):
    """Recria os objetos no Django a partir dos objetos serializados em XML."""
    help = 'Create mmedia objects from serialized objects in given path.'

    def handle(self, *args, **options):
        createObjectsFromFiles(args[0])
```

Referências Bibliográficas

[1] Vinton Cerf, Yogen Dalal, and Carl Sunshine. Specification of internet transmission control program. *IETF RFC Archive*, December 1974. URL http://tools.ietf.org/html/rfc675.

[2] Ernst & Young. Six global trends shaping the business world. *EY Insights*, May 2011. URL http://is.gd/OGR3vi.

[3] Vilém Flusser. *Für eine Philosophie der Fotografie*. European Photography, 1983.

[4] Rede Mocambos. Rede mocambos, uma rede de comunicação social. *Portal da Rede Mocambos*, 2007. URL http://www.mocambos.net/sobre.

[5] Jim Highsmith and Martin Fowler. The agile manifesto. *Software Development Magazine*, 9(8):29–30, 2001.

[6] William Forrest. Clearing the air on cloud computing. *McKinsey*, April 2009.

[7] Marty Alchin. *Pro Django*. Apress, 2009. ISBN 978-1-4302-1047-4.

[8] Gerald Carter. *LDAP System Administration*. O'Reilly, March 2003. ISBN 1-56592-491-6.

[9] Carolina Gutierrez and Lidiane Guedes. *Baobá: comunicação da resistência*. Faculdade de Comunicação da Universidade Metodista de São Paulo, 2009.

[10] AA. VV. *Manual do Usuário do Programa GESAC, Governo Eletrônico Serviço de Atendimento ao Cidadão*. Ministério das Comunicações, 2008. URL http://is.gd/OgcKVt.