INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Radia Store

Armazenamento e Preservação de Programas de Rádio

Daniel António Quaresma Costa

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Júri

Presidente: José Manuel Nunes Salvador Tribolet

Orientador: José Luis Brinquete Borbinha

Vogal: Luis Manuel Antunes Veiga

Setembro de 2008

Agradecimentos

Para o desenvolvimento desta dissertação houve o contributo de pessoas amigas, às quais quero prestar o meu agradecimento.

Os meus agradecimentos ao orientador desta dissertação, o Professor José Luis Brinquete Borbinha, pela orientação, motivação e ajuda que disponibilizou ao longo do trabalho desenvolvido. De seguida, ao Professor Rodrigo Rodrigues pelas orientações iniciais.

Quero agradecer aos colaboradores da Rádio Zero, especialmente à Rute Correia, Ricardo Reis e João Bacalhau pelo esclarecimento dos processos de funcionamento da Rádio e tempo disponibilizado para as reuniões efectuadas.
Um obrigado aos elementos da RUA (Rádio Universitária do Algarve) pela disponibilidade de informação acerca do funcionamento do seu estúdio e recepção na sua rádio.

Agradeço também a todos os meus amigos pelo apoio dado ao longo deste tempo todo.

Por fim, gostaria de agradecer à minha família (em especial pais e irmão) pelo apoio, força e carinho de sempre. É a eles que dedico esta dissertação. Um Bem haja, família.

Um muito Obrigado a todos...

Resumo

Com a constante evolução das tecnologias e dos sistemas de informação surgem necessidades onde este factor pode ser aplicado para a resolução dos mais variados problemas. As vantagens benéficas são imensas e permitem uma automação de processos criando uma mais-valia para quem os utiliza.
As rádios comunitárias são um exemplo dessa necessidade para a qual se pretende encontrar uma solução.

O presente documento tem como intuito analisar os problemas que existem nas rádios comunitárias no que dizem respeito à gestão, armazenamento e preservação dos seus conteúdos digitais. Além disso, é apresentada uma solução para esses problemas permitindo obter um melhor desempenho na execução destes processos criando um impacto positivo neste tipo de organizações.
Para complementar a solução apresentada é desenvolvido um sistema de informação que permite aos autores das rádios comunitárias armazenarem e preservarem os conteúdos digitais que produzem e aos utilizadores dessas rádios adquirirem mais informação sobres esses conteúdos, permitindo uma maior difusão dessa informação.

Palavras-Chave: Rádio Comunitária, Conteúdos Digitais, Sistemas de Informação, Armazenamento e Preservação.

Abstract

With the constant evolution of technology and information systems needs araise where this factor can be applied to the resolution of several problems. There are plenty benefits that allow executing processes in an automated way with value creation to those who use them.
Community radio stations is an example of this need for which is important to find a solution.

This document has the aim to analyze the problems that exist in community radio stations related to management, storage and preservation of their digital contents. Moreover, is presented a solution to these problems allowing a better performance in these processes implementation, creating a positive impact in this type of organization.
To complement the solution presented is developed an information system that allows authors of community radio station store and preserve digital content that they produce and users of these radio stations acquire more information on these contents, allowing a greater dissemination of information.

Keywords: *Community Radio Stations, Digital Contents, Information Systems, Storage and Preservation*

Conceitos

Os conceitos aqui definidos servem para um enquadramento nos capítulos iniciais desta dissertação.

Ficheiro (*Audio Asset*) - um Ficheiro é a entidade que representa o conteúdo digital que é armazenado num **Sistema de Armazenamento**. É constituído por um ficheiro áudio (***Essence***) e pelos respectivos **Metadados**.

Metadados (*Metadata File*) - contém informação acerca dos Ficheiros armazenados. E.g.: Nome do Ficheiro, Autores, Formato.

Emissão (*Broadcast*) - uma Emissão é composta por um ou mais **Ficheiros**. É associada a um **Programa** pelos respectivos **Autores** e ocorre num determinado espaço de tempo.

Programa (*Program*) - entidade que representa um Programa de Rádio. Contém as **Emissões** adicionadas pelos respectivos **Autores de Programas**.

Sistema de Armazenamento (*Replica System*) - um Sistema de Armazenamento, como o próprio nome indica, armazena os Ficheiros adicionados pelos respectivos **Autores**. Para além disso representa também o espaço onde os **Ficheiros** são replicados.

Tags - palavra ou conjunto de palavras associadas pelos actores aos **Programas**, **Emissões** e **Ficheiros**. Permitem estruturar o sistema através da classificação destas entidades e facilitar a pesquisa das mesmas.

Actor Anónimo - é o único tipo de actor que não é autenticado no sistema. As acções sobre o sistema são apenas de consulta.

Actor Registado - privilégio concedido a um Actor Anónimo quando se regista.

Autor de Programas - um Actor Registado é Autor de Programas quando pode criar Emissões para um determinado Programa.

Editor - o Editor é quem gere os Programas existentes no sistema e todo o conteúdo relacionado com este. Tem também a possibilidade de gerir os privilégios (Autor e Editor) dos actores do sistema.

Administrador - um Administrador tem como função configurar e gerir o sistema para que funcione correctamente.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ACL	*Access Control List*
API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
DNS	*Domain Name System*
FTP	*File Transfer Protocol*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
MPEG	*Movie Picture Experts Group*
MP3	*MPEG-1 Audio Layer 3*
M3U	*MP3 Uniform Resource Locator*
PC	*Personal Computer*
REST	*Representational State Transfer*
RF	Requisito Funcional
RNF	Requisito Não Funcional
RSS	*Really Simple Syndication*
SQL	*Structured Query Language*
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*

1 Introdução

1.1 Contexto

> "*A community radio station is one that is operated in the community, for the community, about the community and by the community.*" [12]

O conceito de rádio comunitária depende de vários factores, como por exemplo, da área geográfica onde a rádio se insere ou dos factores culturais da comunidade da rádio. No entanto, o que distingue as rádios comunitárias de outros meios de comunicação é a elevada participação das pessoas na produção e gestão de conteúdos digitais da rádio.
Uma rádio comunitária opera para uma comunidade onde as pessoas que a compõem têm algo em comum. Pode ser a região onde opera (e.g. uma comunidade universitária, ou uma vila), ou os interesses que as pessoas ligadas à radio partilham entre si (e.g. grupos étnicos, grupos etários ou interesses musicais).
As Emissões que passam neste tipo de rádios são de variados formatos (e.g. entrevistas, documentários, música) e é da competência de cada Autor definir o que cada Emissão contém produzindo os respectivos Ficheiros. Isto faz com que um maior número de pessoas adira à comunidade e exponha as suas ideias, originando uma maior troca de informação e uma maior disseminação da mesma.

A Rádio Zero[1] é um exemplo de uma rádio comunitária onde os seus colaboradores produzem Ficheiros para as Emissões dos Programas de Autor. Esta rádio, que pertence ao Instituto Superior Técnico, situa-se num dos *campus* desta faculdade (campus da Alameda). Actualmente, as suas Emissões são emitidas via *stream* na Internet para o *campus* da Alameda desta faculdade. A Rádio Zero será utilizada como caso de estudo para uma melhor percepção do problema em análise que existe nas rádio comunitárias.

1.2 Objectivos

Esta dissertação tem como objectivo principal modelar o problema de gestão, armazenamento e preservação dos Ficheiros dos Programas de Autor produzidos em rádios comunitárias. A modelação do problema resultará num sistema onde os Autores podem gerir as Emissões e os respectivos Ficheiros e que funcionará como sistema de armazenamento principal de uma rádio comunitária.
Deste modo, os objectivos a atingir são os seguintes:

- Estudar e identificar as principais características dos sistemas de armazenamento de conteúdos digitais actuais;
- Possibilitar o armazenamento dos Ficheiros criados pelos Autores dos Programas de Rádio, possibilitando a associação dos respectivos Metadados;
- Fornecer informação acerca dos Ficheiros, Programas e Emissões associados à rádio;
- Preservar os Ficheiros criados pelos Autores aproveitando as funcionalidades dos sistemas de armazenamento estudados para replicar os Ficheiros criados pelos Autores;

[1] http://www.radiozero.pt

- Permitir a gestão dos Ficheiros replicados. Após a replicação dos Ficheiros o sistema gere o estado dos mesmos;
- Adopção de conceitos relacionados com a Internet e com comunidades para impulsionar a interacção entre os actores da rádio. Os principais conceitos são o processo de *Tagging* (associar Tags a Ficheiros, Programas e Emissões) e a associação de comentários a Ficheiros;
- Disponibilizar ferramentas de pesquisa de Ficheiros, Programas e Emissões. Estas entidades podem ser pesquisadas através das Tags previamente associadas, ou através de palavras-chave livres (e.g. nome de um Programa, Género de uma Emissão);
- Funcionar como serviço Web, usando o padrão arquitectural REST, para disponibilizar a informação mais relevante existente no sistema (e.g. informação sobre Ficheiros, Programas e Emissões).

1.3 Estrutura da Dissertação

No **Capítulo 2** é apresentado o Estado da Arte que consiste num estudo intensivo de sistemas de armazenamento e preservação de conteúdos multimédia, apresentando as suas principais características e funcionalidades. Este capítulo serve para ter a percepção do estado actual da área onde se enquadra o problema apresentado.

O **Capítulo 3** descreve em detalhe a análise do problema abordado e que se pretende resolver. É utilizada a rádio do Instituto Superior Técnico, a Rádio Zero, como caso de estudo e que permite uma análise mais completa do problema existente neste tipo de rádios.

No **Capítulo 4** é apresentada a Proposta da Solução para o problema descrito no capítulo anterior.

No **Capítulo 5** é descrita a Implementação de um sistema de informação baseada na Proposta da Solução.

No **Capítulo 6** são feitas as conclusões acerca do trabalho desenvolvido e apresentadas oportunidades para trabalho futuro.

No final são apresentados os Apêndices que complementam todo o trabalho realizado ao longo dos capítulos anteriores.
No **Apêndice A** são apresentados os Casos de Uso que suportam os requisitos que surgiram da Análise do Problema.
Nos **Apêndices B e C** são apresentados os Modelos de Domínio e o Modelo de Classes, respectivamente.

2 Estado da Arte

Neste capítulo serão descritos sistemas de armazenamento de conteúdos digitais, mais propriamente, conteúdos multimédia, com alguma relevância para o problema apresentado. Foi feito um estudo intensivo sobre estes sistemas, tendo principal realce para o problema desta dissertação os quatro primeiros apresentados. Os restantes sistemas são soluções emergentes potencialmente relevantes para o mesmo estudo. Deste modo, são descritas as principais características e funcionalidades dos sistemas para fornecer um contexto sobre a área onde esta dissertação se enquadra.
Cada um dos sistemas será classificado como **cooperativo** no caso de ser um sistema de código aberto e não estar agregado a nenhuma organização tecnológica. Caso contrário, será classificado como **institucional**.

2.1 Sistema *StreamOnTheFly*

Classificação: Cooperativo.
O *StreamOnTheFly*[1] é um sistema que permite criar uma rede cujo objectivo é reutilizar programas de rádio já emitidos. É necessário que estes sejam partilhados por todos para uma possível reutilização.

2.1.1 Arquitectura

A arquitectura deste sistema é composta por três partes:

- ***Node Servers* (NS):** rede distribuída de servidores que contém os programas de rádios com os respectivos metadados, estatísticas de acessos e histórico de utilização
- **Estação de Rádio:** também designada por estação e controlo, é uma Aplicação Web que tem como objectivo fazer o *broadcast* dos programas de rádio e também gerir os programas de rádio nos NS (ou seja, armazená-los e publicá-los)
- **Portal:** *website* personalizado pelos utilizadores que fornecem conteúdo aos NS

Para o estudo em questão, o NS é a componente desta arquitectura com mais interesse. Cada estação de rádio está associada a um NS e cada NS pode suportar várias estações de rádio. Os portais gerados a partir do conteúdo dos NS são completamente independentes das estações de rádio que emitem e geram esses mesmos conteúdos.

2.1.2 *Node Servers*

Cada NS tem associado a si um conjunto de vizinhos (que também são *node servers*) com os quais troca informações para se manterem actualizados. Deste modo, é feita a autenticação dos vizinhos perante cada NS. Um NS só se pode conectar a outro NS se ambos forem vizinhos um do outro. Caso contrário, não é possível efectuar qualquer tipo de operação.
Quando alguém armazena um programa num NS, apenas os metadados associados a esse programa são replicados pelos vizinhos desse NS.

[1] http://www.streamonthefly.org/

Apenas os metadados que são necessários para obter o que se pretende nas pesquisas são replicados. Portanto, qualquer outro tipo de conteúdo (e.g. ficheiros de áudio ou de outro formato) é armazenado no NS que suporta a estação de rádio.

As actualizações entre os vizinhos de cada NS são feitas com base em mensagens de sincronização. Cada NS deve enviar periodicamente uma mensagem com o seu estado actual para cada vizinho. Em resposta a esta mensagem, o NS recebe as alterações de cada vizinho. A marca temporal (*timestamp*) destas mensagens é tida em consideração para validar a mensagem. Se esta marca temporal for válida (ou seja, se não for uma mensagem antiga) a mensagem é aceite. Caso contrário, a mensagem é simplesmente ignorada.
Com a troca destas mensagens - as alterações que cada NS sofre - a replicação pode tornar-se um pouco exaustiva para o sistema.
Para um novo NS entrar na rede, apenas tem de notificar um NS já presente na rede, que fica automaticamente como o seu primeiro vizinho. Quando é enviada a primeira mensagem de sincronização, a rede de vizinhos é formada pelo novo NS. Quando um NS está em baixo, um dos seus vizinhos ao enviar a mensagem de sincronização há-de reparar que está em baixo, e essa informação será passada juntamente com a informação de actualização do NS que detectou a falha para o resto da rede. Entretanto se o nó recuperar da falha, o processo de adaptação será similar ao de entrar na rede como sendo um novo NS.

Cada NS tem de cumprir três objectivos principais para funcionar correctamente na rede em que está inserido:

1. Manter actualizada a informação dos NS disponíveis, assim como as estações de rádio
2. Manter os metadados sincronizados
3. Controlar a respectiva rede no qual está inserido

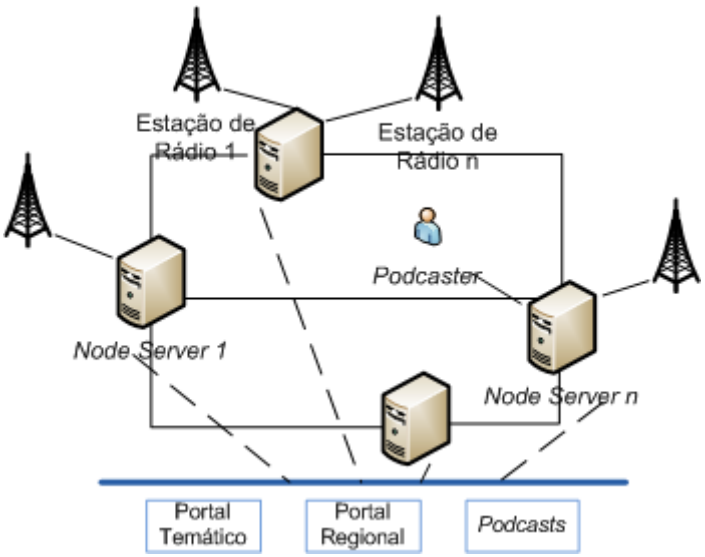

Figura 2.1: Arquitectura *StreamOnTheFly*

2.1.3 Metadados

Os criadores deste sistema criaram um formato para os metadados dos ficheiros gerados, o XBMF (*Exchange Broadcast Binary And Metadata Format*) [4], baseado no formato standard para este tipo de recursos, o *Dublin Core*[2], do qual utiliza os seus elementos para modelar dados. Para além de conter informação sobre o ficheiro que se pretende armazenar ou transferir, pode-se também especificar dependências desse mesmo ficheiro (como por exemplo, uma imagem ou texto).
Cada NS implementa também um *data provider* segundo o protocolo OAI-PMH[3] o que permite fornecer os dados de cada NS para outros sistemas externos. Para além disso é possível criar *podcasts* e disponibilizá-los através de *feeds* RSS.

2.2 Projecto *Campcaster*

Classificação: Cooperativo.
O *Campcaster*[4] é uma aplicação de código aberto da iniciativa *Campware* para gestão de rádios, usada em algumas rádios europeias, africanas e da América do Sul. Esta iniciativa, suportada pelo *Media Development Loan Fund*, tem como objectivo desenvolver software livre para organizações independentes ligadas à multimédia.
O *Campcaster* não é muito utilizado em rádios cuja transmissão também é feita por internet (embora possa existir *streaming* pelo mesmo meio) mas sim por rádios convencionais.

Os ficheiros de áudio (como por exemplo um ficheiro de música, ou uma reportagem) são armazenados no servidor do *Campcaster* que é acedido pelos utilizadores que utilizam o sistema. Ao serem armazenados é solicitada a inserção de metadados, relacionados com esse mesmo ficheiro, segundo o formato ID3[5]. Existe um grande conjunto de metadados que o utilizador pode inserir consoante o tipo de ficheiro que é. Por exemplo, se for um ficheiro de voz (como uma reportagem) é possível inserir informação tal como a data/hora da reportagem e localização onde foi efectuada.
É feita um aposta bastante grande na inserção de metadados nos ficheiros a armazenar, pois quanto maior a quantidade e qualidade de metadados inseridos, melhor será a qualidade e eficiência das pesquisas que se efectuam ao sistema de armazenamento.

Existe um conceito nesta aplicação que se designa de *Network Hub* (Figura 2.2). O conceito baseia-se num servidor onde uma comunidade de estações de rádio se liga e partilha conteúdos em forma de ficheiros áudio ou em forma de *playlists* criadas pelos utilizadores da rádio. O interessante em partilhar uma *playlist* é que todo o conteúdo associado também é partilhado. Os formatos utilizados para estas playslists são:

- SMIL[6] (*Synchronized Multimedia Integration Language*) que é mais utilizado para partilha entre es-

[2] http://dublincore.org/documents/dcmi-terms
[3] http://www.openarchives.org/pmh/
[4] http://www.campware.org/en/camp/campcaster_news
[5] http://www.id3.org
[6] http://www.w3.org/AudioVideo

tações de rádio da aplicação *Campcaster*

- M3U, que é um formato bastante comum e conhecido para outros leitores de ficheiros MP3
- XSPF (XML *Shareable Playlist Format*)[7], um formato que permite portabilidade das playlists entre vários dispositivos

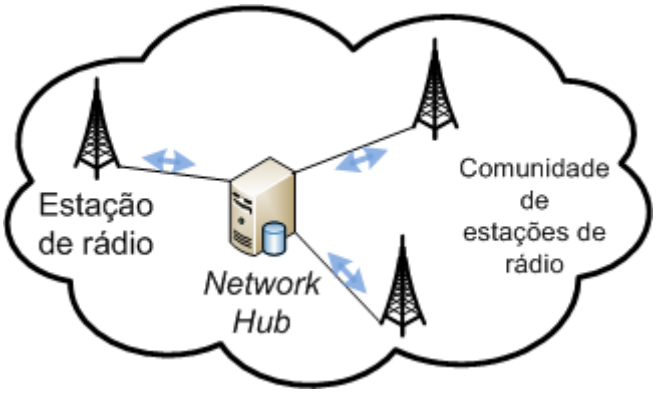

Figura 2.2: *Network Hub*

Quando alguém pretende utilizar os ficheiros que se encontram armazenados no servidor para construir uma *playlist*, por exemplo, tem várias maneiras de o fazer através da biblioteca multimédia existente:

- **Pesquisa:** este método de procura funciona como um motor de busca. A informação a procurar é aquela que foi inserida previamente, quando foi solicitada a inserção de metadados. Neste método pode-se definir também alguns critérios de pesquisa, como por exemplo, adicionar mais termos para pesquisar ou filtrar a pesquisa por tipo de ficheiro
- **Navegação:** neste método para obter um ficheiro pode-se refinar mais os critérios escolhidos. Esses critérios também são baseados nos metadados inseridos previamente. O utilizador pode filtrar os resultados segundo três opções com categorias existentes nos metadados associados aos ficheiros
- **Pesquisa directa na biblioteca:** esta pesquisa é a mais simples que está disponível ao utilizador. Este pode pesquisar o ficheiro que quer inserindo texto livre associado aos metadados dos ficheiros que se encontram armazenados
- **Pesquisa no *Hub*:** este método apenas é possível quando a estação de rádio está ligada a um *Network Hub* referido anteriormente

O arquivo desta aplicação contém ainda uma outra funcionalidade importante para este tipo de sistemas de armazenamento. Esta funcionalidade é a gravação de um registo áudio com tudo o que passa em *broadcast*. Em alguns países, esta informação deve estar acessível quando solicitada por entidades governamentais. Na Inglaterra, por exemplo, as estações de rádio são obrigadas a guardarem estes registos por quarenta e dois dias depois a emissão dos programas de rádio. Esta aplicação utiliza um sistema implementado pela rádio da Universidade de Warwick, na Inglaterra, que se designa de *Darklog*[8] com as seguintes características:

- É livre, de código aberto

[7] http://xspf.org
[8] http://radio.warwick.ac.uk/software/logging.htm

- Os registos áudio são gravados em vários ficheiros que depois podem ser recuperados mais tarde, com base em certos critérios de procura
- Apenas tem limitações a nível de espaço, ou seja, pode estar a gravar estes registos constantemente desde que tenha suporte físico para tal
- Suporta formatos áudio comuns (e.g. MP3)
- Interfaces para recuperar registos já armazenados

2.3 *Open Metadata Archive*

Classificação: Cooperativo.
O *Open Metadata Archive* é um sistema que permite armazenar os metadados de conteúdos multimédia dos mais variados tipos. Baseia-se num princípio em que não é preciso saber tudo, apenas conhecer a referência para o que se quer saber[9]. Este sistema armazena qualquer tipo de metadados - de conteúdo multimédia - mas os requisitos de indexação para estes diferentes conteúdos é diferente, o que pode levar à existência de diferentes sistemas para o fazer. O OMA tem como objectivo ligar estes possíveis sistemas a um nível de metadados.

A categorização dos conteúdos a armazenar é baseada numa estrutura de árvores. Esta é organizada segundo categorias associadas aos conteúdos multimédia. Para isso, é preciso definir estas categorias, do ponto de vista dos criadores dos mesmos, o que pode ser uma tarefa difícil de conseguir. Feito isto, a organização é feita de uma forma bastante flexível e para recuperar os conteúdos multimédia pretendidos, basta seguir a estrutura da árvore consoante a classificação dada previamente.

Um exemplo de utilização do OMA foi a *reboot.fm*[10], baseada em Berlim, que cessou o *broadcast* em 2004. Esta rádio usava os metadados dos programas, que os utilizadores inseriam, para criar a estrutura da árvore que dava suporte aos conteúdos multimédia. As folhas desta árvore eram, por exemplo, os programas emitidos em *broadcast*. Esta rádio, para exportar informação sobre os seus conteúdos, utilizava *XML* para obter informação sobre os seus conteúdos multimédia. Um dos formatos possíveis de exportação era o *Dublin Core*.

2.4 *NewsTuner*

Classificação: Institucional.
O *NewsTuner* [8] é uma aplicação Web que permite a procura e pesquisa de grandes arquivos de rádio. A diferença desta aplicação para as aplicações convencionais de procura é que os resultados da procura combinam uma procura semântica e acústica nos ficheiros armazenados, feita de forma eficiente.

Para encontrar um ficheiro áudio ou um conjunto de ficheiros, existem duas formas de o fazer:

[9] http://oma.sourceforge.net
[10] http://reboot.fm:9000

- Procura por palavras-chave: permite ao utilizador procurar o que pretende por palavras-chave nos textos presentes nos ficheiros e nos metadados associados ao mesmo
- Pesquisa por similaridade: é retornada uma lista de ficheiros áudio semelhantes em conteúdo ao ficheiro áudio previamente escolhido por uma destas duas formas de procura

A procura por palavras-chave é útil nos casos onde o utilizador pretende encontrar algo em específico enquanto que a pesquisa por similaridade é usada para pesquisar ficheiros áudio sem um objectivo em particular.

A indexação e gestão dos ficheiros são feitas aparte da aplicação de pesquisa e consumo de ficheiros de áudio. Esta é a parte mais importante desta aplicação. Inicialmente a indexação era feita segundo treze categorias e quinze subcategorias em cada uma das principais. O objectivo era classificar os ficheiros segundo categorias fixas para facilitar a interacção dos utilizadores na procura de conteúdos. No entanto, provou-se que esta classificação fixa era ineficiente:

- Por vezes existiam ficheiros cobrindo múltiplos assuntos e não podiam ser classificados com múltiplas classificações
- Os produtores nem sempre estavam de acordo com a classificação a dar a cada ficheiro áudio criado
- Os utilizadores do *NewsTuner* nem sempre encontravam o que queriam com a classificação fixa que existia inicialmente
- A hierarquia fixa de classificação não era flexível o suficiente para responder a casos de excepção, como por exemplo a actualização de conteúdos multimédia já existentes

Com todos estes problemas surgira então dois tipos de indexação. A indexação acústica e a indexação semântica de ficheiros áudio.

2.4.1 Indexação Acústica

A indexação acústica consiste em gerar automaticamente os textos presentes nos ficheiros áudio usando sistema de reconhecimento de voz com um vocabulário de grandes dimensões. Depois do reconhecimento efectuado, são usados algoritmos de recuperação de informação para indexar os respectivos textos.
Esta parece uma óptima opção de indexação, não requerendo grande intervenção humana, a não ser a gestão do vocabulário. No entanto é um pouco imprecisa quando surgem problemas como a qualidade dos ficheiros de áudio, o não reconhecimento de termos em queries e enganos em reconhecimento de termos.

2.4.2 Indexação Semântica

A indexação semântica permite o utilizador pesquisar conteúdos por tópicos. Esta indexação é feita de forma automática para eliminar os problemas acima descritos. Para isso é utilizado um modelo designado de *Probabilistic Latent Semantic Analysis* (PLSA). Este modelo consiste em transformar os documentos, que são representados num espaço de conjunto de palavras em cada uma delas representa uma dimensão, em espaços de tópicos (as possíveis classificações para indexação) muito mais pequenos. Para gerar estes

espaços de classificação são utilizados os metadados (no caso de existirem) ou então é utilizado o texto presente nos ficheiros áudio.

2.5 *Amazon Simple Storage Service* (Amazon S3)

Classificação: Institucional.
O *Amazon S3*[11] é um serviço disponibilizado pelos serviços Web da *Amazon* (companhia de comércio electrónico), que permite aos utilizadores armazenar ficheiros (ao qual passaremos a chamar de objectos, por uma questão de terminologia) para disponibilizar na Internet. Este serviço disponibiliza APIs para os programadores, o que facilita bastante a implementação de qualquer sistema baseado neste serviço, permitindo armazenar e recuperar qualquer tipo de objecto de uma forma eficaz.

O *Amazon S3* é bastante recente (surgiu em 2006), e implica um registo nos serviços Web da Amazon para que se possa utilizar. As principais funcionalidades são:

- O número de objectos que se pretende guardar é ilimitado. Estes só podem ter um tamanho máximo de 5 *gigabytes*
- O acesso aos objectos é feito por uma identificação única. Uma chave que lhe é associada no momento em que se armazena pela primeira vez
- Possuí métodos de autenticação para melhor segurança
- Disponibiliza APIs, para desenvolvimento, para o padrão arquitectural REST e protocolo SOAP
- A transferência de objectos pode ser feita por HTTP ou por *BitTorrent*

A ideia do *Amazon S3* é dar a possibilidade da não preocupação com o armazenamento, e as dificuldades de manutenção associadas a este, a quem cria sistemas de informação. Para tal, este serviço, tem as seguintes características:

- **Escalabilidade:** como o número de objectos a guardar pode ser infinito, este serviço, torna-se bastante escalável em termos de armazenamento e tempo de resposta para recuperação desses mesmos ficheiros
- **Fiabilidade:** como se pretende fornecer um serviço em que o utilizador não tem de se preocupar com problemas de armazenamento, então o tempo de indisponibilidade do serviço tem de ser aproximadamente nulo. Para além disso não se podem tolerar faltas para que a disponibilidade seja máxima
- **Rapidez:** a latência de servir os pedidos tem de ser baixa. O servidor tem de ser eficaz a atender esses mesmos pedidos
- **Barato:** tendo em conta o custo que se tem de suportar em manter um sistema com todos estes princípios, o valor a pagar pela oferta do serviço é mínimo
- **Simples:** construir um sistema com todas estas características é difícil. Com os métodos de utilização disponíveis do *Amazon S3* torna-se muito mais fácil de construir bons sistemas de informação com o mínimo de credibilidade

[11] http://aws.amazon.com/s3

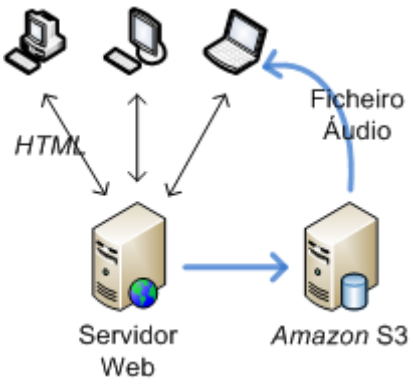

Figura 2.3: *Generalização de um sistema usando o* Amazon S3

O *Amazon S3* foi desenhado para armazenar objectos. Cada objecto é guardado dentro de um ***Bucket*** e é-lhe associada uma ***Key***. Para além disso, um objecto possui os seus metadados, o seu ***Controlo de Acesso*** e método de ***Autenticação*** [10].

2.5.1 *Buckets*

Os *Buckets* são os "contentores"dos objectos. Quando se pretende armazenar um objecto, este é associado a um *Bucket* que o contém. Para isso, é necessário que os nomes dos *Buckets* sejam únicos. É como se tratassem de domínios de internet, uma vez que o nome de um *Bucket* tem mapeamento directo com os subdomínios dos serviços Web disponibilizados pela Amazon. Os nomes correspondem a uma conta dos Amazon Web Services e consequentemente do *Amazon S3*.
Uma particularidade deste sistema, é que os *Buckets* não podem conter outros *Buckets*.

2.5.2 *Keys*

Uma *Key* é o atributo dos objectos que os identifica de forma única dentro de um *Bucket*. Esta permite a recuperação dos objectos quando armazenados.

2.5.3 Metadados

Cada objecto armazenado num *Bucket* tem associado a si metadados. O tamanho máximo permitido de metadados neste sistema é de apenas 2 *Kilobytes*. No entanto existem dois tipos de metadados para cada objecto:

- Metadados de sistema: estes são utilizados internamente pelo *Amazon S3* e difere consoante a API que o programador usa
- Metadados do utilizador: estes são introduzidos pelo utilizador e o *Amazon S3* apenas os armazena.

2.5.4 Controlo de Acesso e Autenticação

A Autenticação é o processo de identificar um utilizador ou serviço que tenta aceder ao *Amazon S3*. Por outro lado o Controlo de Acesso controla quem é que pode fazer o que sobre os objectos existentes (ler,

escrever, entre outras acções).
Através destes dois conceitos é definida uma política de acesso que permite os utilizadores de efectuarem operações, ou não, sobre os objectos ao *Amazon S3*.

Quando um utilizador adquire um registo nos serviços Web disponibilizados pela *Amazon*, é-lhe atribuída uma chave de acesso e uma chave secreta. Estas duas chaves autenticam qualquer pedido do utilizador que as possuí perante esses serviços. Quando um utilizador necessita de efectuar um pedido, o processo é o seguinte:

1. Através do pedido mais a chave secreta, é gerada uma assinatura desse mesmo pedido
2. O pedido normal, juntamente com a assinatura, é enviado para o serviço pretendido (por exemplo, o *Amazon S3*)
3. Quando o pedido, mais a assinatura, chegam ao serviço, é feito o mesmo processo com o pedido original e com a chave secreta obtida pela base de dados da *Amazon* para o utilizador que efectuou o pedido
4. Se a assinatura gerada coincidir com a que o utilizador enviou, então o acesso o pedido é autenticado, caso contrário, a autenticação falha

Figura 2.4: Método de Autenticação do *Amazon S3*

Para controlar o acesso sobre os objectos são definidas ACLs. Uma ACL é definida na criação de um Bucket ou na escrita de um objecto. No caso de não ser definida uma ACL no momento da criação de um Bucket ou objecto, é atribuída uma ACL por omissão, onde o criador fica com acesso total sobre o recurso. Existem cinco maneiras diferentes de conceder permissões aos utilizadores que queiram aceder a este serviço:

1. **Dono/Criador:** esta concessão é dado ao utilizador que cria um *Bucket* ou objecto. O criador destes, não pode ser alterado, embora quando um objecto é modificado por outro utilizador, esse objecto adquire outro criador
2. **Utilizador por e-mail:** o método de identificação no *Amazon S3* é por e-mail. Deste modo, podemos conceder permissões aos utilizadores que estejam registados nos serviços Web disponibilizados pela *Amazon*. No entanto, este tipo de acesso só é possível se o e-mail do utilizador estiver associado apenas a uma conta dos serviços Web da *Amazon*. Caso contrário, é gerado um erro e têm de ser efectuadas medidas para a resolução do problema, atribuindo outro tipo de concessão de acesso, por exemplo

3. **Utilizador por representação canónica:** este é outro método para garantir acesso a utilizadores com uma conta nos serviços Web da *Amazon*. Através da chave que o identifica e da respectiva chave secreta, o utilizador pode aceder aos objectos que tem acesso. A concessão de permissões são é dada pela chave que o identifica
4. **Utilizadores registados nos serviços Web da *Amazon*:** este tipo de acesso permite que todos os utilizadores com uma conta nos serviços Web da *Amazon* possam aceder aos *Buckets* e objectos de um determinado utilizador (aquele que especificou esta concessão)
5. **Todos os Utilizadores:** qualquer utilizador pode aceder ao *Bucket* ou objecto que tenha este tipo de acesso

2.6 *Amazon SimpleDB*

Classificação: Institucional.
O *Amazon SimpleDB*[12] é mais um serviço disponibilizado pelos serviços Web da *Amazon*. Permite o armazenamento de dados de forma simples e estruturada sem possuir qualquer tipo de esquema (comparando aos modelos de base de dados relacionais).

A grande diferença para o *Amazon S3* é que este serviço não permite o armazenamento de grandes quantidades de dados (um exemplo disso é que o valor dos atributos de cada item não poder ultrapassar os 1024 *bytes*). No entanto, se conjugarmos este serviço com o *Amazon S3* ou outros serviços Web da *Amazon*, podemos criar aplicações Web, escaláveis e com grande potencial.

Os dados neste serviço têm uma estrutura organizada em items onde cada item é descrito em pares de atributo-valor representando os dados em si. O seu tipo é sempre do tipo *string* sendo a comparação feita sempre de forma lexicográfica. O inconveniente é que a comparação de certos tipos de dados, como datas e números, seja mais difícil de efectuar. O armazenamento destes items de dados é organizado em domínios onde são efectuadas *queries* sobre os dados de cada um dos domínios. No entanto, não existe a possibilidade de efectuar queries envolvendo mais do que um domínio

A analogia da estrutura deste serviço é uma folha de cálculo. Cada domínio pode ser comparado a uma folha de cálculo, onde cada linha representa um item de dados. Os cabeçalhos das colunas podem ser associados aos atributos de cada item, e as células internas os valores respectivos.

2.7 *Hadoop*

Classificação: Cooperativo.
O *Hadoop*[13] é uma plataforma de software, que permite a escrita e leitura de ficheiros (de grande tamanho) usando um sistema de ficheiros próprio (HDFS - *Hadoop Distributed File System*) que replica os dados por

[12] http://aws.amazon.com/simpledb
[13] http://hadoop.apache.org

clusters formados por esta plataforma.
O *Hadoop* é parte de um software de pesquisa na Web, designado de *Nutch*. As principais características do *Hadoop* são:

- **Escalabilidade:** pode armazenar grandes quantidades de dados
- **Económico:** distribui os dados e processa-os sobre os clusters, que podem correr em computadores normais, ou seja, que não necessitem de grandes recursos
- **Eficiência:** como os dados são distribuídos pelos diferentes nós dos *clusters*, o processamento destes dados pode ser feito em paralelo
- **Fiabilidade:** existem várias cópias dos dados e quando existem falhas, o *Hadoop* consegue recuperar delas facilmente

A arquitectura do sistema de ficheiros do Hadoop é bastante parecida com o sistema de ficheiros do *Google* (*GoogleFS*) [9].
É baseado na arquitectura *master/slave*, onde existe um servidor *master* (designado de *Namenode*) que gere o espaço de nomes do sistema de ficheiros e regula o acesso aos ficheiros do mesmo. Para além disso existem vários nós (designados de *Datanodes*) que gerem os dados que são armazenados nos *clusters*.
Os metadados associados a todo o sistema são guardados pelo *Namenode* que tem como função fazer o mapeamento dos ficheiros para o *datanode* (ou *datanodes*) correspondentes.

O acesso aos ficheiros é efectuado por uma API, em Java, disponibilizada pelo HDFS. Além disso, está em desenvolvimento o acesso via o protocolo WebDAV.

2.8 Sistema LOCKSS (*Lots of Copies Keep Stuff Safe*)

Classificação: Cooperativo.
O sistema LOCKSS[14] é uma ferramenta utilizada para preservar conteúdos publicados na Internet, desenvolvida pela Universidade de *Stanford*. É essencialmente utilizada por bibliotecários e para preservar conteúdos governamentais, o que permite preservar a médio/longo termo o conteúdo de informação sensível e recuperá-la quando solicitada [7].
O sistema LOCKSS parte do princípio que toda a informação a armazenar tem de ser preservada por um tempo considerável, mais do que os sistemas tradicionais que conhecemos. Por isso, quando um ficheiro é armazenado são feitas réplicas pelos outros elementos da rede LOCKSS para assegurar isso. Além disso, esses conteúdos são constantemente auditados e reparados (quando estão danificados), com baixos custos associados.

Este sistema é livre e descentralizado funcionando em *peer-to-peer*.
Cada nó (i.e., cada *peer* da rede, designados também por LOCKSS *Box*) do LOCKSS tem três funcionalidades básicas:

[14] http://www.lockss.org

- Coleccionar os conteúdos usando um Web *crawler*[15] semelhante aos que os motores de busca utilizam
- Distribuir os conteúdos que coleccionou *apriori*, pelos outros nós da rede LOCKSS
- Preservar os conteúdos usando métodos de detecção e recuperação de erros

2.8.1 Coleccionar conteúdos

Antes de um dos nós começar a coleccionar conteúdos, a pessoa que pretende que o LOCKSS coleccione os seus conteúdos, tem de lhe dar permissão para tal. Além disso é preciso especificar alguns parâmetros, como por exemplo o número máximo da profundidade da procura para não cair no erro de recuperar informação desnecessária.

2.8.2 Distribuir conteúdos

Os vários nós LOCKSS utilizam a internet para comunicarem entre eles e para trocarem conteúdos para a sua replicação. Quando é solicitado um conteúdo específico, é efectuada uma votação entre os nós para verem o conteúdo em comum a ser fornecido.

2.8.3 Preservar conteúdos

Quando se nota que um nó LOCKSS tem conteúdo danificado tomam-se providências para reparar esse conteúdo baseado no que os outros nós possuem. A vantagem desta solução, é que é desnecessário efectuar cópias de segurança individuais a cada nó. Portanto, quanto mais réplicas o sistema tiver maior é a fiabilidade do mesmo.

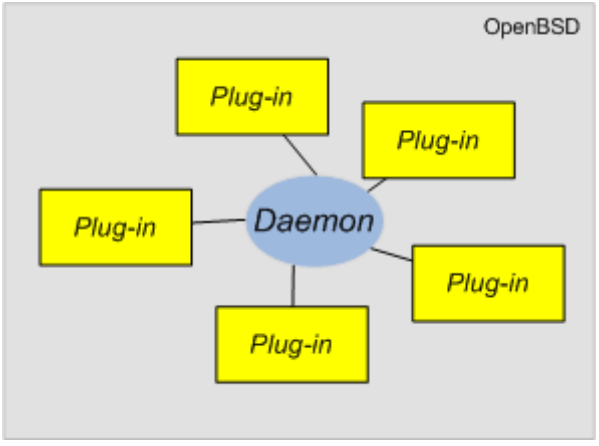

Figura 2.5: Arquitectura do sistema LOCKSS

2.8.4 Arquitectura

Existem dois componentes básicos na arquitectura de um nó LOCKSS: um *daemon* e vários *plugins*.
Um *daemon* é uma aplicação que proporciona as três funcionalidades descritas (colecção, distribuição e

[15] Web *crawler* - programa que percorre a Web de uma forma automatizada, para adquirir conteúdos, para posterior indexação e processamento de informação

preservação de conteúdos).

Por outro lado, os *plugins* são pequenos programas que adaptam as funcionalidades do *daemon* a um publicador de conteúdos em particular. Este sistema tem um bom desempenho em armazenar conteúdos com bastante fiabilidade e a baixo custo. No entanto, os Web *crawler* utilizados podem recuperar informação desnecessária. Seria uma boa aposta, para quem quisesse publicar conteúdos, inseri-los automaticamente num dos nós do sistema LOCKSS e a partir daí o sistema operar conforme o descrito.

2.9 Tabelas Comparativas

As Tabelas 2.1 e 2.2 apresentam respectivamente as vantagens e desvantagens de cada sistema descrito permitindo uma comparação entre os mesmos. As descrições apresentadas pretendem focar conceitos relacionados com a área em estudo, como por exemplo, a fiabilidade, extensibilidade, escalabilidade e custos de manutenção associados.

Tabela 2.1: Tabela comparativa com as vantagens do sistemas apresentados

StreamOnTheFly	- Boa replicação dos metadados associados aos dados de cada nó; - Adição/remoção de nós do sistema de forma transparente o que o torna bastante extensível; - Escalabilidade; - Formatos normalizados.
Campcaster	- Bons formatos para disponibilizar *playlists* com boa portabilidade; - Boa recuperação de informação.
Open Metadata Archive	- Possibilidade de incorporar vários sistemas o que permite extensibilidade; - Boa classificação de conteúdos através dos metadados.
NewsTuner	- Boa classificação de conteúdos; - Indexação e armazenamento de conteúdos.
Amazon S3	- Sistema rápido, escalável e simples de usar; - Baixos custos de manutenção de uma aplicação que implemente este sistema.
Amazon SimpleDB	- Armazenamento flexível e simples; - Bom para apoiar outros sistemas; - Análogo a outras tecnologias já existentes o que facilita o seu uso.
Hadoop	- Processamento em paralelo dos dados devido à replicação; - Boa replicação; - Sistema escalável.
LOCKSS	- Boa replicação; - Auditoria e reparação de ficheiros nos nós do sistema; - Arquitectura *loose coupling*.

Tabela 2.2: Tabela comparativa com as desvantagens do sistemas apresentados

StreamOnTheFly	- Fracas interfaces gráficas para os utilizadores; - Se um dos nós falha, as rádios associadas ficam sem servidor para armazenar conteúdos; - Replicação de conteúdos inexistente; - Recuperação de informação.
Campcaster	- Poucos formatos normalizados para disponibilizar conteúdos; - Interfaces para inserção de informação complexas; - Custos de manutenção.
Open Metadata Archive	- Apenas permite armazenar e manipular metadados.
NewsTuner	- Para a indexação acústica, é necessário um bom vocabulário por idioma; - Se a indexação não for bem efectuada, pelos métodos que apresenta, gera pesquisas incoerentes e de fraca qualidade.
Amazon S3	- Métodos de replicação desconhecidos; - Controlos de acesso aos dados; - Necessita registo na *Amazon*.
Amazon SimpleDB	- Restrição de armazenamento; - Restrição na manipulação de dados; - Necessita registo na *Amazon*.
Hadoop	- Sistema dependente de um componente *master* para controlo de todo o sistema.
LOCKSS	- Permissão para recolha de conteúdos; - Sistema fechado à comunidade onde efectua as recolhas de conteúdos.

3 Análise do Problema

A programação que passa nas rádios comunitárias é constituída principalmente por Programas de Autor, onde os seus actores são os Autores basicamente de todo o conteúdo que é emitido: os Ficheiros. Estes conteúdos - que proporcionam cultura, informação e ideias - devem ser armazenados e disponibilizados para a comunidade segundo uma gestão própria feita pelos mesmos actores que os produzem.
Como o âmbito do problema se enquadra nas rádios comunitárias são poucos os Ficheiros disponibilizados on-line, vindo-se a notar um maior crescimento na sua partilha ao longo do tempo. Apenas são disponibilizados alguns *podcasts* e alguns Ficheiros de todo o conteúdo que é produzido (e.g. Ficheiros das Emissões de determinado Programa). Uma das principais causas deste problema é a falta de verbas para suportar os custos associados à gestão de um Sistema de Informação que consiga suportar os requisitos mínimos exigidos, para que se possa partilhar todo o conteúdo que é gerado nestas rádios.

Disponibilizar os Ficheiros gerados por uma rádio comunitária permite alargar a sua comunidade e permite de igual modo aumentar a disponibilidade de informação que gera. Para isto ser possível é necessário disponibilizar acessos a quem queira consultar estes Ficheiros de uma forma simples e eficaz. Muitas das vezes o acesso a estes recursos é limitado e moroso, não existindo um mecanismo que permita uma procura rápida e objectiva. Mesmo que existam mecanismos de procura, a rapidez com que se obtêm os resultados não é a desejada e maior parte destes mecanismos têm regras confusas e aborrecidas para quem utiliza o sistema.

Por outro lado, os Autores de Programas que pretendem adicionar Ficheiros aos repositórios da rádio têm de o fazer localmente, na rádio, ou enviá-los para estes repositórios respeitando certas regras estabelecidas pelos seus administradores (e.g. através do protocolo FTP). Existem também soluções que permitem o upload de Ficheiros através de formulários, via HTTP. No entanto, as limitações que apresentam, como por exemplo a instabilidade no sucesso de entrega dos Ficheiros e más interfaces gráficas é uma barreira a quebrar para que os Autores as utilizem com maior frequência.
A associação de Metadados aos Ficheiros é uma mais valia para a organização dos Sistemas de Armazenamento sendo necessário que os Autores insiram estes dados, e esta actividade deve ser incentivada de forma a que não a ignorem por completo. Normalmente este processo é dispensado porque são solicitados demasiados dados sobre os novos Ficheiros e que os seus Autores acabam por não preencher devido ao tempo que este tipo de acções requerem.

Outro aspecto a ter em conta é a preservação dos Ficheiros. A solução mais viável (devido ao custo de sistemas de replicação actuais) das rádios comunitárias é o armazenamento a longo prazo em suportes físicos (e.g. discos rígidos, DVDs) onde apenas uma cópia dos Ficheiros é feita. Estudos [6] demonstram que a maioria destes suportes não dura mais do que alguns anos, colocando em risco todos os Ficheiros e consequentemente as Emissões armazenadas de épocas mais antigas até à actualidade.

3.1 Rádio Zero

A Rádio Zero serve como caso de estudo para a Análise do Problema e serve para obter uma melhor percepção da realidade das rádios comunitárias actuais.

3.1.1 Descrição da Rádio e Intervenientes

A Rádio Zero é uma rádio sem fins lucrativos e orientada para os Programas de Autor. É a rádio do Instituto Superior Técnico, sendo uma secção autónoma da Associação de Estudantes desta faculdade, e actualmente emite via *stream* na Internet. Pertence a uma rede de estações de rádio independentes, *The Radia Network*[1], que tem como objectivo a produção de Emissões relacionadas com a comunidade de cada rádio que faz parte do projecto.
A Rádio Zero é composta por quatro departamentos sendo eles o de Administração, Técnico e Logístico, Programação e o de *Marketing* e Publicidade. Os principais intervenientes dividem-se em dois grupos: os Colaboradores e os Membros. A principal diferença entre estes é que os Colaboradores possuem cargos nos diversos departamentos da rádio, mas qualquer elemento destes dois grupos pode ser Autor em vários Programas.

A estação de rádio possui um estúdio que permite aos Autores produzirem os Ficheiros que dão origem às Emissões e que ficam armazenados, em formato digital, nos suportes disponíveis. A Figura 3.1 mostra a Arquitectura actual do Sistema de Informação utilizado pela Rádio Zero para armazenar esses Ficheiros. Os nós que o compõem são os seguintes:

- **PC de Estúdio:** Arquivo principal da Rádio onde o Sistema de Ficheiros é utilizado gerir os Ficheiros das Emissões. Existe uma directoria por Programa da Rádio onde os respectivos Autores adicionam os seus Ficheiros;
- **PC de Gestão das Emissões:** computador que corre o *Playout*[2] e contém uma directoria específica onde são adicionados os Ficheiros das Emissões a serem emitidas por este módulo;
- **Servidor de FTP e Armazenamento de *Podcasts*:** este servidor serve para dois propósitos:
  - servidor de FTP para os Autores que produzem remotamente os Ficheiros para as Emissões dos Programas;
  - armazenar os *podcasts* que são disponibilizados para os ouvintes da Rádio Zero.
- **Servidores de Armazenamento:** servidores gratuitos que armazenam conteúdo digital, disponíveis na Internet, que são utilizados por alguns Autores em alternativa ao Servidor de FTP.

[1] http://www.radia.fm

[2] Módulo que a Rádio utiliza para gerir e transmitir as Emissões previamente armazenadas. Consultar [11] para melhor esclarecimento deste módulo.

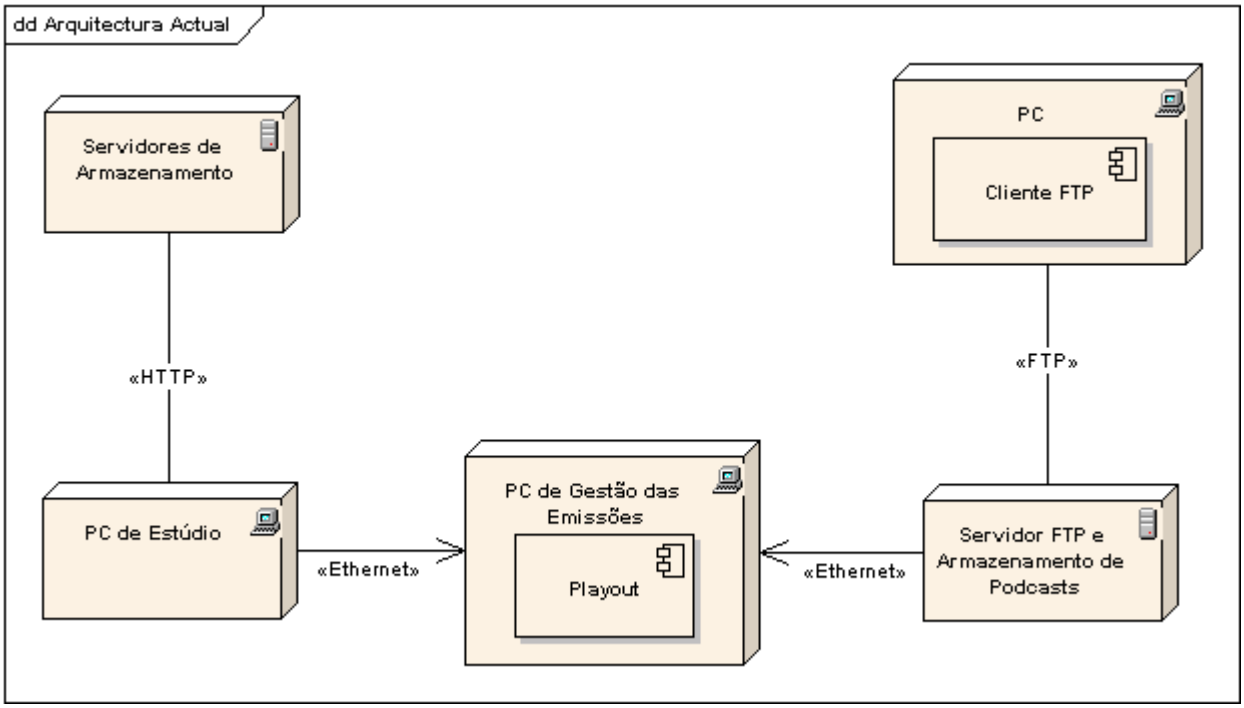

Figura 3.1: Arquitectura actual do Sistema de Armazenamento da Rádio Zero

3.1.2 Processo de Armazenamento e Preservação dos Ficheiros

3.1.2.1 Processo de Armazenamento

Antes de um Ficheiro ser armazenado tem que ser normalizado[3], pelos respectivos Autores, segundo normas definidas pelo Departamento de Programação. Quando o Ficheiro de uma Emissão é entregue remotamente

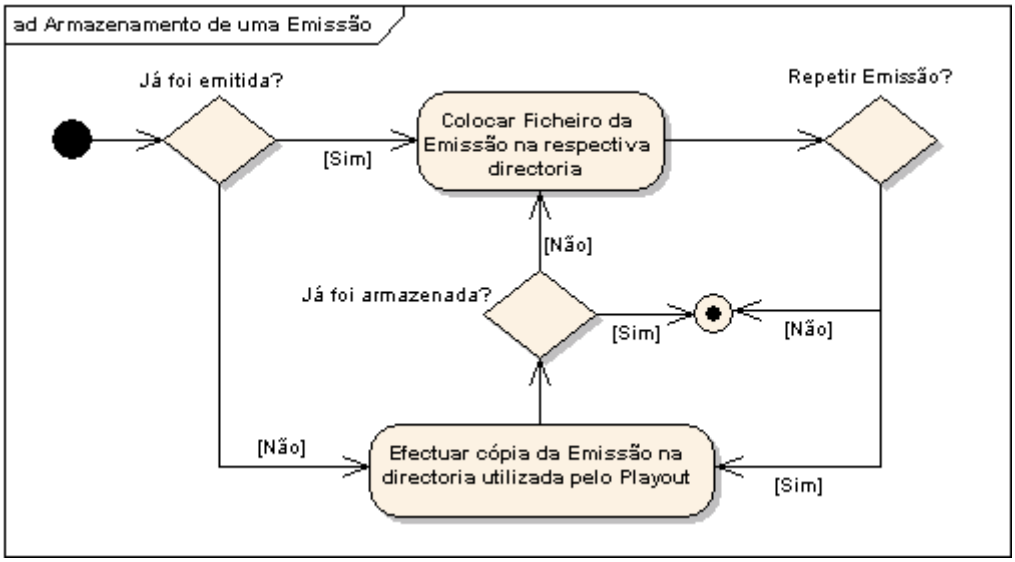

Figura 3.2: Processo de armazenamento das Emissões

[3] A normalização consiste em aumentar ou diminuir a amplitude do sinal de áudio dos Ficheiros, respeitando a amplitude definida pela rádio.

é necessário aceder previamente à sua localização antes de efectuar o processo de armazenamento. A sua localização pode ser o Servidor de FTP local, ou um URL que aponta para um Servidor de Armazenamento. Quando o Ficheiro se encontra pronto para ser armazenado, o processo de armazenamento segue o esquema da Figura 3.2.
No caso da Emissão já ter sido emitida tem de se proceder da seguinte forma:

- Normalizar o Ficheiro, caso não se encontre normalizado, segundo as regras estabelecidas pelo Departamento de Programação;
- Colocar o Ficheiro na directoria com o nome do Programa para o qual a Emissão foi criada;
- Se a Emissão é para ser repetida, efectua-se uma cópia do Ficheiro associado na directoria utilizada pelo *Playout*.

Caso contrário, o procedimento é o seguinte:

- Normalizar o Ficheiro;
- Copiar o Ficheiro para a directoria que o *Playout* utiliza;
- Colocar o Ficheiro na directoria com o nome do Programa para o qual a Emissão foi criada, caso ainda não se encontre armazenado.

O nome dos Ficheiros das Emissões segue uma norma estabelecida pelo Departamento de Programação consoante a directoria onde são armazenados. Quando o armazenamento é feito na directoria do Programa para o qual a Emissão foi criada a terminologia a utilizar é o nome do Programa juntamente com a data da Emissão no formato estabelecido pela norma ISO 8601[4]. Quando o Ficheiro da Emissão é para ser adicionado à directoria que o *Playout* utiliza a terminologia utilizada é apenas o nome do Programa.
Por exemplo, se determinado Autor cria uma nova Emissão para o Programa semanal PIB no dia 15 de Setembro do ano 2008, o Ficheiro deverá ter como nome '*pib20080915*' quando for adicionado à directoria PIB e '*pib*' quando for adicionado à directoria utilizada pelo *Playout*.

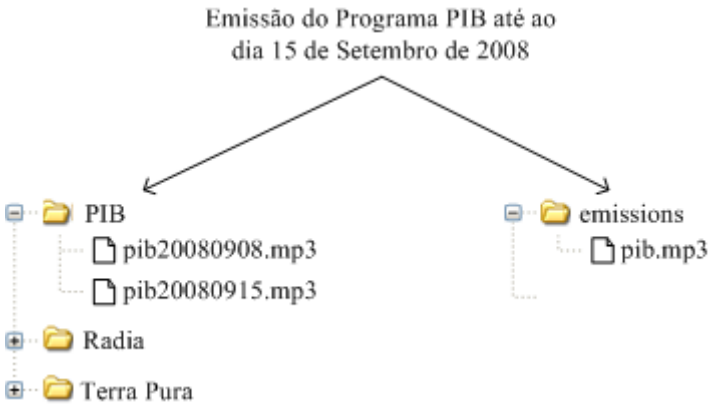

Figura 3.3: Nomenclatura dos Ficheiros das Emissões

[4] http://www.iso.org/iso/date_and_time_format

3.1.2.2 Processo de Preservação

O processo de preservação dos Ficheiros das Emissões da Rádio Zero é um pouco rudimentar, devido à limitação financeira da rádio e à inexistência de sistemas de informação que satisfaçam os requisitos para a resolução deste problema. Quando os suportes de armazenamento comuns da rádio (PC de Estúdio e Servidor de FTP) ficam perto do seu limite, os Ficheiros são gravados em DVD sendo esta a sua única cópia.

3.1.3 Metadados associados aos Ficheiros

A única maneira de associar Metadados aos Ficheiros, que são armazenados no repositório principal da Rádio, é através de tags ID3[5] e apenas quando estão no formato MP3. No entanto, os Ficheiros neste formato estão dependentes deste tipo de Metadados uma vez que para os associar ou consultar é necessário fazê-lo através do cabeçalho que os suporta. Os Autores conseguem associar aos Ficheiros os seguintes Metadados:

- Nomes dos Autores;
- Nome do Programa a que o Ficheiro pertence;
- Data da Emissão;
- Nome da Rádio (Rádio Zero);
- URL da Rádio;
- Descrição Livre acerca do Ficheiro.

Para além dos Metadados descritos também é possível obter o tamanho dos Ficheiros, assim como a sua duração.

3.1.4 Pesquisa de Ficheiros Armazenados

Antes do Processo de Preservação ser efectuado, os actores têm de percorrer as directorias dos Programas para encontrarem os Ficheiros que necessitam. Não existe nenhuma indexação que facilite este trabalho e durante o processo de pesquisa apenas o nome do Ficheiro ajuda a encontrar o que pretendem. Se algum dos Ficheiros possui Metadados têm de recorrer a operações adicionais (e.g. ver as Propriedades do Ficheiro) para os consultar, o que torna a pesquisa ainda mais morosa.
Após o Processo de Preservação, e caso os Ficheiros já não se encontrem nos suportes de armazenamento comuns da rádio, os actores para pesquisarem Ficheiros têm de o fazer manualmente nos DVDs de armazenamento.

3.2 Análise do Problema e Requisitos

Nesta secção, inicialmente, é dada uma explicação de conceitos que são abordados na Análise do Problema e utilizados posteriormente na Proposta da Solução e definição dos requisitos.
Os requisitos funcionais e não-funcionais para o problema descrito são apresentados nas secções 3.2.3 e 3.2.4, respectivamente. Para além de terem como referência o problema descrito, têm também a contribuição

[5] http://www.id3.org

do caso de estudo da Rádio Zero. Esta contribuição teve origem nas entrevistas realizadas a alguns elementos (dos departamentos de Administração e de Programação) da rádio.
Juntamente com os requisitos são descritos, resumidamente, os casos de uso que lhe dão suporte. No Apêndice A encontram-se os mesmos casos de uso com a totalidade da respectiva descrição.

3.2.1 Novos objectivos de negócio

Os conceitos apresentados e analisados de seguida são utilizados para a modelação do problema e permitem uma definição e enquadramento dos mesmos na solução apresentada.

3.2.1.1 *Tagging*

O processo de *Tagging* é um processo social que consiste na associação de palavras-chave (**Tags**), por parte dos actores, às entidades existentes no sistema. Permite categorizar as entidades e modelar o sistema de forma subjectiva. Por outro lado, ajuda os actores a procurarem de forma rápida as entidades que previamente categorizaram com as Tags. O objectivo deste processo não é associar Tags de uma forma correcta ou incorrecta, mas sim lembrar o que foi associado, conseguindo o actor moldar o sistema individualmente contribuindo também para toda a comunidade onde está inserido. Alguns do exemplos mais conhecidos que utilizam este processo são o *Flickr*[6] (Serviço que permite organizar e partilhar fotografias.), *Delicious*[7] (Permite gerir e partilhar *bookmarks*.) e o *Last.fm*[8] (Serviço social que permite ouvir e partilhar música.).

3.2.1.2 Comentários feitos por Actores de um Sistema de Informação

A associação de comentários a diversas entidades de um sistema permite uma maior interactividade entre os seus actores. Por outro lado, os comentários permitem aos actores avaliar os conteúdos existentes num sistema, medindo o impacto que causam para quem os consulta. Consequentemente, os comentários influenciam a produção de novos conteúdos possibilitando a melhoria da sua qualidade.

3.2.2 Casos de Uso

Os casos de uso que servem de suporte para os requisitos são apresentados de seguida. A descrição de cada caso de uso é uma descrição resumida, podendo ser consultada a descrição mais completa no Apêndice A. **Casos de Uso do Actor Anónimo:**

- Efectuar Registo: Caso de uso que permite um Actor Anónimo registar-se no sistema
- Pesquisar Ficheiros: Permite pesquisar Ficheiros previamente adicionados ao sistema
- Pesquisar Programas: Permite pesquisar Programas associados ao sistema
- Pesquisar Emissões: Permite pesquisar Emissões associadas ao sistema
- Ver Informação dos Ficheiros: Este caso de uso permite ao actor ver a informação associada a um Ficheiro

[6] http://www.flickr.com
[7] http://delicious.com
[8] http://www.last.fm

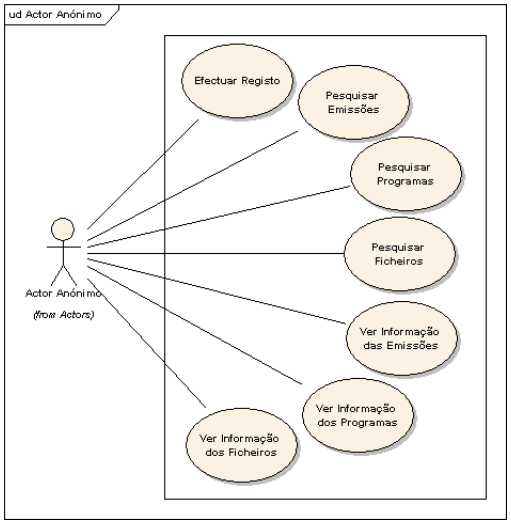

Figura 3.4: Casos de Uso do Actor Anónimo.

- Ver Informação dos Programas: Permite consultar a informação de cada Programa
- Ver Informação das Emissões: Permite consultar a informação de cada Emissão

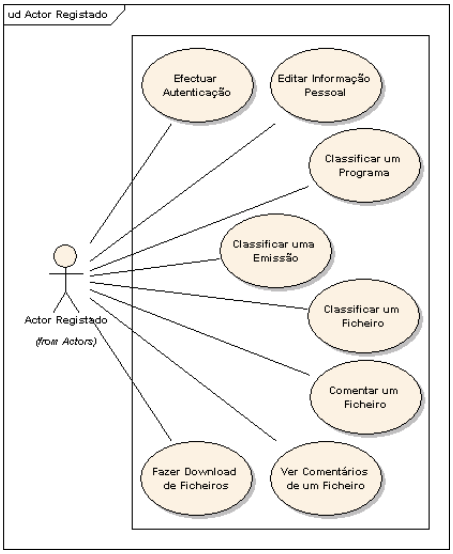

Figura 3.5: Casos de Uso do Actor Registado.

Casos de Uso do Actor Registado:

- Efectuar Autenticação: Um Actor Anónimo autentica-se no sistema
- Editar Informação Pessoal: O actor pode editar a sua informação pessoal sempre que necessite
- Classificar um Programa: Permite classificar um Programa com Tags
- Classificar uma Emissão: Permite classificar uma Emissão com Tags
- Classificar um Ficheiro: Permite classificar um Ficheiro com Tags
- Comentar um Ficheiro: Permite associar um comentário a um Ficheiro
- Ver Comentários associados a um Ficheiro: Permite ver os comentários associados a um Ficheiro
- Fazer Download de Ficheiros: Permite efectuar o download de um Ficheiro ou dos seus Metadados

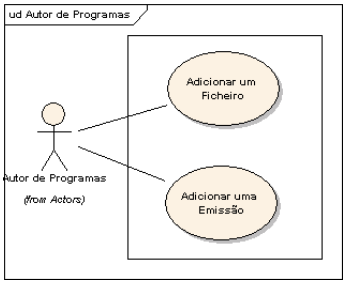

Figura 3.6: Casos de Uso do Autor de Programas.

Casos de Uso do Autor de Programas:

- Adicionar um Ficheiro: Este caso de uso permite ao actor adicionar novos Ficheiros ao sistema, incluindo os seus Metadados
- Adicionar uma Emissão: Permite a um Autor de Programa criar uma nova Emissão e associá-la ao respectivo Programa

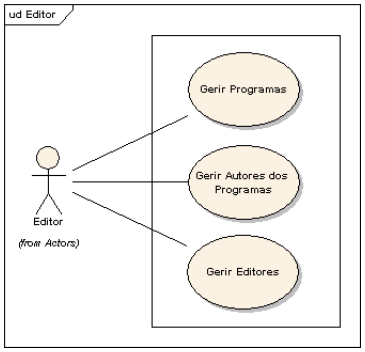

Figura 3.7: Casos de Uso do Editor da Rádio.

Casos de Uso do Editor:

- Criar um novo Programa: Permite a criação de novos Programas na Rádio
- Editar a Informação de um Programa: Permite alterar a informação de um Programa sempre que necessário
- Gerir Autores dos Programas: Este caso de uso permite ao actor adicionar ou remover a autoria de um Programa a outro actor
- Gerir Editores da Rádio: Um Editor pode adicionar ou remover o privilégio de Edição a outros actores do sistema

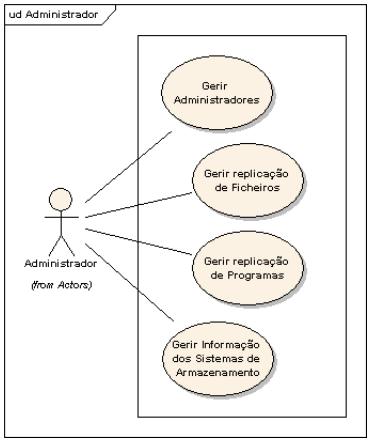

Figura 3.8: Casos de Uso do Administrador da Rádio.

Casos de Uso do Administrador:

- Gerir Administradores: Permite ao actor gerir o privilégio de administração dos actores da rádio
- Gerir Replicação de Ficheiros: Permite gerir a replicação de cada Ficheiro nos Sistemas de Armazenamento associados ao sistema
- Gerir Replicação de Programas: Permite gerir a replicação de cada Programa nos Sistemas de Armazenamento associados ao Sistema
- Criar um Sistema de Armazenamento: Permite adicionar um novo Sistema de Armazenamento ao sistema. Este novo Sistema de Armazenamento servirá como sistema de replicação para os Ficheiros associados ao sistema
- Editar Informação de um Sistema de Armazenamento: Permite alterar a informação dos Sistemas de Armazenamento presentes no sistema
- Gerir Replicação de Programas por Sistema de Armazenamento: Permite gerir a replicação dos Programas por Sistema de Armazenamento e visualizar no momento que Programas estão ou não replicados num determinado Sistema
- Gerir Replicação de Ficheiros por Sistema de Armazenamento: Permite gerir a replicação dos Ficheiros por Sistema de Armazenamento e visualizar no momento que Ficheiros se encontram replicados, ou não, num determinado Sistema

3.2.3 Requisitos Funcionais

Os requisitos funcionais especificam o comportamento do sistema apresentado a funcionalidade e detalhes técnicos do mesmo. De seguida são apresentados os requisitos funcionais que resultaram da análise ao problema especificado.

RF 1 : Adicionar, editar e remover Programas

RF 2 : Gerir Autores dos Programas (dar e retirar o privilégio de Autor)

RF 3 : Aceder às Emissões associadas a um Programa

RF 4 : Permitir a consulta de todos os Programas

RF 5 : Adicionar, editar e remover Emissões

RF 6 : Preencher automaticamente os dados genéricos das novas Emissões (e.g. Nome e Descrição) com os dados do Programa ao qual as Emissões são associadas

RF 7 : Aceder aos Ficheiros associados a uma Emissão

RF 8 : Adicionar, editar e remover Ficheiros

RF 9 : Associar Ficheiros às Emissões quando estas são criadas

RF 10 : Associar outros Autores a novos Ficheiros

RF 11 : Aceder às Emissões onde o Ficheiro em consulta se encontra associado

RF 12 : Aceder aos Programas onde existem Emissões que contêm o Ficheiro em consulta

RF 13 : Adicionar Comentários aos Ficheiros

RF 14 : Apresentar os comentários associados a um Ficheiro

RF 15 : Fazer download de Ficheiros

RF 16 : Fazer download dos Metadados associados aos Ficheiros

RF 17 : Permitir a consulta de todos os Ficheiros

RF 18 : Adicionar Tags aos Ficheiros, Programas e Emissões

RF 19 : Adicionar Tags a tempos específicos dos Ficheiros

RF 20 : Consultar Ficheiros, Programas e Emissões através de Tags

RF 21 : Apresentar as Tags associadas a cada Ficheiro, Programa e Emissão

RF 22 : Pesquisar Ficheiros, Programas e Emissões através de qualquer palavra-chave (e.g. nome, género)

RF 23 : Pesquisar Ficheiros, Programas e Emissões através de Tags

RF 24 : Disponibilizar aos Autores os respectivos Ficheiros de autoria e os Programas onde possui o privilégio de Autor

RF 25 : Gerir actores do sistema e respectivos privilégios

RF 26 : Permitir a um Actor Registado alterar os seus dados

RF 27 : Adicionar, editar e remover Sistemas de Armazenamento

RF 28 : Adicionar e remover réplicas dos Ficheiros nos Sistemas de Armazenamento

RF 29 : Obter Ficheiros replicados para o Repositório Principal quando se verificar a ausência destes

RF 30 : Remover Ficheiros temporários quando expirar a data escolhida como validade no sistema

3.2.4 Requisitos Não-Funcionais

Os requisitos não-funcionais dão suporte ao requisitos funcionais e especificam restrições do sistema, como por exemplo critérios de qualidade. Alguns dos requisitos não-funcionais apresentados têm como base a Rádio Zero que serviu como caso de estudo para a análise do problema.

Os requisitos não-funcionais para o sistema são:

RNF 1 : Os dados introduzidos pelos actores devem ficar disponíveis imediatamente em todo o sistema

RNF 2 : A interface disponibilizada aos actores deve ser de fácil compreensão

RNF 3 : A organização das entidades do sistema (e.g. Ficheiros, Emissões, Programas) deve ser idêntica à actual, utilizada pelos actores da rádio

RNF 4 : A conclusão das operações de replicação dos Ficheiros tem de ser efectuada com sucesso

RNF 5 : A disponibilidade do sistema deve ser acima dos 90%

RNF 6 : O sistema deve correr numa máquina com poder de processamento heterogéneo

RNF 7 : A máquina onde se encontra o Sistema de Armazenamento Principal deve possuir uma capacidade de armazenamento suficiente para atender todos os pedidos com sucesso

RNF 8 : O Sistema de Armazenamento Principal deve ser escalável, no sentido em que é possível aumentar a capacidade de armazenamento do sistema através da adição de novos suportes físicos de armazenamento

RNF 9 : Os Ficheiros do Sistema de Armazenamento Principal podem ser preservados através da replicação noutros Sistemas de Armazenamento que se encontram associados ao sistema

4 Proposta da Solução

Este capítulo apresenta a solução para o problema apresentado tendo como principal foco os **Autores, a comunidade de rádio e a interacção com Ficheiros**, abstraindo-se do paradigma que os sistemas estudados no Capítulo 2 utilizam. A solução apresentada está alinhada com a dissertação de Daniel Z. Silva [11] de modo a que ambos os sistemas consigam interagir entre eles, apresentando conceitos de Domínio idênticos.

4.1 Introdução

Um dos problemas cruciais a resolver é disponibilizar informação acerca dos Ficheiros produzidos, assim como de todas as interacções que os envolve (e.g. a que Emissões pertencem e respectivos Autores). Para colmatar este problema, terá de existir um ponto de acesso onde os intervenientes da rádio (e.g. Autores, ouvintes e outros Sistemas de Informação) possam consultar toda esta informação e usá-la para proveito próprio.

Com a constante utilização de Ficheiros já armazenados, os actores da rádio têm necessidade de lhes aceder e reutilizá-los, pelo que é necessário um acesso rápido. Um exemplo disso, é a repetição de algumas Emissões previamente emitidas. Deste modo, é necessário existir um **Sistema de Armazenamento Principal** onde os actores possam aceder imediatamente, para recuperarem os Ficheiros pretendidos.

Por outro lado, existe a necessidade de os armazenar a longo prazo e de forma segura. A solução mais viável é a replicação dos Ficheiros por outros **Sistemas de Armazenamento**, excluindo o Principal, de modo a preservá-los ao longo do tempo e conseguir aceder-lhes quando se justificar (Figura 4.1).

De seguida será abordado o conceito de Arquitectura de Software e o que a define, assim como alguns Padrões Arquitecturais para dar um melhor enquadramento para a descrição da solução.

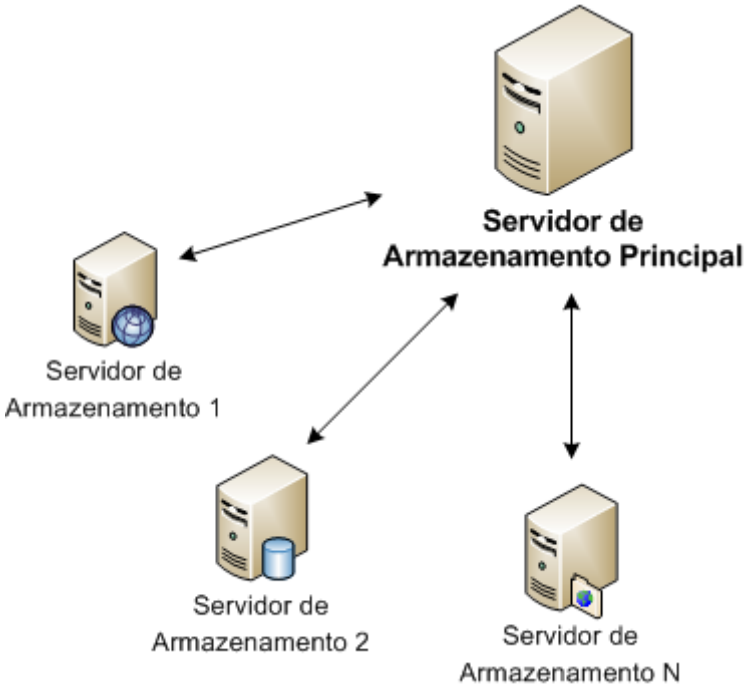

Figura 4.1: Organização dos Sistemas de Armazenamento

4.2 Arquitectura de Software e Padrões Arquitecturais

Nesta secção será dada uma descrição sobre o que é uma arquitectura de software [5] e alguns exemplos dos padrões arquitecturais [3] que serão utilizados no trabalho desenvolvido.

4.2.1 Arquitectura de Software

A Arquitectura de Software de um sistema computacional é um conjunto de componentes de software, a descrição das suas propriedades e relações entre esses componentes.

Quando se projecta a arquitectura de software de um sistema "divide-se para conquistar", dividindo as funcionalidades do sistema por partes resultando no final um todo que funciona correctamente segundo as especificações do negócio para o qual se pretende resolver o problema. Para além disso, uma arquitectura de software também é influenciada por outros factores: intervenientes no sistema (e.g. clientes e actores), organização que desenvolve a arquitectura, experiência do arquitecto de software e ambiente técnico onde a arquitectura é desenvolvida.
Pela definição dada, os **componentes** de uma arquitectura representam os elementos de software que abstractamente disponibilizam funcionalidades do sistema por meio de uma API (as suas **propriedades**). A parte mais detalhada e privada de cada componente é deixada para outra fase do processo de desenvolvimento de software, a fase de Implementação.
As **relações** entre os diversos componentes (também denominadas de conectores) descrevem o modo como estes interagem entre eles, quando o sistema se encontra em funcionamento, transportando as respectivas funcionalidades de componente para componente.

4.2.2 Padrões Arquitecturais

Um padrão arquitectural (ou estilo arquitectural) descreve um problema que ocorre frequentemente num determinado ambiente, assim como a solução base para esse mesmo problema. Descreve as ideias gerais a aplicar ao problema em questão, para que se possa resolver sem ter de executar todos os passos para determinar a arquitectura do problema.
Um padrão baseia-se nas observações práticas (e.g. de como os processos se comportam, como as pessoas agem) para conseguir obter uma solução que possa ser aplicada em casos semelhantes ao problema que se está a observar. No entanto, um padrão não é "a solução"definitiva para o problema que se pretende resolver. *Martin Fowler* refere-se aos padrões arquitecturais como uma receita meia confeccionada: dão-nos os elementos base para a resolução do problema mas quem o resolve tem de terminá-lo no contexto do problema que está a resolver.

Depois de analisados os requisitos do problema no Capítulo 3 vão ser abordados alguns padrões arquitecturais para a resolução do problema desta dissertação.
O primeiro a abordar é o padrão arquitectural **Modelo-Vista-Controlador** (MVC). Este é um padrão de Apresentação Web e é dos mais conhecidos neste ambiente. A sua composição baseia-se em três componentes

(Figura 4.2):

- **Modelo:** representa o domínio da aplicação. Contém todos os dados e comportamento da aplicação, necessários para apresentar no componente da Vista;
- **Vista:** este componente tem como função apresentar a informação gerada pelo Modelo na interface do utilizador. Exemplo de uma Vista é uma página HTML com informação vinda de um objecto do Modelo;
- **Controlador:** suporta qualquer alteração à informação disponibilizada nas Vistas e reflecte essas alterações no Modelo.

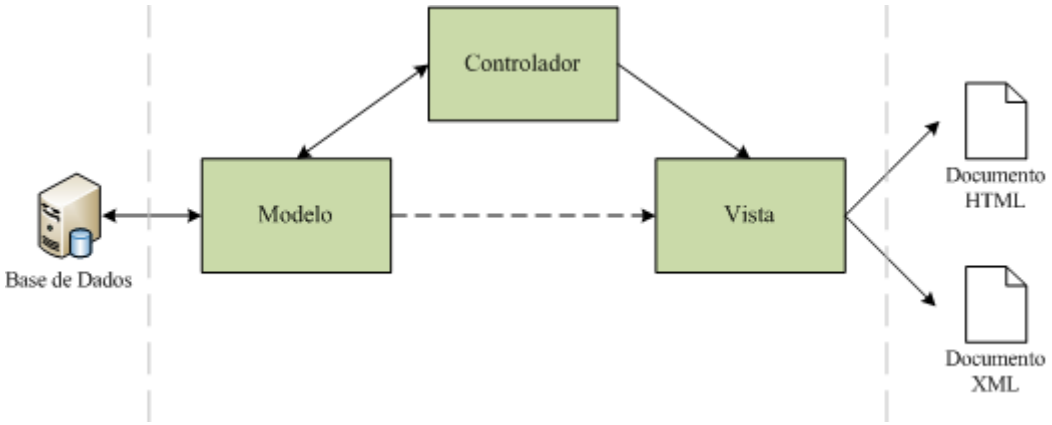

Figura 4.2: Padrão Arquitectural Modelo-Vista-Controlador

A relação entre o componente do Modelo com o componente da Vista é das mais importantes. É necessário existir uma separação entres estes dois componentes para que resulte da aplicação deste padrão um óptimo desenho da arquitectura do sistema. Uma das principais vantagens desta separação é que com o mesmo Modelo pode-se construir variadas Vistas por diferentes intervenientes. Por outro lado, consegue-se criar uma independência entre a interface, que é apresentada ao utilizador, e a lógica de negócio da aplicação.

O segundo padrão arquitectural a abordar é o **Modelo de Domínio** (Figura 4.3) que é englobado nos padrões de Lógica de Domínio. Este padrão é composto essencialmente por componentes de software que possuem os dados e/ou modelam a lógica do negócio. Normalmente, estes componentes são classes (programação

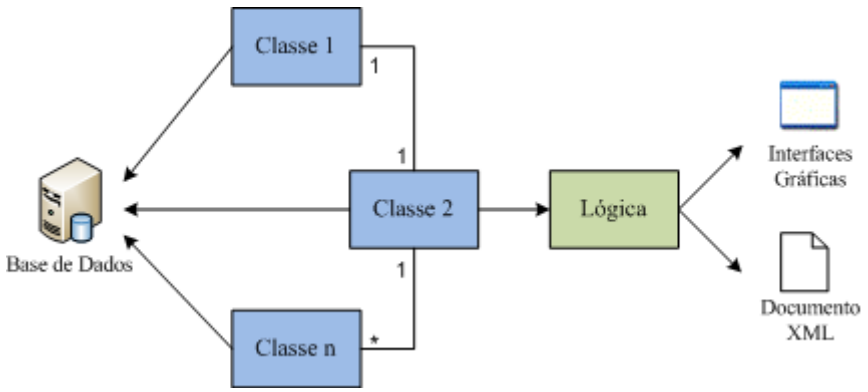

Figura 4.3: Padrão Arquitectural Modelo de Domínio

Orientada a Objectos) e podem estar directamente relacionados com a base de dados do sistema ficando,

por vezes, este padrão parecido com o desenho da base de dados que utiliza. Por exemplo, uma classe denominada de *Pessoa* contém os dados de determinado indivíduo que são obtidos através da base de dados (e.g. nome, data de nascimento, morada) mas também pode conter lógica de negócio onde é utilizada (e.g. pode calcular determinado desconto para esse indivíduo consoante a idade dele).
Um dos principais problemas deste padrão é o tamanho que os objectos de domínio tomam à medida vão sendo desenvolvidos. Esta preocupação é resolvida separando a lógica dos objectos que resolve situações pontuais noutro componente (e.g. outra classe ou colocá-la na parte de apresentação), mas por outro lado se é separada do objecto que representa leva a uma complexidade e inconsistência do sistema devido a possíveis replicações de funcionalidades ou alterações indesejáveis. Fica da responsabilidade de quem desenvolve o sistema tomar este tipo de decisões e adoptá-las da melhor maneira possível.

4.3 Padrão Arquitectural REST

O padrão arquitectural REST[1] é um padrão utilizado em Sistemas Distribuídos relacionados com hipermédia, como por exemplo a *World Wide Web*. Foi *Roy Thomas Fielding* que introduziu o conceito, em 2000, na sua dissertação de doutoramento [2], onde tenta combinar vários padrões arquitecturais já conhecidos (descritos em [2]) para chegar a este padrão.
A sua grande vantagem é que não se foca nos detalhes da implementação dos componentes e na sintaxe do protocolo que utiliza, mas sim na funcionalidade que os componentes desempenham, nas restrições das suas relações e interpretação dos dados que são trocados através dos componentes.
Assim como a definição de uma Arquitectura de Software, *Roy T. Fielding* também descreve o padrão REST através dos seus componentes, conectores e elementos de dados. As tabelas seguintes descrevem os elementos de dados (Tabela 4.1), conectores (Tabela 4.2) e componentes do padrão REST (Tabela 4.3).

Tabela 4.1: Elementos de dados do padrão REST

Recurso	Qualquer informação que possa ser nomeada (e.g. uma imagem, um ficheiro áudio, uma colecção de outros recursos ou um objecto não-virtual como um Utilizador do sistema). Elemento de abstracção do padrão REST.
Identificador do Recurso	Identifica o recurso específico envolvido na interacção entre dois componentes (e.g. URL de um ficheiro áudio, onde o ficheiro áudio representa o recurso).
Representção	Captura o estado de um recurso e é trocada entre os diversos componentes (e.g. documento HTML, documento XML).
Metadados da Representação	Descreve os dados que são incluídos numa representação.
Metadados do Recurso	Fornece informação acerca do recurso que não está incluída na representação que representa esse mesmo recurso.
Dados de Controlo	Dados que definem o objectivo da mensagem trocada entre os componentes, assim como todas as restrições que a envolvem (e.g. tipo de representação).

[1] *Representational State Transfer*

Tabela 4.2: Conectores do padrão REST

Cliente	Efectua o pedido ao servidor para efectuar operações sobre um ou mais recursos (e.g. solicitar uma representação de um recurso).
Servidor	Atende os pedidos dos clientes, e responde aos mesmos consoante o serviço solicitado.
Cache	Guarda respostas de interacções já feitas (entre cliente e servidor) de modo a fornecer os mesmos pedidos mais rapidamente, se forem solicitados. Este conector pode estar localizado no cliente ou no servidor.
Resolver	Traduz identificadores de recursos (parcialmente ou todo o identificador) em endereços de rede para que possa ser feita a comunicação entre diversos componentes (e.g. traduzir o URL do ficheiro de áudio para o respectivo DNS).
Tunnel	Transmite a comunicação de uma ligação (e.g. *firewall* ou um *proxy* HTTP).

Tabela 4.3: Componentes do padrão REST

Servidor de Origem	Utiliza o conector servidor para controlar o espaço de nomes quando um recurso (ou operação sobre um recurso) é solicitado. Fornece também as representações solicitadas para os recursos.
Gateway	Componente intermédio que pode ser imposto pela rede ou pelo Servidor de Origem e que fornece interfaces para encapsular outros serviços para tradução de dados, melhoramento de performance ou protecção.
Proxy	Este componente é idêntico ao componente *Gateway* com a diferença de ser seleccionado pelo cliente.
Agente	Utiliza o conector cliente para iniciar a comunicação, com um pedido, e é o último destinatário da resposta (e.g. Web *browser*).

4.4 Arquitectura do Sistema

O sistema apresentará dois componentes principais: uma aplicação que interage com os actores da rádio (que será denominada de *Radiastore*) e um *daemon*[2] (denominado de *Radiastore Daemon*) que gere a replicação e manutenção dos Ficheiros dos Autores da rádio.

A separação do sistema nestes dois componentes deve-se a vários factores, entre eles, o facto de toda a gestão efectuada pelo componente *Radiastore Daemon* ter de ser feita de forma automática. Para além disso, permite que este componente possa ser substituído por outro componente que tenha uma gestão de replicação diferente.

A componente *Radiastore* será uma aplicação Web que permitirá uma maior escalabilidade para a comunidade da rádio assim como uma melhor gestão dos Ficheiros para os respectivos Autores (e.g. processos de entrega, download de Ficheiros), acrescentando a disponibilidade de interfaces (através do padrão REST) possibilitando o acesso aos recursos presentes no sistema. Estas interfaces podem ser consumidas por quem mostrar interesse: actores da rádio, actores fora da comunidade da rádio e outros sistemas computacionais (e.g. sistema de gestão de processos de uma estação de rádio apresentado por Daniel Z. Silva [11]).

A comunicação entre os dois componentes será feita através da partilha de dados por meio de uma base de

[2] Programa computacional que corre em segundo plano sem intervenção dos actores.

dados.

Os Modelos de Domínio de cada componente encontram-se no Apêndice B devido ao tamanho que as figuras apresentam.

4.4.1 Actores do Sistema

As classes de actor do sistema estão definidos na secção de Conceitos, no início deste documento e estão representados graficamente com as respectivas relações na Figura A.1, do Apêndice A. As operações que cada classe de actor pode efectuar no sistema são especificadas de seguida.

Actor Anónimo:

- Pesquisar Ficheiros, Programas e Emissões;
- Consultar informação de um Ficheiro, Programa ou de uma Emissão;
- Registar-se no sistema obtendo o privilégio de Actor Registado.

Para além das operações que o Actor Anónimo pode efectuar, um **Actor Registado** pode também:

- Classificar Ficheiros, Programas e Emissões;
- Fazer download de Ficheiros e respectivos Metadados.

Um **Autor de Programas**, em conjunto com todas as operações definidas até agora, pode ainda Gerir (adicionar, remover e editar) Ficheiros e Emissões.

Os **Editores** da rádio ainda podem:

- Gerir Programas;
- Adicionar e remover Autores aos Programas;
- Adicionar e remover Editores do sistema.

O **Administrador** da rádio pode:

- Adicionar e remover Administradores de rádio;
- Replicar Ficheiros nos diversos Sistemas de Armazenamento;
- Gerir Sistemas de Armazenamento presentes no sistema.

Opcionalmente, pode existir um **Administrador de Sistema** que tem de efectuar as seguintes operações:

- Configuração dos componentes do sistema;
- Gestão da(s) máquina(s) onde correm os componentes do sistema e o Sistema de Armazenamento Principal.

4.5 Aplicação Web *Radiastore*

4.5.1 Descrição

Este componente foca-se na interacção dos actores com o sistema onde efectuam todas as operações envolvendo as principais entidades do sistema: Ficheiros, Programas e Emissões. Para além disso disponibiliza as APIs para os recursos que se encontram disponíveis no sistema.

Como suporte para a explicação deste componente deve recorrer-se ao seu Modelo de Domínio (Figura B.1

no Apêndice B). Neste modelo são apresentadas todas as entidades que fazem parte deste domínio e as respectivas relações.

4.5.2 Estrutura

A aplicação Web *Radiastore* tem de fornecer aos actores do sistema, informação sobre os Ficheiros armazenados e todas as relações destes com as restantes entidades que compõem o sistema conforme foi especificado nos requisitos funcionais.
Para que os Ficheiros (cuja informação é disponibilizada por este componente) estejam sempre disponíveis para os actores deve existir um Sistema de Armazenamento Principal com todos os Ficheiros disponíveis. Nos casos de falha, em que algum dos Ficheiros não se encontre nesse Sistema de Armazenamento, o componente *Radiastore Daemon* trata de recuperá-los, como descrito na secção 4.6.4.
Para além deste componente disponibilizar um sistema de autenticação próprio, também é possível que um actor se autentique através do serviço *OpenID*[3] ficando automaticamente com o privilégio de Actor Registado. Este método de autenticação é mais um meio para o alargamento das comunidades das rádios, eliminando a necessidade de um Actor Anónimo efectuar o registo no sistema no caso de possuir um identificador *OpenID*.
Algumas das entidades descritas (secção 4.5.2.1) já foram definidas na secção de Conceitos para uma melhor compreensão do documento nos capítulos anteriores.

4.5.2.1 Entidades

Um **Ficheiro** (*Audio Asset*) é a entidade que representa qualquer conteúdo digital (e.g. áudio, imagem, documento de texto) que pode ser armazenado no sistema. É composto pelos respectivos dados (o conteúdo digital) denominado de **Essência** (*Essence*) e um **documento de Metadados** (*MetadataFile*). O formato deste documento pode conter qualquer representação (e.g. MPEG-7[4], *Dublin Core*), desde que contenha a informação do Ficheiro que está representada no sistema. Para os requisitos apresentados, apenas serão considerados os ficheiros áudio como Essência dos Ficheiros que podem ser adicionados ao sistema.
Um Ficheiro possui uma data de validade, no caso do Autor que o adiciona querer que seja removido do sistema ao fim de determinado tempo. Para garantir a integridade do Ficheiro, é-lhe associado um *checksum* através da função criptográfica MD5[5], quando é armazenado. A associação de um documento de Metadados ao Ficheiro faz com que a associação destes dados possa ser efectuada de forma automática, no caso de o sistema reconhecer o formato de Metadados que se encontra representado nesse documento. Desta forma, a inserção dos dados pelo Autor será desnecessária tornando o processo de criação de um Ficheiro mais eficiente e cómodo, deixando para o Autor apenas a confirmação dos dados associados.
As principais entidades que se relacionam com os Ficheiros são os **Actores** e **Emissões**. Entre os Actores existe uma relação de Autoria indicando que determinado actor é Autor de um Ficheiro, como é descrito na sub-secção 4.5.2.2. Relativamente às Emissões, um Ficheiro pode pertencer a múltiplas destas entidades.

[3] http://openid.net
[4] http://www.chiariglione.org/MPEG/standards/mpeg-7/mpeg-7.htm
[5] http://www.ietf.org/rfc/rfc1321.txt

Esta relação permite que os Ficheiros sejam reutilizados nas Emissões sem ter de os replicar fisicamente nos Sistemas de Armazenamento. Apenas se cria a relação entre o Ficheiro e a Emissão a ser criada.

Uma **Emissão** (*Broadcast*) é definida por ocorrer num determinado espaço temporal e é constituída por um ou mais Ficheiros. Uma Emissão deve pertencer a um Programa e serve para manter um histórico do que é emitido na rádio. Deste modo, é possível definir **Ocorrências** (*Occurrences*) refinando o que é emitido em cada Emissão. As Ocorrências permitem definir o momento exacto em que os Ficheiros são emitidos numa Emissão. Uma Ocorrência contém o momento inicial e o momento final em que o Ficheiro é emitido na Emissão.
O conjunto destas associações permite construir uma **Grelha** (*Schedule*) com as Emissões emitidas num determinado intervalo de tempo definido por quem consulta o sistema.

Um **Programa** (*Program*) é a entidade temática da rádio e é onde as Emissões são associadas. Existe uma relação com os **Actores**, indicando que um actor é Autor do Programa, como é descrito na sub-secção 4.5.2.2.
A criação desta entidade serve também para organizar as Emissões criadas e também para manter uma correspondência com os Sistemas de Informação das rádios actuais. Temos como exemplo a Rádio Zero, que organiza as Emissões por directorias que representam os Programas da rádio.

4.5.2.2 Relações de Autoria

Existem dois tipos de relações de Autoria: **Autoria de Ficheiros** (envolvendo **Ficheiros** e **Actores**) e **Autoria de Programas** (envolvendo **Programas** e **Actores**). O privilégio de Autoria é concedido a um actor conforme é indicado na secção 4.4.1.
A Autoria de Ficheiros depende da Autoria de Programas porque um Autor apenas pode adicionar novos Ficheiros se for Autor de, pelo menos, um Programa. Isto faz com que exista uma restrição na adição de novos Ficheiros ao sistema e consequentemente novas Emissões (apenas pode criar Emissões quem possuir privilégio de Autor de Programa).
O motivo da existência destes dois tipos de Autoria é o ciclo de vida de um Autor de Programa. No caso de um Editor decidir que determinado actor já não é Autor de um Programa, todas as relações entre esse actor e os Ficheiros que lhe estão associados deixariam de existir. Deste modo, quando adiciona um Ficheiro é-lhe associado também o privilégio de Autor do novo Ficheiro.
Os actores podem assim consultar os Ficheiros e Programas dos quais são Autores.

4.5.3 *Tagging*

Aproveitando todas as vantagens do processo de *Tagging* (descrito na secção 3.2.1.1), os actores do componente Web *Radiastore* poderão associar Tags a Ficheiros, Programas e Emissões na informação detalhada de cada entidade. Como é comum nas páginas Web sociais que adoptam este processo, será disponibilizado um motor de busca onde se poderá pesquisar cada entidade por Tags (ver secção 4.5.5). Por outro lado, se

o actor optar por consultar Ficheiros, Programas ou Emissões por Tags será disponibilizada uma *Tag Cloud*[6] por entidade.

Uma funcionalidade emergente associada ao processo de *Tagging* é a associação de Tags a diversas partes das entidades, e até mesmo pequenas descrições textuais[7]. Esta funcionalidade pode ser adoptada para os Ficheiros existentes no sistema, permitindo melhorar a especificação da sua categorização e pesquisas mais refinadas.

4.5.4 Comentários para Ficheiros

De modo a incentivar mais a interactividade dos actores na comunidade da rádio, podem ser associados comentários aos Ficheiros.
Um actor pode introduzir texto livre e limitado, transmitindo a sua opinião relativamente ao Ficheiro a comentar, para que os outros actores tenham a percepção do âmbito e conteúdo do Ficheiro para além da descrição e informação fornecida pelos Autores (ou Autor) do respectivo Ficheiro. Por outro lado, os Autores podem avaliar o impacto do conteúdo do Ficheiro na comunidade da rádio, podendo tomar decisões relativamente à produção de novos Ficheiros sobre o mesmo tema.

4.5.5 Pesquisa de Entidades

O processo de pesquisa será feito objectivamente em Ficheiros, Programas ou Emissões. Esta opção limita a pesquisa apenas a uma entidade refinando melhor os seus resultados. Caso o actor deseje uma consulta mais abrangente, pode consultar a listagem completa que é disponibilizada para cada entidade.
O processo de pesquisa pode ser efectuado de duas maneiras: pesquisa através de palavras-chave relacionadas com os atributos das entidades (e.g. nome do Ficheiro, palavra na descrição de um Programa), ou através das Tags associadas previamente às entidades.

Quando a pesquisa é efectuada através de palavras-chave presentes na informação de cada entidade (Ficheiro, Programa ou Emissão), para além de seleccionar a entidade que pretende pesquisar o actor pode ainda refinar os atributos da pesquisa. Por omissão, serão incluídos na pesquisa os atributos mais utilizados (nome e descrição de cada entidade). Se o actor pretender será mostrada uma lista com os atributos da entidade seleccionada onde pode escolher aqueles que pretende incluir na pesquisa.

Se o actor optar por efectuar uma pesquisa por Tags, para além de seleccionar a entidade que pretende pesquisar, tem de alterar o modo de pesquisa inserindo de seguida as Tags.

4.5.6 Recursos disponíveis

Os recursos web disponíveis no sistema serão listados de seguida. A descrição dos recursos é composta pelas partes descritas na Tabela 4.4, e será feita parcialmente em inglês para coincidirem e facilitarem as

[6] http://blog.pietrosperoni.it/2005/05/25/tag-clouds-metric
[7] Anotações em vídeos do *YouTube*: http://www.youtube.com/t/annotations_about

representações a serem implementadas.

Tabela 4.4: Composição dos recursos web do sistema

URLs	Identificadores que permitem aceder aos recursos web disponíveis no sistema. Os identificadores podem conter variáveis que serão representadas da seguinte forma: /localização/do/recurso/{variável}. Para além disso, podem retornar um recurso simples (recurso) ou um conjunto de recursos (colecção).
Elementos	Atributos que são retornados na representação de um recurso simples, juntamente com os respectivos tipos de dados. No caso de um atributo ser um recurso, o tipo de dados indicado será *URL do Recurso* (*Resource URL*)

4.5.6.1 Programas

Os Programas (ou Programa) podem ser acedidos a partir dos URLs especificados na Tabela 4.5. A representação de um Programa apresenta o seu nome, descrição e data de criação.

Tabela 4.5: Recursos disponíveis para Programas

URLs	/programs (colecção)
	/programs/{id} (recurso)
Elementos	*Name* (*String*)
	Description (*Text*)
	Creation Date (*Date*)

4.5.6.2 Emissões

Pode ser consultada a totalidade de Emissões existentes na rádio, com a possibilidade de filtrar a consulta por data (indicando o ano e opcionalmente o mês e dia) como é apresentado na Tabela 4.6, ou consultar cada Emissão individualmente especificando o seu identificador.
A consulta de Emissões por Programa também é possível, sendo o procedimento idêntico à consulta descrita anteriormente.
A representação de uma Emissão é composta pelo nome, descrição, data em que foi emitida, Género, Idioma, Programa a que pertence e os Ficheiros que a compõem.

4.5.6.3 Ficheiros

Como foi referido no início deste capítulo, a interacção com os Ficheiros é dos aspectos mais relevantes para a solução apresentada. Deste modo, é importante que seja disponibilizada uma representação para estes recursos que seja a mais completa possível. A Tabela 4.7 apresenta o método de acesso a estes recursos. Para consulta de um conjunto de Ficheiros, é dada a possibilidade de os consultar na totalidade ou filtrar a consulta por data de criação (indicando o ano e opcionalmente o mês e dia). Para obter uma representação de um Ficheiro em particular é necessário especificar o seu identificador, ou indicar a data em que foi criado juntamente com o respectivo identificador.

Tabela 4.6: Recursos disponíveis para Emissões

URLs	/broadcasts (colecção)
	/broadcasts/{id} (recurso)
	/broadcasts/{year}/{month}/{day} (colecção)
	/broadcasts/{year}/{month}/{day}/{id} (recurso)
	/programs/{program_id}/broadcasts (colecção)
	/programs/{program_id}/broadcasts/{year}/{month}/{day} (colecção)
	/programs/{program_id}/broadcasts/{year}/{month}/{day}/{id} (recurso)
Elementos	*Name* (*String*)
	Description (*Text*)
	Broadcast Date (*DateTime*)
	Genre (*String*)
	Language (*String*)
	Program (*Resource URL*)
	Audio Assets (*Resource URLs*)

Outro procedimento para obter uma representação de Ficheiros (ou Ficheiro em particular) é através das Emissões, indicando a Emissão para a qual se deseja consultar os Ficheiros.
A representação fornecida apresenta o nome, descrição, data de criação, formato, tamanho (em *bytes*), *checksum* MD5 e a lista de Programas e Emissões aos quais o Ficheiro pertence.

Tabela 4.7: Recursos disponíveis para Ficheiros

URLs	/audio_assets (colecção)
	/audio_assets/{id} (recurso)
	/audio_assets/{year}/{month}/{day} (colecção)
	/audio_assets/{year}/{month}/{day}/{id} (recurso)
	/broadcasts/{id}/audio_assets (colecção)
	/broadcasts/{id}/audio_assets/{id} (recurso)
Elementos	*Name* (*String*)
	Description (*String*)
	Creation Date (*Date*)
	Format (*String*)
	Length (*Integer*)
	MD5 (*String*)
	Programs (*Resource URLs*)
	Broadcasts (*Resource URLs*)

4.6 *Radiastore Daemon*

4.6.1 Descrição

Este componente tem o objectivo de gerir a replicação dos Ficheiros existentes no sistema e efectuar a manutenção do mesmo. É um componente que pretende obter a maior independência possível da gestão das entidades apresentadas na secção anterior (secção 4.5) porque implica que as suas operações sejam efectuadas de forma automática, possibilitando a menor intervenção humana possível. Para além disso, per-

mite que seja substituído por qualquer outro componente de gestão de replicação de Ficheiros ou que realize quaisquer operações de manutenção do sistema, respeitando a definição das entidades utilizadas.
O Modelo de Domínio deste componente encontra-se no Apêndice B (Figura B.2), representando os elementos de domínio e as respectivas relações.

4.6.2 Estrutura

Devido aos requisitos impostos, o componente *Radiastore Daemon* terá de apresentar uma estrutura que permita o seu funcionamento por tempo indeterminado, uma vez que é necessário o controlo e gestão permanente dos Ficheiros. É preciso que este componente tenha acesso ao repositório de dados do componente Web *Radiastore* uma vez que depende disso para efectuar as operações necessárias. Esta dependência implica a sincronização das entidades partilhas pelos dois componentes (Web *Radiastore* e *Radiastore Daemon*) sendo inevitável o seu alinhamento quando são efectuadas alterações a uma destas entidades partilhas.

4.6.2.1 Entidades

A entidade **Ficheiro** já foi definida no componente anterior, na secção 4.5.2.1, sendo assim uma entidade partilhada entre os dois componentes.

Um **Sistema de Armazenamento** (*Replica System*) é considerado um sistema computacional com poder de armazenamento de conteúdos digitais. Para um Sistema de Armazenamento ser definido no sistema é necessário a sua localização, protocolo de rede utilizado, localização interna onde se armazenam os conteúdos multimédia e caso seja necessário os dados para autenticação (nome de utilizador e palavra-passe). A definição é feita pelo Administrador no componente Web *Radiastore* onde para além dos elementos já descritos pode associar um nome ao Sistema de Armazenamento, e caso ainda não exista, indicar se é o Sistema de Armazenamento Principal. A Figura 4.4 é um exemplo da definição de um Sistema de Armazenamento.
Não existe qualquer diferença entre o Sistema de Armazenamento Principal e os restantes Sistemas de Armazenamento. O sistema apenas sabe que determinado Sistema de Armazenamento é o Principal.

Figura 4.4: Exemplo de um Sistema de Armazenamento com as respectivas propriedades

Uma **Réplica** (*Replica*) é definida como sendo a representação de um Ficheiro num Sistema de Armazenamento.
A criação de uma Réplica ocorre em duas situações:

1. Quando um Ficheiro é adicionado pelo respectivo Autor ao sistema (implicando a adição do Ficheiro ao Sistema de Armazenamento Principal), sendo criada a Réplica principal;
2. Quando o Ficheiro é marcado (automaticamente pelo sistema ou por indicação do Administrador) para ser replicado num Sistema de Armazenamento.

Independentemente de um Ficheiro estar no Sistema de Armazenamento Principal, ou noutro Sistema de Armazenamento, é considerado como sendo uma Réplica. Isto permite tratar de igual forma os Ficheiros em todos os Sistemas de Armazenamento onde se encontram replicados.

O **Examinador** (*Fetcher*) indica que determinado Ficheiro tem de ser replicado, recuperado ou removido num Sistema de Armazenamento. São definidos três tipos de Examinadores:

1. **Examinador de replicação**: contém informação sobre a replicação de um Ficheiro num Sistema de Armazenamento;
2. **Examinador de recuperação**: através do qual se recupera um Ficheiro para o Sistema de Armazenamento Principal;
3. **Examinador de remoção**: contém informação para a remoção de um Ficheiro num Sistema de Armazenamento.

Estes Examinadores obtêm os dados necessários do Sistema de Armazenamento onde a operação (replicação, recuperação ou remoção de um Ficheiro) vai ser efectuada. Isto vai implicar uma ligação, através do protocolo do Sistema de Armazenamento, para a transferência de Ficheiros (ou execução de operações, no caso da remoção) entre o Sistema de Armazenamento Principal e o Sistema de Armazenamento onde se replica, recupera ou remove o Ficheiro.

4.6.3 Processo de Replicação

O processo de replicação ocorre entre o Sistema de Armazenamento Principal e um Sistema de Armazenamento secundário onde é efectuada uma nova Réplica de um Ficheiro. Existem duas fases para este processo ser efectuado. Em primeiro lugar, o Administrador escolhe o modo de replicação para os Ficheiros e numa segunda fase o componente *Radiastore Daemon* efectua a replicação do Ficheiro.
Os modos de replicação disponíveis são:

1. A nível do Ficheiro, onde o Administrador pode escolher em que Sistemas de Armazenamento o Ficheiro é replicado;
2. A nível do Programa, onde o Administrador define em que Sistemas de Armazenamento os Ficheiros associados a determinado Programa são replicados.

Dependendo do modo de replicação escolhido, os Ficheiros vão sendo marcados para replicação. No caso de o Administrador escolher replicar Ficheiros pelo primeiro modo (onde o Administrador diz explicitamente que um Ficheiro é para ser replicado em determinado Sistema de Armazenamento), é criado de imediato um

Examinador de replicação para o Ficheiro.

Caso contrário, quando o Administrador indica que os Ficheiros associados a determinado Programa são para ser replicados, o Examinador de replicação é criado apenas quando o Ficheiro é adicionado ao Sistema. A única diferença é que no primeiro caso o Ficheiro já se encontra no sistema, enquanto que no segundo caso, conforme os Ficheiros são adicionados ao sistema também vão sendo marcados para replicação.

Com os Examinadores de replicação criados, o componente *Radiastore Daemon* adquire-os e procede em

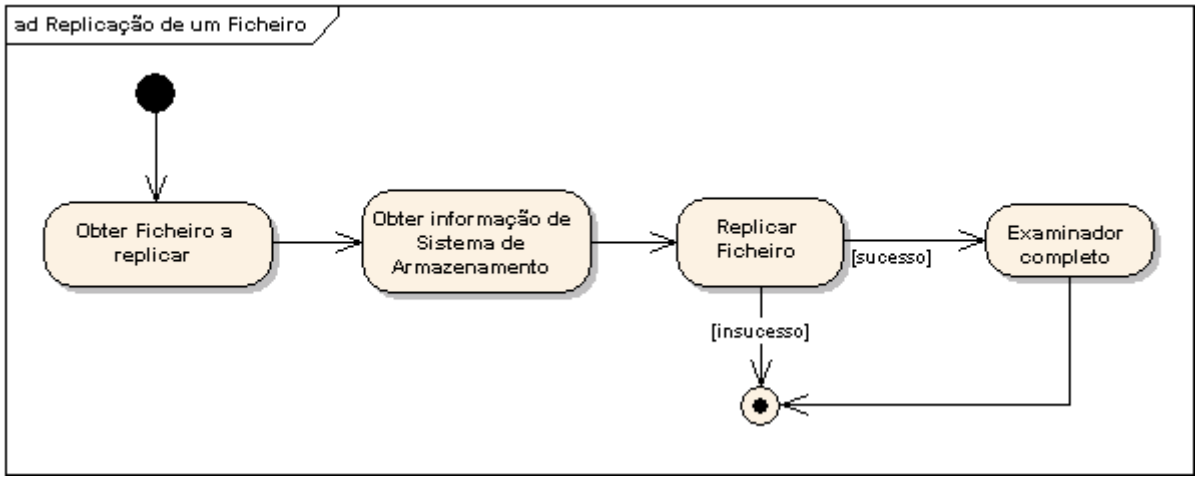

Figura 4.5: Diagrama de estados da replicação de um Ficheiro

conformidade. Por cada Examinador de replicação, obtém os dados do Sistema de Armazenamento onde o Ficheiro vai ser replicado e inicia o processo de transferência do Ficheiro do Sistema de Armazenamento Principal para o Sistema de Armazenamento especificado no Examinador. No caso da transferência ser efectuada com sucesso, o Examinador altera o seu estado para indicar que a replicação já foi efectuada. Caso contrário, o Examinador não altera o seu estado e o processo repetir-se-á novamente.

O processo de replicação escolhido é simples, e assume que os Sistemas de Armazenamento têm capacidades suficientes para receber as Réplicas dos Ficheiros. Podem ser adaptados processos de replicação mais complexos, permitindo uma melhor gestão de espaço nos Sistemas de Armazenamento e uma maior distribuição das Réplicas nos Sistemas de Armazenamento, no caso de apresentarem restrições que sejam um inconveniente para o processo apresentado.

4.6.4 Processo de Recuperação

O processo de recuperação serve para recuperar Ficheiros, através das suas Réplicas, quando se detecta a falta deles no Sistema de Armazenamento Principal. Este processo é opcional podendo o Administrador de Sistema decidir que não é necessário ter todos os Ficheiros da rádio no Sistema de Armazenamento Principal.

A detecção da falta de Ficheiros no Sistema de Armazenamento é feita de forma automática pelo componente *Radiastore Daemon*. Ao detectar essa falta cria um Examinador de recuperação que será tratado e

recuperar o Ficheiro num dos Sistemas de Armazenamento que contenha uma Réplica. No caso de não existir nenhuma Réplica, o Ficheiro é considerado como inacessível não podendo ser recuperado.

Quando existem Examinadores de recuperação para serem tratados, por cada um deles são adquiridos

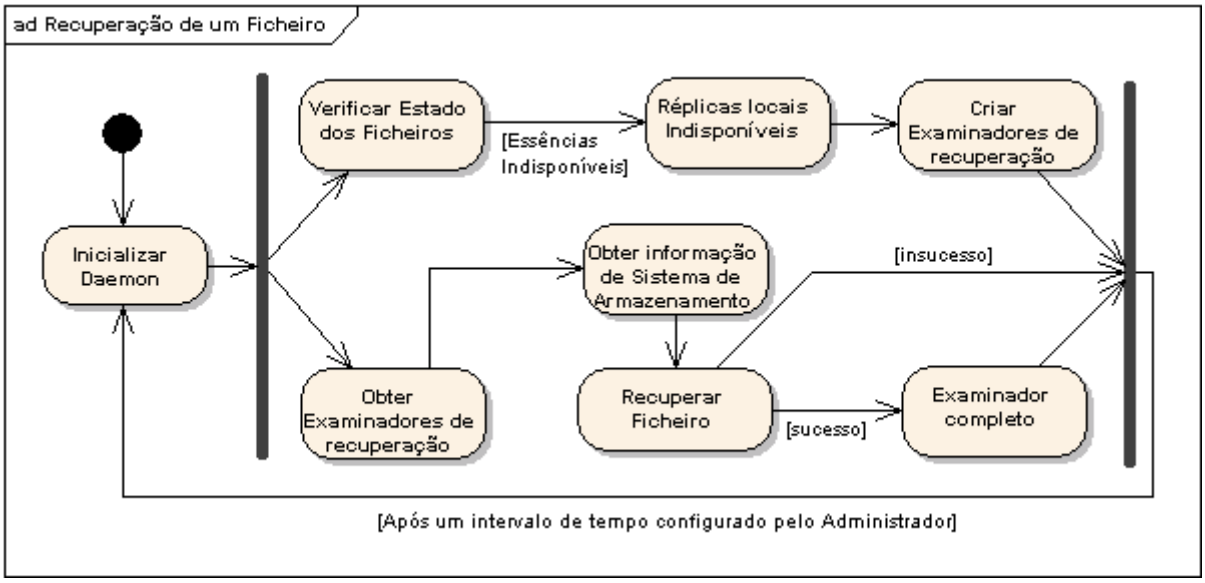

Figura 4.6: Recuperação de um Ficheiro

os dados do Sistema de Armazenamento da Réplica do Ficheiro a recuperar e é iniciada a transferência da Réplica, desse Sistema de Armazenamento para o Sistema de Armazenamento Principal, actualizando o estado do Ficheiro como disponível e o estado do Examinador para informar que o processo foi concluído com sucesso. No caso de a transferência falhar, o mesmo processo é tentado novamente mais tarde pelo componente *Radiastore Daemon*.

Outra situação que pode ocorrer, e resultar na criação de um Examinador de recuperação, é quando se pretende fazer o download da Essência (ou documento de Metadados) de um Ficheiro e não se encontra disponível. Nesta situação, pode-se proceder da seguinte forma:

1. Criar um Examinador de recuperação e informar que o Ficheiro não se encontra disponível de momento. Esta situação criará um inconveniente para o actor que pretende fazer o download do Ficheiro, e implica que o Ficheiro só se encontre disponível mais tarde;

2. Redireccionar o pedido para o Sistema de Armazenamento onde se encontra uma Réplica do Ficheiro em falta. Esta solução é mais viável que a primeira. No entanto, podem ocorrer situações em que o actor não pode aceder directamente ao Sistema de Armazenamento onde se encontra a Réplica (caso do Sistema de Armazenamento ser uma máquina local, com partilha local), sendo a única solução a primeira apresentada.

4.6.5 Processo de Remoção

Este processo serve para remover Ficheiros do Sistema de Armazenamento Principal, se possuírem data de validade e essa data de validade tenha expirado. O componente *Radiastore Daemon* detecta automaticamente os Ficheiros com data de validade expirada e cria para cada um deles os respectivos Examinadores de remoção (Figura 4.7). Numa outra fase, esse componente obtém esses Examinadores e por cada um deles, são obtidos os dados do Sistema de Armazenamento onde a Réplica será removida (neste caso, o Sistema de Armazenamento Principal) e procede-se à remoção do Ficheiro. No caso do processo ser efectuado com sucesso, o Examinador é também removido. Caso contrário, o estado do Examinador mantém-se e o processo repetir-se-á quando o componente *Radiastore Daemon* efectuar este processo novamente.
Como o Examinador tem a possibilidade de saber qual o Sistema de Armazenamento onde se encontra o Ficheiro a remover, o processo de remoção pode ser facilmente estendido para suportar remoção das Réplicas dos Ficheiros, caso se pretenda replicar também Ficheiros temporários.
Este processo implica também remover qualquer associação entre os Ficheiros removidos e outras entidades presentes no sistema.

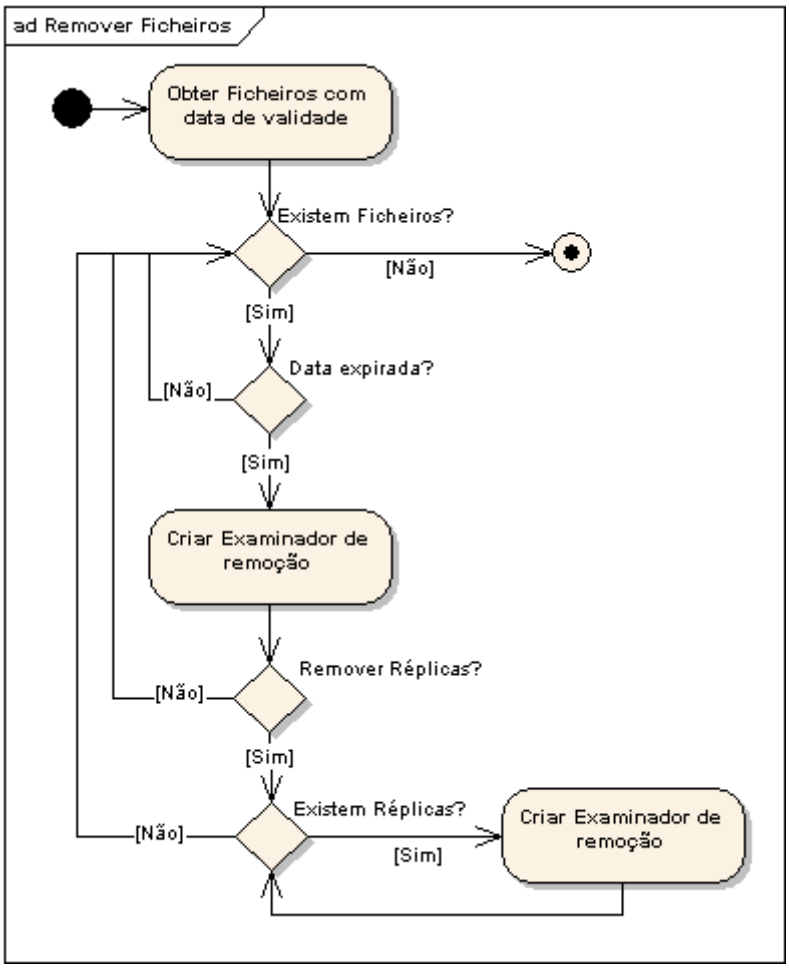

Figura 4.7: Detecção de Ficheiros a remover do sistema

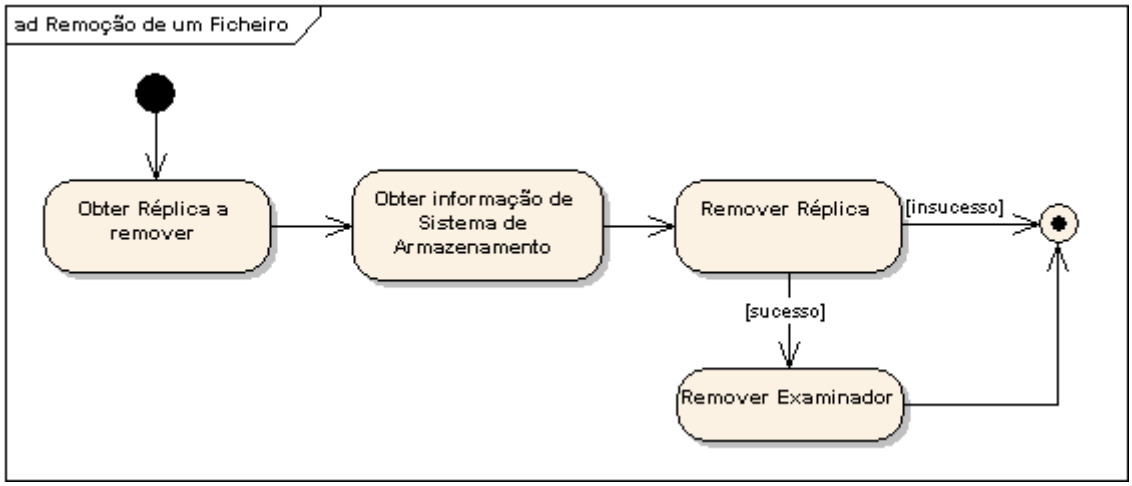

Figura 4.8: Diagrama de estados da remoção de uma Réplica

4.7 Metodologia de Desenvolvimento

O desenvolvimento da solução foi feito de modo incremental, alinhando a especificação da Proposta da Solução com a Implementação dos dois componentes apresentados. Para tal contribuiu a análise ao funcionamento da Rádio Zero e da Rádio Universitária do Algarve[8], permitindo extrair dessa análise o contributo para a especificação do problema.

[8] http://www.rua.pt

5 Implementação

5.1 Introdução

Neste capítulo será apresentada a implementação baseada na Proposta da Solução do capítulo 4. Serão descritas as escolhas tecnológicas feitas para implementar os componentes Aplicação Web *Radiastore* e *Radiastore Daemon*. Nem todas as funcionalidades descritas no capítulo da Proposta da Solução foram implementadas. Essas funcionalidades acrescentam valor ao sistema mas não impedem o correcto funcionamento do mesmo para garantir a satisfação dos requisitos mais cruciais para a resolução do problema.

5.2 Tecnologia utilizada

5.2.1 *Ruby*

Ruby [1] é uma linguagem de programação orientada a objectos, onde tudo o que é manipulado, e o que resulta dessas manipulações, é um objecto. Foi criada por *Yukihiro Matsumoto*, com o primeiro lançamento público em 1995, e resulta da combinação das características de outras linguagens de programação já existentes (entre elas a linguagem de programação *Perl*, *Smalltalk* e *Lisp*). As suas principais características são o suporte a diversos paradigmas de programação (e.g. orientado a objectos, funcional e imperativo), gestão de memória automática (incluindo *garbage collector*), é *open-source*, é multi-plataforma e é uma linguagem interpretada.

5.2.2 *Ruby on Rails*

Ruby on Rails [13] (também denominada de *Rails*) é uma *framework* que permite criar aplicações Web. Tem suporte para o seu desenvolvimento, instalação e manutenção fornecendo as ferramentas necessárias para isso. Esta *framework* foi criada por *David Heinemeier Hansson*, com o primeiro lançamento público em 2004 tornando-se completamente *open-source* apenas em 2005. Surgiu da necessidade de criar uma tecnologia fácil de usar relativamente às tecnologias existentes (e.g. Java, PHP, .NET). As aplicações *Rails* são todas escritas em *Ruby* o que permite escrever aplicações facilmente (devido às potencialidades desta linguagem de programação) e de fácil leitura. A adopção de dois conceitos permite aos programadores criarem aplicações eficientes:

- **DRY** (*Don't Repeat Yourlsef*), que significa que não nos devemos repetir continuamente, e que cada porção de conhecimento num sistema deve estar apenas num único sítio. Este conceito evita a duplicação de funcionalidades nas aplicações, dizendo o que se pretende em determinado sítio e reutilizado quando for necessário;
- **Convenção além de configuração**. Seguindo as convenções aconselhadas é possível escrever uma aplicação com muito menos código do que uma aplicação que usa configurações (e.g. aplicação Web em Java com configurações em XML).

A *framework Rails* utiliza o padrão arquitectural Modelo-Vista-Controlador, descrito na secção 4.2.2. O programador apenas escreve a funcionalidade da aplicação nos modelos, controladores e vistas e a *framework*

trata de os interligar durante a execução do programa, através de um processo que lhe está inerente e não necessita de nenhuma configuração por parte do utilizador. Cada aplicação é englobada num conjunto de directorias (Figura 5.1) previamente criadas através de um simples comando. Os componentes do padrão arquitectural utilizado são colocados na directoria *app* possuindo cada um deles a respectiva directoria. Este é um exemplo da filosofia desta *framework* que favorece as convenções para além das configurações.

A execução de um pedido é descrito na Figura 5.2 e segue os seguintes passos:

Figura 5.1: Estrutura das directorias utilizada pela *framework Ruby on Rails*

1. O pedido é feito através *browser* e o componente *Routing* separa o URL do pedido para obter o respectivo Controlador e método (denominado de acção) a executar;
2. O *Routing* redirecciona o pedido para o Controlador e acção correspondente;
3. Para a acção ser executa pode ser necessária a consulta ao Modelo de Dados e consulta à Base de Dados utilizada. Ao fim de a acção ser executada, a resposta é redireccionada para o Controlador;
4. O Controlador fornece à Vista a resposta do pedido invocado inicialmente;
5. A Vista constrói a página em HTML que será posteriormente processada pelo *browser*.

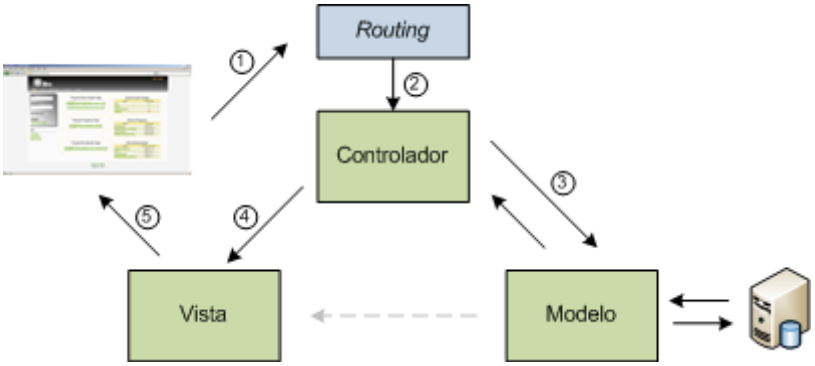

Figura 5.2: *Rails* e a utilização do Modelo-Vista-Controlador

Uma das principais vantagens desta *framework* é a independência entre o sistema de gestão de base de dados utilizado e o mapeamento nos objectos do Modelo, do padrão utilizado. O utilizador não tem que escrever nenhum código SQL especificando apenas o sistema de gestão utilizado.
A geração de páginas HTML dinâmicas é feita através de uma ferramenta chamada ERb (*Embedded Ruby*) que consiste em colocar pedaços de código *Ruby* combinado com código HTML e executado quando os pedidos são solicitados ao servidor.

A utilização desta *framework* facilita também a implementação de serviços Web utilizando o padrão REST,

que permite mapear os recursos disponíveis nos controladores da aplicação. Respeitando a convenção DRY (*Don't Repeat Yourself*) é possível disponibilizar uma representação (e.g. em XML) para os pedidos do serviço Web utilizando o mesmo código que gera a representação (páginas HTML) para os utilizadores finais da aplicação. Aproveitando os comandos do HTTP (*GET*, *PUT*, *POST* e *DELETE*), o REST consegue utilizá-los para efectuar as operações básicas sobre um recurso: criar, ler, actualizar e eliminar o recurso. Esses comandos são posteriormente mapeados para acções particulares dos Controladores:

- Para criar um recurso, o comando HTTP utilizado é o ***PUT*** e a acção correspondente nos Controladores é ***create***;
- Para a consulta (leitura) de recursos, o comando HTTP correspondente é o ***GET*** e a respectiva acção nos Controladores é ***show***;
- O comando HTTP ***POST*** permite a actualização de recursos correspondendo à acção ***update*** nos Controladores;
- Para eliminar um recurso é utilizado o comando ***DELETE*** que corresponde à acção ***delete*** nos Controladores.

5.3 Sistema de Armazenamento Principal

Esta secção apresenta uma especificação ao requisito não-funcional número sete (RNF 7) da secção 3.2.4. O valor apresentado para a capacidade do Sistema de Armazenamento Principal é baseado em vários factores da Rádio Zero. Deste modo, a capacidade do Sistema de Armazenamento Principal tem em conta os seguintes factores:

- Semanalmente são gerados **setenta e quatro Ficheiros** [11] que correspondem às Emissões dos sessenta e seis Programas existentes na Rádio Zero;
- Cada Emissão tem apenas um Ficheiro;
- Cada Ficheiro tem em média **setenta e oito *Megabytes*** de tamanho (pela regras de normalização impostas pela rádio);
- Existem, aproximadamente, **quatro semanas por mês**.

São necessários, aproximadamente, 277056 *Megabytes* (aproximadamente 271 *Gigabytes*) por ano.

$$74 Ficheiros/Semana \cdot 78 MB/Ficheiro \cdot 4 Semana/Mês \cdot 12 Meses/Ano = 277056 MB/Ano$$

Num espaço de cinco anos, chega-se à conclusão que são necessários cerca de 1353 *Gigabytes* (aproximadamente 1,3 *Terabytes*) para dar suporte ao armazenamento de Ficheiros da rádio.

5.4 Aplicação Web *Radiastore*

Para implementar este componente foi utilizada a *framework Ruby on Rails* (secção 5.2.2). Desta forma, o componente contém dez classes de Modelos da aplicação e treze classes de Controladores. As Vistas (que originam a interface Web) foram implementadas na linguagem HTML combinadas com código *Ruby*. Todo o código e elementos de apoio à aplicação (*plugins*, documentos *javascript*, CSS, imagens, testes,

configurações da base de dados e *logs*) encontra-se distribuído pela estrutura de directorias que a *framework* utiliza.

5.4.1 Interface Web

A interface Web é a parte mais importante da implementação, uma vez que se pretende fornecer a maior quantidade de informação possível sobre os Ficheiros, Programas, Emissões e outras entidades do sistema modelado, para além de ser o componente que interage com os actores.

A página inicial (Figura 5.3) da Aplicação Web *Radiastore* fornece informação sobre o histórico do sistema. Disponibiliza informação sobre as entidades (Ficheiros, Programas e Emissões) mais categorizadas/consultadas do sistema pelos actores e também as entidades mais recentes do sistema, incentivando os actores a consultarem o sistema a partir deste ponto.
Também permite a um Actor Registado autenticar-se no sistema. Aqueles que pretendem registar-se podem fazê-lo através da hiperligação disponível no canto superior direito, preenchendo um formulário onde são solicitados os dados necessários para o registo. Este processo fica concluído com o envio de um e-mail para o actor onde confirma a sua subscrição.

A pesquisa é efectuada exclusivamente em Ficheiros, Programas ou em Emissões onde o actor escolhe

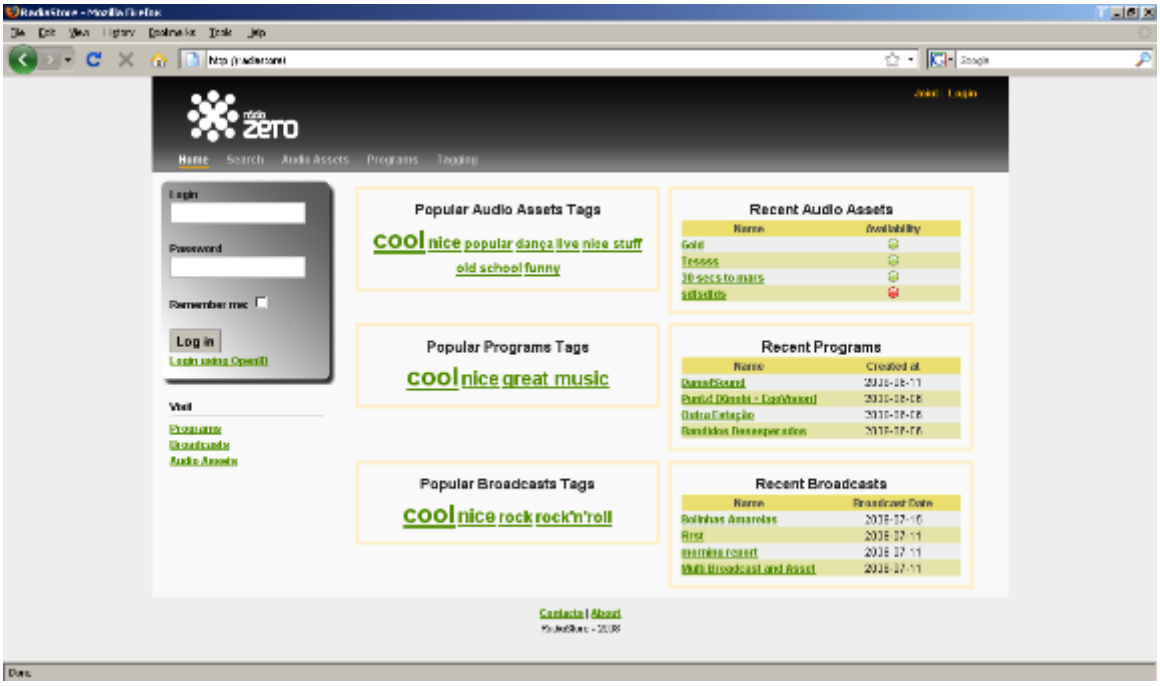

Figura 5.3: Página inicial da Aplicação Web *Radiastore*

a entidade a pesquisar como se pode verificar na Figura 5.4. O actor pode efectuar a pesquisa de duas formas, como foi descrito na secção 4.5.5 da Proposta da Solução. A primeira opção é inserir as palavras-chave relacionadas com as entidades (e.g. nome dos Ficheiros, palavras na descrição de um Programa) na caixa de texto disponível. Estas palavras-chave são procuradas nos campos das entidades, que estão definidos

directamente nos métodos de procura das respectivas entidades.

A outra opção é alterar o modo de pesquisa (que é possível através da hiperligação disponível no menu

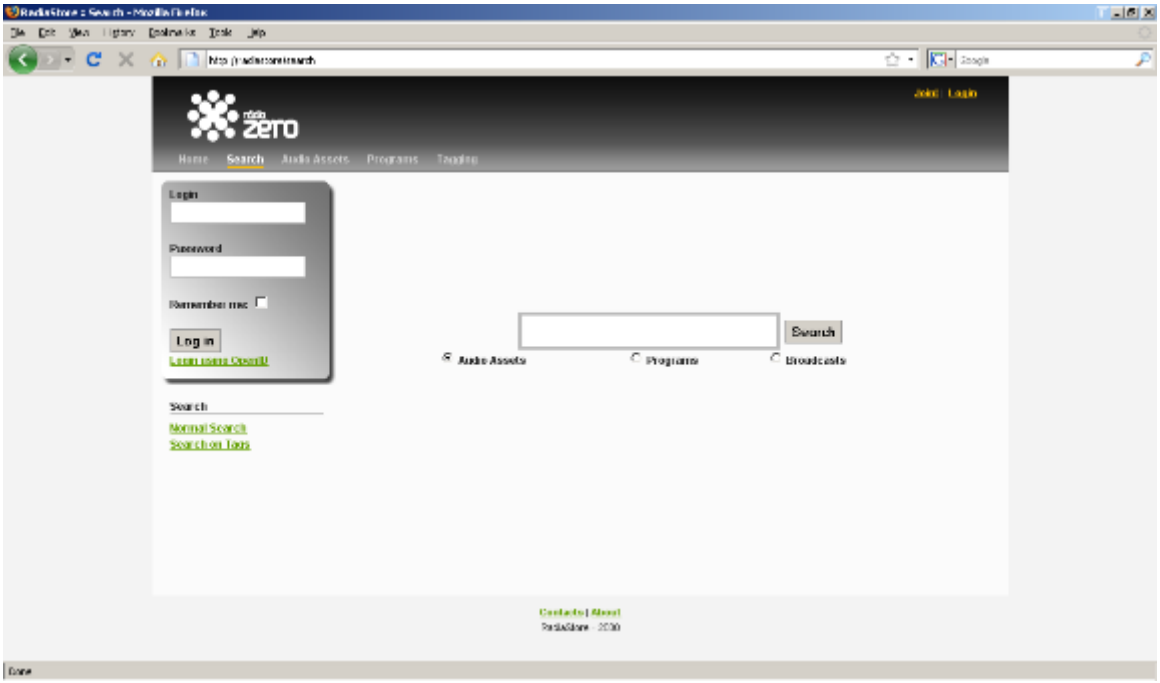

Figura 5.4: Interface para a pesquisa de Entidades

lateral da página) para a pesquisa ser feita nas Tags associadas aos Ficheiros, Programas e Emissões. O actor apenas tem de escolher a entidade e inserir a Tag que pretende pesquisar.

Devido à flexibilidade da *framework Ruby on Rails* é fácil de adicionar novas entidades para serem pesquisadas. Foi definido um controlador único que trata da pesquisa, onde apenas é necessário indicar o modelo e as palavras da pesquisa sendo retornados os resultados em função disso. Por exemplo, se surgir um requisito onde é necessário pesquisar Autores, a entidade Autor (mais especificamente o seu modelo) apenas tem de implementar um método que recebe as palavras-chave da pesquisa e retorna os Autores que nos parâmetros de pesquisa contenham essas palavras, ficando o resto do processamento (e.g. apresentação de resultados) a cargo do controlador que trata da pesquisa.

Os Ficheiros e os Programas encontram-se disponíveis através do menu disponível para os actores podendo consultá-los livremente e ver a informação de cada entidade em particular.

A listagem de Programas apresenta uma estrutura hierárquica onde se pode ver por cada Programa as Emissões que o compõem, e por cada Emissão os Ficheiros que lhe estão associados. Este paradigma facilita a navegação por parte dos actores criando uma analogia com a organização do sistema de ficheiros que é normalmente utilizado nos Sistemas de Informação deste tipo de rádios (e.g. Sistema de Informação actual da Rádio Zero). Na informação de um Programa pode-se encontrar os dados associados (nome, descrição, data de criação e Autores do Programa), as Emissões que fazem parte do Programa, um espaço para associar Tags e visualizar as que já foram associadas por meio de uma *Tag Cloud* - caso os privilégios do actor o permitam.

Os dados disponíveis na informação de um Ficheiro são o nome, descrição, formato, género, tamanho (em *Megabytes*), número de vezes que foi feito o download, duração, data de validade (se existir), *checksum* MD5 e a disponibilidade do Ficheiro, que é indicada por um símbolo verde no caso de estar disponível no Sistema de Armazenamento Principal ou por um símbolo vermelho no caso de estar indisponível. A informação de um Ficheiro apresenta também as Emissões e Programas a que o Ficheiro pertence. É possível associar Tags ao Ficheiro e ver as que já foram associadas através de uma *Tag Cloud*, se os privilégios do actor permitirem tais acções.
Na informação de uma Emissão (acedidas através dos Programas ou pela informação dos Ficheiros) podemos consultar o seu nome, descrição, idioma, género, data em que foi emitida, os Ficheiros que a compõem e o espaço para associar e visualizar Tags como as duas entidades já descritas.

Os Autores quando acedem ao sistema têm um espaço próprio onde podem gerir as suas autorias. Têm acesso aos Programas e Ficheiros dos quais são Autores, alternando as respectivas vistas pelo menu lateral que têm à disposição. Nos Programas onde um actor tem o privilégio de Autor pode associar outros actores do sistema como Autores desse Programa e alterar qualquer outro tipo de informação associado ao Programa. Esse privilégio também lhe permite associar Autores desse Programa a novos Ficheiros que adiciona ao sistema.

Os actores podem consultar Ficheiros, Emissões e Programas através de uma *Tag Cloud* (Figura 5.5) que contém todas as Tags associadas a cada entidade, podendo alternar entre as *Tag Clouds* de cada entidade através do menu que se encontra do lado esquerdo.
O tamanho de letra de cada Tag depende da frequência com que a Tag aparece associada a uma entidade. Os actores podem obter Ficheiros, Programas ou Emissões de uma forma mais eficaz e rápida consultando esta funcionalidade.

A inserção de um Ficheiro no sistema é feita através da interface da Figura 5.6. A interface é simples e permite inserir os dados básicos relativos a um Ficheiro focando-se em captar a atenção do Autor tornando esta operação o menos aborrecida possível para esta classe de actor. Os dados solicitados são o nome, descrição, formato, género, a data de validade (se for um Ficheiro temporário no sistema) e a associação de outros Autores, no caso de terem contribuído para a produção do Ficheiro. Para além destes dados, têm de ser associados a Essência e opcionalmente o documento de Metadados que descreve o Ficheiro. Todos os outros dados que compõem o Ficheiro, presentes na respectiva informação quando criados, são associados ao Ficheiro pela Aplicação Web *Radiastore* (e.g. tamanho do Ficheiro, *checksum* MD5). Um caso especial desta associação de dados automática é o nome que é dado a cada Essência e documento de Metadados, que é diferente do nome do Ficheiro em si. O nome destes elementos é gerado automaticamente através de Identificadores Únicos e Universais (UUID[1]). Este tipo de identificadores é bastante utilizado para evitar a colisão de nomes, devido à precisão da sua unicidade e à baixa probabilidade de haver dois UUIDs iguais. Exemplo do nome de uma Essência (um ficheiro áudio mp3, por exemplo) pode ser

[1] *Universally Unique Identifier*: http://www.ietf.org/rfc/rfc4122.txt

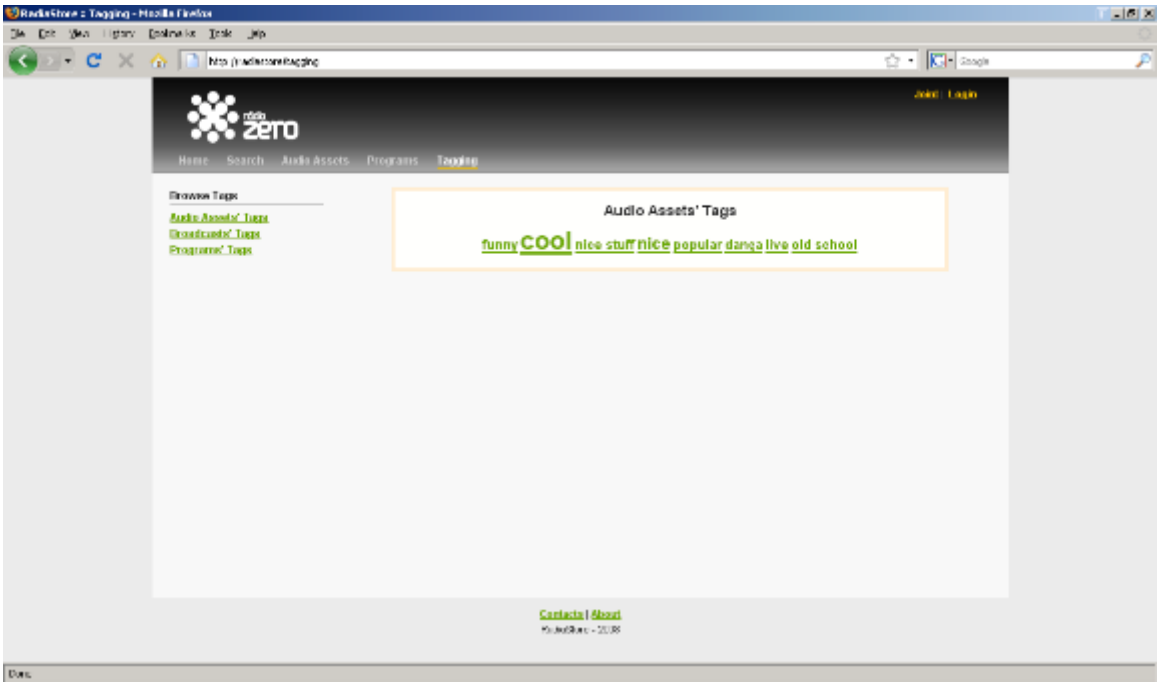

Figura 5.5: *Tag Cloud* associada aos Ficheiros

bd4fb081325a012b097f0018dea09a22.mp3. No caso de possuir um documento de Metadados (documento em XML) associado, o nome seria *bd4fb081325a012b097f0018dea09a22.xml*.

A gestão dos Sistemas de Armazenamento, a replicação dos Ficheiros e a gestão de Programas cujos

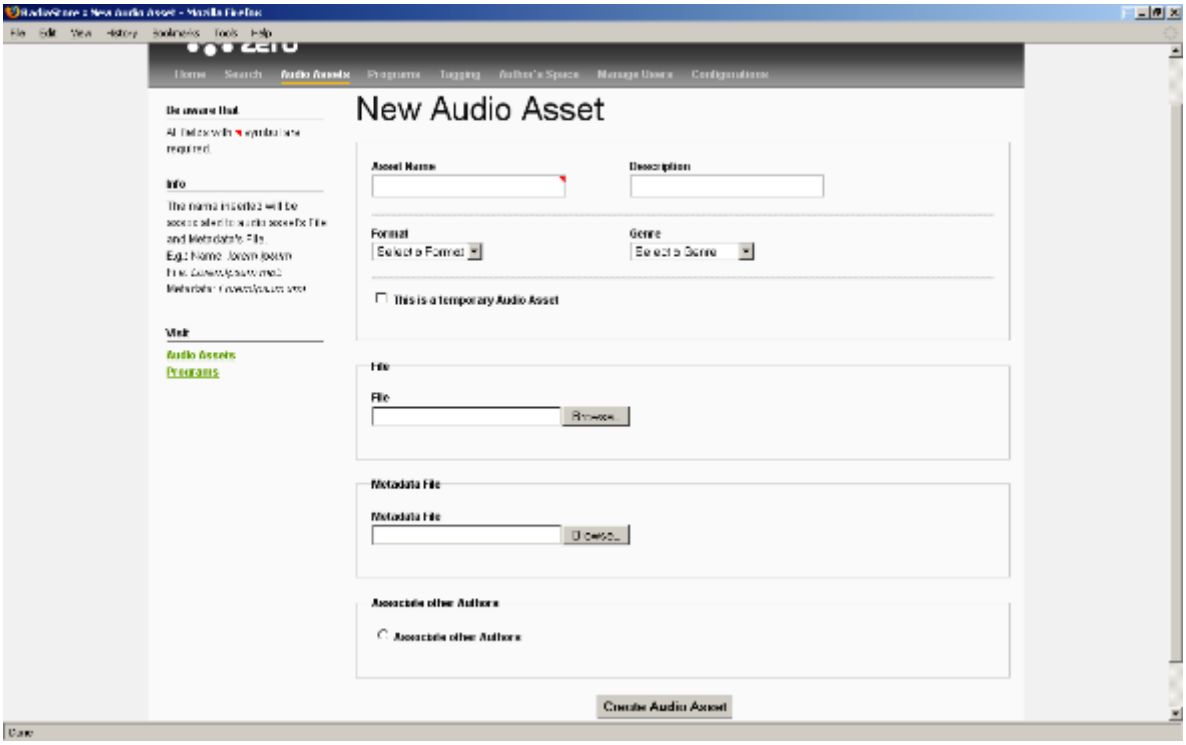

Figura 5.6: Interface para criação de um Ficheiro

Ficheiros são para replicar, apenas está ao alcance do Administrador. São disponibilizadas as funcionalidades que permitem adicionar novos Sistemas de Armazenamento e a respectiva alteração de parâmetros associados (e.g. localização do Sistema de Armazenamento, alteração da autenticação). Na informação de cada Sistema de Armazenamento (excepto no Principal) é possível consultar os Ficheiros que estão replicados nesse Sistema de Armazenamento e consultar os Programas assinalados para que os respectivos Ficheiros (quando adicionados ao sistema) sejam replicados, podendo o Administrador replicar os que não se encontram replicados, e eliminar do Sistema de Armazenamento o que se encontra replicado.
Por outro lado, na informação de cada Ficheiro e Programa é possível verificar em que Sistemas de Armazenamento estão replicados, como podemos ver o exemplo da Figura 5.7, onde é indicado a vermelho os Sistemas de Armazenamento que possuem uma Réplica do Ficheiro.

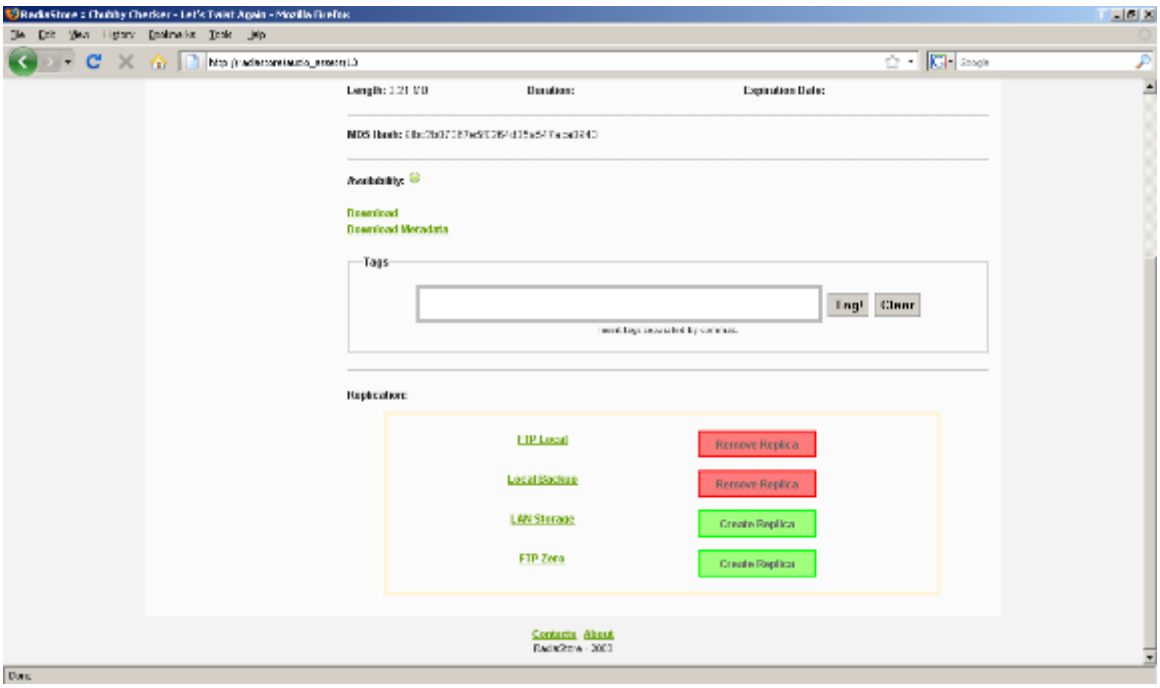

Figura 5.7: Gestão da replicação de um Ficheiro

5.4.2 Recursos

Nesta secção são apresentados os recursos disponíveis através do componente Aplicação Web *Radiastore* descritos na secção 4.5.6. Estes recursos são uma fonte de informação, especialmente para os sistemas de informação que os aproveitam para criarem funcionalidades e gerar novo conhecimento. Exemplo disso, é a futura integração e alinhamento entre este sistema e o trabalho realizado paralelamente por D. Z. Silva [11].

As principais representações para estes recursos são em HTML e XML. É possível ter diversas representações destes recursos juntando a vantagem do padrão arquitectural REST em abstrair-se da implementação de cada componente que possui (como foi referido na secção 4.3) e as funcionalidades dos Controladores e Modelos do padrão MVC que a *framework Ruby on Rails* utiliza. Nem todos os verbos HTTP são imple-

mentados nestes recursos (como se pode ver pela Tabela 5.1) uma vez que não implicam uma barreira à inutilização do sistema e em alguns deles devido às restrições impostas pelo próprio sistema (e.g. não se pode alterar a informação associada a um Ficheiro ou Emissão após serem criados).

Tabela 5.1: Recursos disponíveis na Aplicação Web *Radiastore*

	Programas (*Programs*)	**Emissões** (*Broadcasts*)	**Ficheiros** (*Audio Assets*)
POST	HTML, XML	HTML, XML	HTML, XML
GET	HTML, XML	HTML, XML	HTML, XML
PUT	HTML, XML	-	-
DELETE	-	-	-

5.4.3 Comparação com a Proposta da Solução

De seguida é apresentada uma tabela indicando o que foi implementado relativamente ao que foi modelado na Proposta da Solução. Permite obter uma visão de síntese do que foi feito na implementação deixando as funcionalidades não implementadas para trabalho futuro. Cada funcionalidade será classificada como **Sim** se estiver implementada, **Parcial** se estiver parcialmente implementada e **Não** se apenas estiver modelada.
A funcionalidade da Pesquisa por palavras-chave livres encontra-se parcialmente implementada devido ao facto de não apresentar os campos da entidade seleccionada, para que o actor possa escolher sobre que campos incide a pesquisa. Quanto à funcionalidade da Pesquisa por Tags encontra-se restringida a pesquisas com apenas uma Tag.
Os Recursos Web encontram-se parcialmente implementados devido às razões já descritas na secção 5.4.2 deste capítulo.

5.5 *Radiastore Daemon*

Para a implementação deste componente foi utilizada a linguagem de programação *Ruby* devido à interacção que este componente apresenta com a Aplicação Web *Radiastore*. Para além disso permite aproveitar certas funcionalidades já implementadas nos Modelos da arquitectura utilizada por esse componente.
A implementação do componente *Radiastore Daemon* resulta num conjunto de oito classes organizadas segundo o padrão arquitectural Modelo de Domínio (descrito na secção 4.2.2). Este componente é executado através de uma consola de linha de comandos, permitindo obter um relatório simples de actividade ao longo do tempo (possuindo um *log* mais detalhado) e terminar a aplicação sempre que o Administrador de Sistema necessite.

5.5.1 Fluxo de Execução

Sempre que o componente *Radiastore Daemon* é inicializado são lançadas três tarefas através de um gestor de processos, uma por cada processo descrito na secção 4.6: processo de replicação, processo de recuperação e processo de remoção. Os lançamentos destas tarefas são feitas em intervalos de tempo diferentes,

Tabela 5.2: Funcionalidades implementadas na Aplicação Web *Radiastore*

Funcionalidade	Implementada
Ficheiros	
Criar Ficheiros	**Sim**
Analisar documento de Metadados e associar dados ao Ficheiro	**Não**
Autoria de Ficheiros	**Sim**
Associar Ficheiros a Emissões	**Sim**
Associar Comentários	**Não**
Gerir Replicação de Ficheiros	**Sim**
Emissões	
Criar Emissões	**Sim**
Definir Ocorrências	**Não**
Construção de grelha	**Não**
Associar Emissões a Programas	**Sim**
Programas	
Criar Programas	**Sim**
Autoria de Programas	**Sim**
Gestão de Replicação	**Sim**
Processo de *Tagging*	
Associar Tags a Ficheiros/*Tag Cloud*	**Sim**
Associar Tags a Programas/*Tag Cloud*	**Sim**
Associar Tags a Emissões/*Tag Cloud*	**Sim**
Associar Tags a em partes diversas dos Ficheiros	**Não**
Pesquisa	
Pesquisa de Ficheiros através de palavras-chave livres	**Parcial**
Pesquisa de Programas através de palavras-chave livres	**Parcial**
Pesquisa de Emissões através de palavras-chave livres	**Parcial**
Pesquisa de Ficheiros através de Tags	**Parcial**
Pesquisa de Programas através de Tags	**Parcial**
Pesquisa de Emissões através de Tags	**Parcial**
Recursos	
Programas	**Parcial**
Emissões	**Parcial**
Ficheiros	**Parcial**
Gestão de Sistemas de Armazenamento	**Sim**
Gestão de Autorias (Programas e Ficheiros) por Autor	**Sim**

de modo a distribuir a carga de consulta ao sistema, sendo esses intervalos de tempo configuráveis pelo Administrador de Sistema.

Em cada intervalo de tempo cada processo efectua o respectivo trabalho:

- **Processo de Replicação**: verifica se existe algum Ficheiro para replicar e em caso afirmativo cria a respectiva Réplica e transfere os componentes do Ficheiro - Essência e documento Metadados, caso exista - para o Sistema de Armazenamento especificado;
- **Processo de Recuperação**: cria um Examinador de recuperação por cada Ficheiro que se encontre ausente (a inexistência de Essência) no Sistema de Armazenamento Principal, e numa segunda fase

trata de o recuperar de um Sistema de Armazenamento de uma das suas Réplicas[2], no caso de existir alguma. Caso contrário, o Ficheiro não pode ser recuperado;

- **Processo de Remoção**: obtém os Ficheiros com data de validade expirada, cria para cada um deles um Examinador de remoção e remove-os do Sistema de Armazenamento Principal logo de seguida.

5.5.2 Extensibilidade aos protocolos dos Sistemas de Armazenamento

Na linguagem de programação *Ruby* existe o conceito de módulo (semelhante a uma classe, com a diferença de não poder ser instanciado) para encapsular determinadas funcionalidades num espaço de nomes próprio (evitando colisão de nomes com outros módulos) e implementar o que se chama de *mixins*, que é uma funcionalidade parecida com a herança múltipla existente em outras linguagens de programação.
O componente *Radiastore Daemon* possibilita a integração de outros protocolos através de um módulo que engloba as classes que contêm a funcionalidade dos protocolos suportados por este componente. Cada classe tem de implementar três métodos, representando cada processo da secção 4.6. Os três métodos são:

- Método ***put*** para o processo de replicação;
- Método ***get*** para o processo de recuperação;
- Método ***delete*** para o processo de remoção;

A funcionalidade de novos protocolos é encapsulada na respectiva classe e fica imediatamente disponível no componente *Radiastore Daemon* através desse módulo.
Um exemplo da facilidade de integração de novos protocolos, é o serviço *Amazon S3* descrito na secção 2.5. Este serviço disponibiliza uma API, por meio do padrão arquitectural REST, que deve ser integrada no módulo disponível através da criação de uma classe que inclua nos três métodos especificados a funcionalidade de cada acção (exemplo da classe em código *Ruby* no fim desta secção).

[2] A escolha da Réplica de um Ficheiro é feita de forma simples. A primeira Réplica a ser adquirida é a escolhida para ser recuperada. No entanto, a adaptação de um algoritmo mais eficiente para este processo pode ser facilmente integrado no componente.

Exemplo da integração de um novo protocolo

```
module StorageSystem
  # (...)

  class AmazonS3Replication
    def self.put(Fetcher)
      # Replication code goes here
    end

    def self.get(Fetcher)
      # Recovery code goes here
    end

    def self.delete(Fetcher)
      # Removal code goes here
    end
  end

  # (...)
end
```

6 Conclusão

6.1 Contribuições

A existência de Sistemas de Informação que consigam responder às necessidades das rádios comunitárias é escassa. Algumas das soluções que respondem, em parte, a estas necessidades são sistemas fechados, pouco extensíveis e apresentam custos que não estão ao alcance de todas as rádios. Por outro lado, existem sistemas cooperativos e abertos para este tipo de problema mas com limitações de funcionalidades, que não abrangem todos os intervenientes no processo de criação de Ficheiros para os Programas de Autor.
Desta forma é necessário a criação de novas ferramentas, recorrendo às tecnologias existentes, para que este tipo de rádio prospere dentro dos limites da sua comunidade, permitindo assim expandir-se fora dela.

O trabalho realizado ao longo desta dissertação serve para responder a estas necessidades e para incentivar novas ideias que permitam a evolução de sistemas relacionados com o armazenamento e preservação de conteúdos digitais produzidos por rádios comunitárias.
Depois de analisar o funcionamento destas rádios constata-se a importância que os Ficheiros criados pelos Autores da rádio têm para a divulgação de ideias dentro e fora da sua comunidade. Se essa divulgação for suportada por uma infra-estrutura que permita a disponibilidade de toda a informação relacionada com os Ficheiros, maior será a sua expansão. Esta é uma das principais falhas que surgem nos sistemas estudados, e que a solução presente nesta dissertação colmata.
Para além disso, a necessidade de preservar estes Ficheiros é da extrema importância da rádio (uma vez que representam conhecimento único), onde surge a necessidade de conseguir com que estes Ficheiros perdurem ao longo do tempo. A solução mais viável para este tipo de necessidade, e com a cultura de partilha a emergir na Internet, é a replicação dos Ficheiros por diversos Sistemas de Armazenamento aproveitando as suas características e integrando-as com o sistema apresentado.

Tendo em conta a partilha de soluções deste tipo de sistemas, as escolhas tecnológicas adoptadas permitem a continuação do desenvolvimento deste sistema e proporcionam a adaptação das suas funcionalidades para necessidades mais específicas que possam surgir.

6.2 Trabalho Futuro

Esta secção apresenta funcionalidades a acrescentar à proposta da solução desta dissertação. Têm como intuito acrescentar valor ao sistema desenvolvido permitindo o seu alargamento e a sua utilização noutros contextos. As oportunidades de trabalho futuro são as seguintes:

- **Considerar como Ficheiro qualquer tipo de conteúdo digital.** A associação de outros tipos de conteúdos digitais (e.g. imagem, vídeo, texto) ao sistema permite alargar os horizontes dos Autores na produção de novas Emissões. Além de associarem os Ficheiros[1] que fazem parte da Emissão podem associar outro tipo de conteúdo, permitindo criar Emissões mais dinâmicas e completas.

[1] Normalmente é um único Ficheiro que é associado a uma Emissão, sendo a sua Essência um ficheiro de áudio.

- **Analisar automaticamente documentos de Metadados associados aos Ficheiros.** Com este processamento automático, o Autor não teria de se preocupar em preencher os dados associados a um novo Ficheiro. Para esta funcionalidade o sistema terá de reconhecer os formatos de Metadados que se encontram nos documentos de Metadados do Ficheiro.
- **Gerar documentos de Metadados de um Ficheiro em formatos suportados pelo Sistema.** Aproveitando a oportunidade do ponto anterior, o sistema pode gerar documentos de Metadados em diversos formatos. Com esta funcionalidade existe a possibilidade de integração com outros sistemas que possam aproveitar o conteúdo presente neste sistema.
- **Criação de novos recursos Web.** A criação de novas APIs através do padrão arquitectural utilizado (padrão REST) permite a facilidade de integração desta solução com outros sistemas e a difusão da informação presente no sistema, atraindo mais actores e consequentemente mais colaboradores para a comunidade da rádio.
- **Inclusão de leitores multimédia para pré-visualização dos Ficheiros.** Com esta funcionalidade, os actores podem ter uma percepção do conteúdo de cada Ficheiro. Para além disso, facilita a integração de outras novas funcionalidades como é o caso da asssociação de Tags a diversas partes dos Ficheiros.

Bibliografia

[1] C. Fowler D. Thomas and A. Hunt. *Programming Ruby*. The Pragmatic Bookshelf, 2nd edition, 2005.

[2] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[3] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison Wesley, 2002.

[4] T. Hassan. Xbmf - exchange broadcast binary and metadata format. Technical report, 2003. http://sotf.berlios.de/documents/XBMF_V0-31.pdf.

[5] P. Clements L. Bass and R. Kazman. *Software Architecture in Practice*. Addison Wesley, second edition, 2003.

[6] K. Keeton M. Baker and S. Martin. Why traditional storage systems don't help us save stuff forever. *1st IEEE Workshop on Hot Topics in System Dependability*, 2005.

[7] D. Rosenthal; M. Roussopoulos; T. Giuli; P. Maniatis and M. Baker. Using hard disks for digital preservation. *IS&T Archiving Conference*, 2004.

[8] B. Logan P. Moreno J. Thong J. Marston and G. MacCarthy. Newstuner: A simple interface for searching and browsing radio archives. *Cambridge Research Laboratory*, 2004.

[9] H. Gobioff S. Ghemawat and S Leung. The google file system. *19th ACM Symposium on Operating Systems Principles*, 2003.

[10] Amazon Web Services. Amazon simple storage service: Developer guide. Technical report, Amazon, 2006.

[11] Daniel Enrique Zacarias Silva. Radia source: Sistema de informação para gestão de processos de uma estação de rádio. Tese de Mestrado, Instituto Superior Técnico, 2008.

[12] Louie Tabing. *How to do community radio: a primer for community radio operators*. UNESCO Office New Delhi and Regional Bureau for Communication and Information in Asia and the Pacific, 2002.

[13] D. Thomas and David H. Hansson. *Agile Web Development with Rails*. The Pragmatic Bookshelf, 2nd edition, 2007.

Apêndice

A Casos de Uso

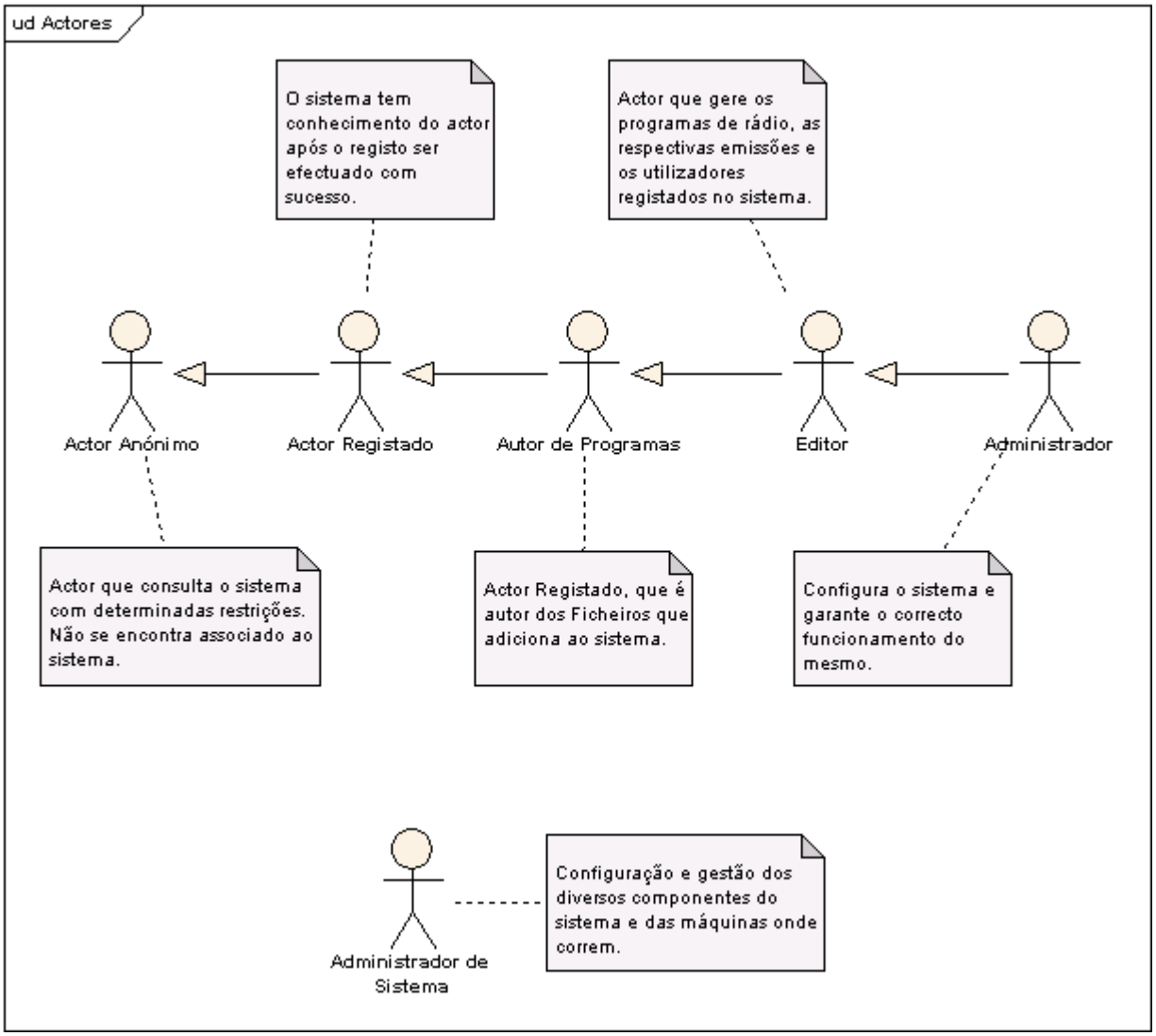

Figura A.1: Actores do Sistema.

A.1 Casos de Uso do Actor Anónimo

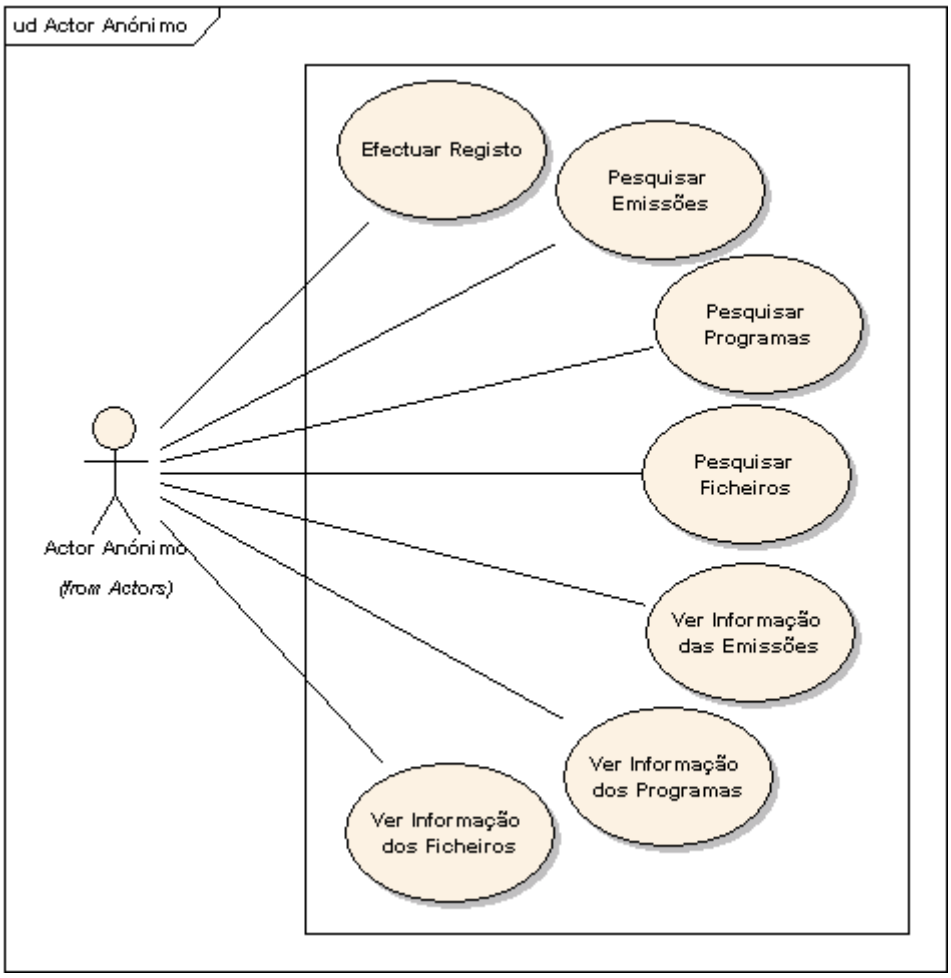

Figura A.2: Casos de Uso do Actor Anónimo.

Caso de Uso 1 - Efectuar Registo

- **Sumário:** Caso de uso que permite um Actor Anónimo registar-se no sistema.
- **Cenário Principal:** Um Actor Anónimo para se registar no sistema acede à opção de registo (*"Join!"*). Após preencher os dados solicitados e o registo ser efectuado com sucesso, passa a ser um Actor Registado.

Caso de Uso 2 - Pesquisar Ficheiros

- **Sumário:** Permite pesquisar Ficheiros previamente adicionados ao sistema.
- **Cenário Principal:** O actor escolhe a opção *Search* do menu, e é-lhe apresentado um campo para inserir as palavras de pesquisa que pretende. Para pesquisar Ficheiros tem de seleccionar a opção *Audio Assets* e clicar no botão *Search*.
- **Cenário Alternativo 1:** O actor pretende pesquisar Ficheiros através de Tags previamente associadas aos mesmos. Após escolher a opção *Search*, o actor tem alterar o modo de pesquisa seleccionando a opção *Search on Tags*. De seguida, a pesquisa decorre como descrita no Cenário Principal.

- **Cenário Alternativo 2:** Através do menu *Tagging* o actor pode filtrar a pesquisa de Ficheiros por Tags. Quando se acede a esta opção do menu, é apresentado um conjunto de Tags que foram associadas previamente aos Ficheiros e seleccionando uma das Tags é apresentado ao actor uma lista de Ficheiros que possuem essa mesma Tag.

Caso de Uso 3 - Pesquisar Programas

- **Sumário:** Permite pesquisar Programas associados ao sistema.
- **Cenário Principal:** O actor escolhe a opção *Search* do menu, e é-lhe apresentado um campo para inserir as palavras de pesquisa que pretende. Para pesquisar Programas tem de seleccionar a opção *Programs* e clicar no botão *Search*.
- **Cenário Alternativo 1:** O actor pretende pesquisar Programas através de Tags que são previamente associadas. Após escolher a opção *Search*, o actor tem de alterar o modo de pesquisa seleccionando a opção *Search on Tags*. De seguida, a pesquisa decorre como descrita no Cenário Principal.
- **Cenário Alternativo 2:** Através do menu *Tagging* o actor pode filtrar a pesquisa de Programas por Tags clicando na opção *Programs' Tags*. Seleccionando uma das Tags é-lhe apresentado uma lista de Programas que possuem essa mesma Tag.

Caso de Uso 4 - Pesquisar Emissões

- **Sumário:** Permite pesquisar Emissões associadas ao sistema.
- **Pré-condições:** Existirem Emissões no sistema inseridas pelos Autores de Programas.
- **Cenário Principal:** O actor escolhe a opção *Search* do menu, e é-lhe apresentado um campo para inserir as palavras de pesquisa que pretende. Para pesquisar Emissões tem de seleccionar a opção *Broadcasts* e clicar no botão *Search*.
- **Cenário Alternativo 1:** O actor pretende pesquisar Emissões através de Tags que são previamente associadas. Após escolher a opção *Search*, o actor tem de alterar o modo de pesquisa seleccionado a opção *Search on Tags*. De seguida, a pesquisa decorre como descrita no Cenário Principal.
- **Cenário Alternativo 2:** Através do menu *Tagging* o actor pode filtrar a pesquisa de Emissões por Tags clicando na opção *Broadcasts' Tags*. Seleccionando uma das Tags é-lhe apresentado uma lista de Emissões que possuem essa mesma Tag.

Caso de Uso 5 - Ver Informação dos Ficheiros

- **Sumário:** Este caso de uso permite ao actor ver a informação associada a um Ficheiro.
- **Cenário Principal:** O actor selecciona a opção *Audio Assets* do menu, e é-lhe apresentado uma lista de todos os Ficheiros existentes no sistema. Pode seleccionar qualquer Ficheiro e consultar detalhadamente a sua informação. Para além disso pode verificar a disponibilidade do Ficheiro actualmente.
- **Cenário Alternativo 1:** Após pesquisar Ficheiros (como descrito no respectivo Caso de Uso) e seleccionar um Ficheiro, o actor pode aceder à sua informação.

Caso de Uso 6 - Ver Informação dos Programas

- **Sumário:** Permite consultar a informação de cada Programa.
- **Cenário Principal:** O actor selecciona a opção *Programs* do menu, e é-lhe apresentado uma lista de todos os Programas existentes no sistema. Nesta lista existe a informação básica de cada Programa e para consultar a informação mais detalhadamente, basta seleccionar o Programa pretendido.
- **Cenário Alternativo 1:** O actor pode consultar a informação de um Programa após efectuar uma pesquisa de Programas (como descrito no respectivo Caso de Uso).

Caso de Uso 7 - Ver Informação das Emissões

- **Sumário:** Permite consultar a informação de cada Emissão.
- **Cenário Principal:** Quando um actor pesquisa Programas todas as Emissões associadas a estes são disponibilizadas, quer na lista de Programas após a pesquisa, ou na informação de cada Programa. O actor ao seleccionar uma Emissão pode consultar detalhadamente toda a sua informação.

A.2 Casos de Uso do Actor Registado

Caso de Uso 8 - Efectuar Autenticação

- **Sumário:** Um Actor Anónimo autentica-se no sistema.
- **Cenário Principal:** O actor selecciona a opção (*Login*) para se autenticar. Introduz o Nome de Utilizador e Palavra-Passe e efectua a respectiva autenticação. De seguida, o actor é redireccionado para a página inicial do sistema.
- **Cenário Alternativo:** Sempre que aparecer um menu de autenticação, o actor pode efectuar a autenticação de imediato.

Caso de Uso 9 - Editar Informação Pessoal

- **Sumário:** O actor pode editar a sua informação pessoal sempre que necessite.
- **Pré-condições:** O actor tem de estar autenticado no sistema.
- **Cenário Principal:** O actor acede à sua informação pessoal através da hiperligação associada ao seu nome e pode editá-la clicando na opção disponibilizada para o efeito.

Caso de Uso 10 - Classificar um Programa

- **Sumário:** Permite classificar um Programa com Tags.
- **Pré-condições:** O actor tem de estar autenticado no sistema.
- **Cenário Principal:** O actor pode associar Tags a um Programa quando acede à sua informação.

Caso de Uso 11 - Classificar uma Emissão

- **Sumário:** Permite classificar uma Emissão com Tags.
- **Pré-condições:** O actor tem de estar autenticado no sistema.

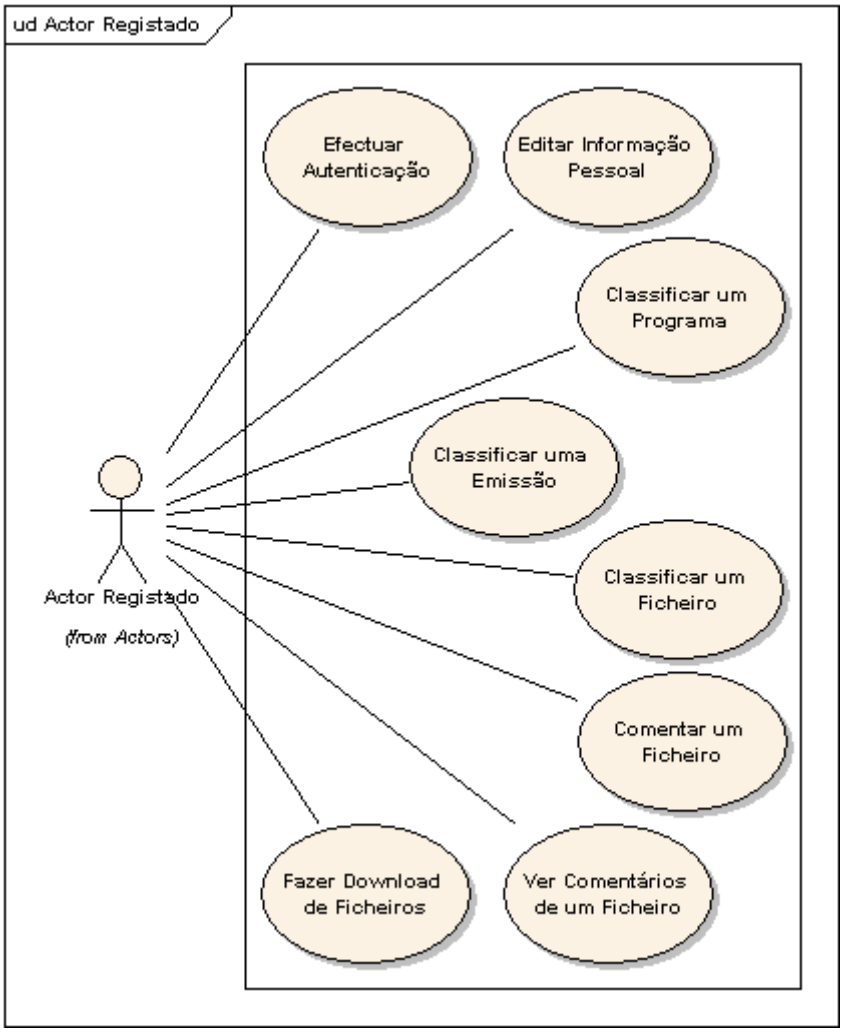

Figura A.3: Casos de Uso do Actor Registado.

- **Cenário Principal:** O actor pode associar Tags a uma Emissão quando acede à sua informação.

Caso de Uso 12 - Classificar um Ficheiro

- **Sumário:** Permite classificar um Ficheiro com Tags.
- **Pré-condições:** O actor tem de estar autenticado no sistema.
- **Cenário Principal:** O actor classifica um Ficheiro associando Tags na sua informação.

Caso de Uso 13 - Comentar um Ficheiro

- **Sumário:** Permite associar um comentário a um Ficheiro.
- **Pré-condições:** O actor tem de estar autenticado no sistema.
- **Cenário Principal:** O actor comenta um Ficheiro, associando texto livre na sua informação.

Caso de Uso 14 - Ver Comentários associados a um Ficheiro

- **Sumário:** Permite ver os comentários associados a um Ficheiro.

- **Pré-condições:** O actor tem de estar autenticado no sistema.
- **Cenário Principal:** O actor, na informação dos Ficheiros, pode ver os comentários associados.

Caso de Uso 15 - Fazer Download de Ficheiros

- **Sumário:** Permite efectuar o download de um Ficheiro ou dos seus Metadados.
- **Pré-condições:**
  - O actor tem de estar autenticado no sistema.
  - O Ficheiro tem de estar disponível no sistema.
- **Cenário Principal:** Quando o actor acede à informação de um Ficheiro tem a possibilidade de efectuar o download do ficheiro áudio associado ou dos respectivos Metadados.

A.3 Casos de Uso do Autor de Programas

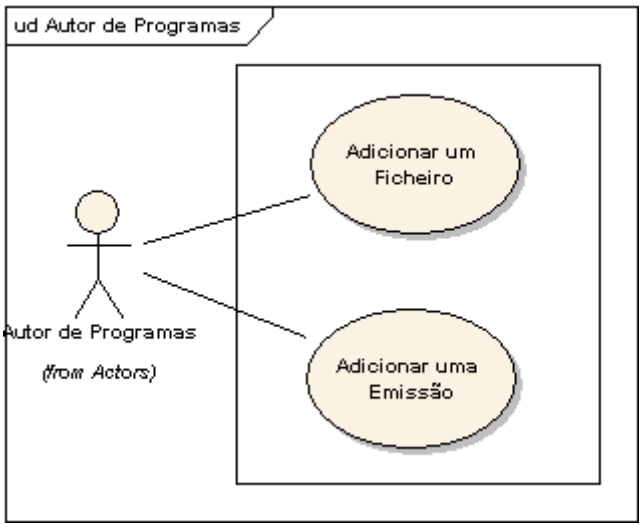

Figura A.4: Casos de Uso do Autor de Programas.

Caso de Uso 16 - Adicionar um Ficheiro

- **Sumário:** Este caso de uso permite ao actor adicionar novos Ficheiros ao sistema, incluindo os seus Metadados.
- **Pré-condições:** O actor tem de ser Autor de, pelos menos, um Programa.
- **Cenário Principal:** O actor acede à opção *Audio Assets* do menu, e pode seleccionar a opção para criar um novo Ficheiro. Após a inserção de todos os dados do novo Ficheiro e a criação tenha sido efectuada com sucesso, o actor é redireccionado para a informação do novo Ficheiro.
- **Cenário Alternativo:** O Ficheiro a criar pode ter mais Autores associados. O actor que insere os respectivos dados selecciona os outros Autores envolvidos na criação deste novo Ficheiro.

Caso de Uso 17 - Adicionar uma Emissão

- **Sumário:** Permite a um Autor de Programa criar uma nova Emissão e associá-la ao respectivo Programa.
- **Pré-condições:**
  - O actor tem de ser Autor do Programa para o qual se adiciona a nova Emissão.
  - Seleccionar pelo menos um Ficheiro para associar à nova Emissão.
- **Cenário Principal:** O actor acede à informação do Programa e selecciona a opção de criar uma nova Emissão. Após a correcta inserção dos seus dados, selecção dos respectivos Ficheiros e sucesso na sua criação o actor é redireccionado para a informação da Emissão criada.

A.4 Casos de Uso do Editor da Rádio

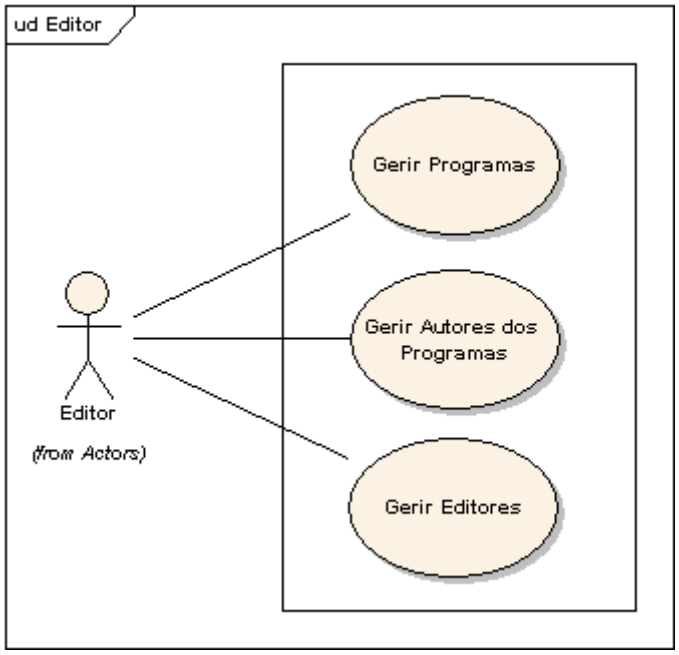

Figura A.5: Casos de Uso do Editor da Rádio.

Caso de Uso 18 - Criar um novo Programa

- **Sumário:** Permite a criação de novos Programas na Rádio.
- **Pré-condições:** O actor tem de ter o privilégio de Editor.
- **Cenário Principal:** O actor selecciona a opção para criar um novo Programa quando acede à opção *Programs* do menu. Após a correcta inserção dos dados e sucesso na criação do novo Programa, o actor é redireccionado para a informação do Programa.

Caso de Uso 19 - Editar a Informação de um Programa

- **Sumário:** Permite alterar a informação de um Programa sempre que necessário.
- **Pré-condições:** O actor tem de ter o privilégio de Editor.
- **Cenário Principal:** O actor quando consulta a informação de um Programa selecciona a opção que permite editar a sua informação. Após o sucesso da sua edição, o actor é redireccionado para a

informação do respectivo Programa.

Caso de Uso 20 - Gerir Autores dos Programas

- **Sumário:** Este caso de uso permite ao actor adicionar ou remover a autoria de um Programa a outro actor.
- **Pré-condições:** O actor tem de ter o privilégio de Editor.
- **Cenário Principal:** O actor quando consulta a informação de um Programa selecciona a opção que permite gerir os Autores desse Programa. São apresentados todos os actores e são mostrados os respectivos privilégios de Autoria. O actor apenas tem de conceder ou retirar esses mesmos privilégios.

Caso de Uso 21 - Gerir Editores da Rádio

- **Sumário:** Um Editor pode adicionar ou remover o privilégio de Edição a outros actores do sistema.
- **Pré-condições:** O actor tem de ter o privilégio de Editor.
- **Cenário Principal:** O actor acede à opção *Manage Users* do menu e são apresentados todos os actores e respectivos privilégios de Edição. O actor pode conceder ou retirar esses mesmos privilégios.

A.5 Casos de Uso do Administrador da Rádio

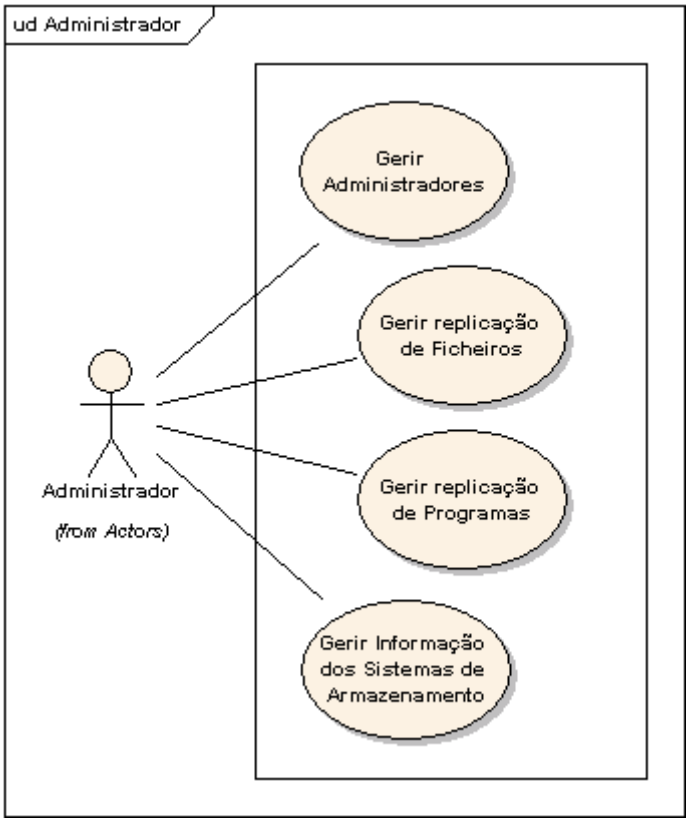

Figura A.6: Casos de Uso do Administrador da Rádio.

Caso de Uso 22 - Gerir Administradores

- **Sumário:** Permite ao actor gerir o privilégio de administração dos actores da rádio.
- **Pré-condições:** O actor tem de ter o privilégio de Administrador.
- **Cenário Principal:** O actor acede à opção *Manage Users* do menu e são apresentados todos os actores e respectivos privilégios de Administração. O actor pode conceder ou retirar esses mesmo privilégios.

Caso de Uso 23 - Gerir Replicação de Ficheiros

- **Sumário:** Permite gerir a replicação de cada Ficheiro nos Sistemas de Armazenamento associados ao sistema.
- **Pré-condições:** O actor tem de ter o privilégio de Administrador.
- **Cenário Principal:** Na informação de cada Ficheiro o actor selecciona por Sistema de Armazenamento, a sua replicação ou cancelamento da mesma.

Caso de Uso 24 - Gerir Replicação de Programas

- **Sumário:** Permite gerir a replicação de cada Programa nos Sistemas de Armazenamento associados ao Sistema.
- **Pré-condições:** O actor tem de ter o privilégio de Administrador.
- **Cenário Principal:** Na informação de cada Programa o actor selecciona por Sistema de Armazenamento, o início da replicação ou cancelamento da mesma do respectivo Programa.

Caso de Uso 25 - Criar um Sistema de Armazenamento

- **Sumário:** Permite adicionar um novo Sistema de Armazenamento ao sistema. Este novo Sistema de Armazenamento servirá como sistema de replicação para os Ficheiros associados ao sistema.
- **Pré-condições:** O actor tem de ter o privilégio de Administrador.
- **Cenário Principal:** O actor acede à opção *Configurations* do menu, e pode consultar todos os Sistemas de Armazenamentos associados ao sistema. Aí pode criar um novo Sistema de Armazenamento seleccionando a opção para o efeito. Após a inserção dos dados e criação com sucesso do novo Sistema de Armazenamento, o actor é redireccionado para a informação do mesmo.

Caso de Uso 26 - Editar Informação de um Sistema de Armazenamento

- **Sumário:** Permite alterar a informação dos Sistemas de Armazenamento presentes no sistema.
- **Pré-condições:** O actor tem de ter o privilégio de Administrador.
- **Cenário Principal:** O actor ao aceder à informação de um Sistema de Armazenamento pode seleccionar a opção que permite alterar informação deste Sistema.

Caso de Uso 27 - Gerir Replicação de Programas por Sistema de Armazenamento

- **Sumário:** Permite gerir a replicação dos Programas por Sistema de Armazenamento e visualizar no momento que Programas estão ou não replicados num determinado Sistema.
- **Pré-condições:** O actor tem de ter o privilégio de Administrador.
- **Cenário Principal:** Na informação do Sistema de Armazenamento, o actor selecciona a opção que permite gerir a replicação de Programas neste Sistema. Deste modo, pode indicar o início ou cancelamento da replicação de um programa.

Caso de Uso 28 - Gerir Replicação de Ficheiros por Sistema de Armazenamento

- **Sumário:** Permite gerir a replicação dos Ficheiros por Sistema de Armazenamento e visualizar no momento que Ficheiros se encontram replicados, ou não, num determinado Sistema.
- **Pré-condições:** O actor tem de ter o privilégio de Administrador.
- **Cenário Principal:** Na informação do Sistema de Armazenamento, o actor selecciona a opção que permite gerir a replicação de Ficheiros neste Sistema. Deste modo, pode indicar que determinado Ficheiro é para replicar ou indicar a sua remoção do Sistema de Armazenamento no caso de já estar replicado.

B Modelos de Domínio

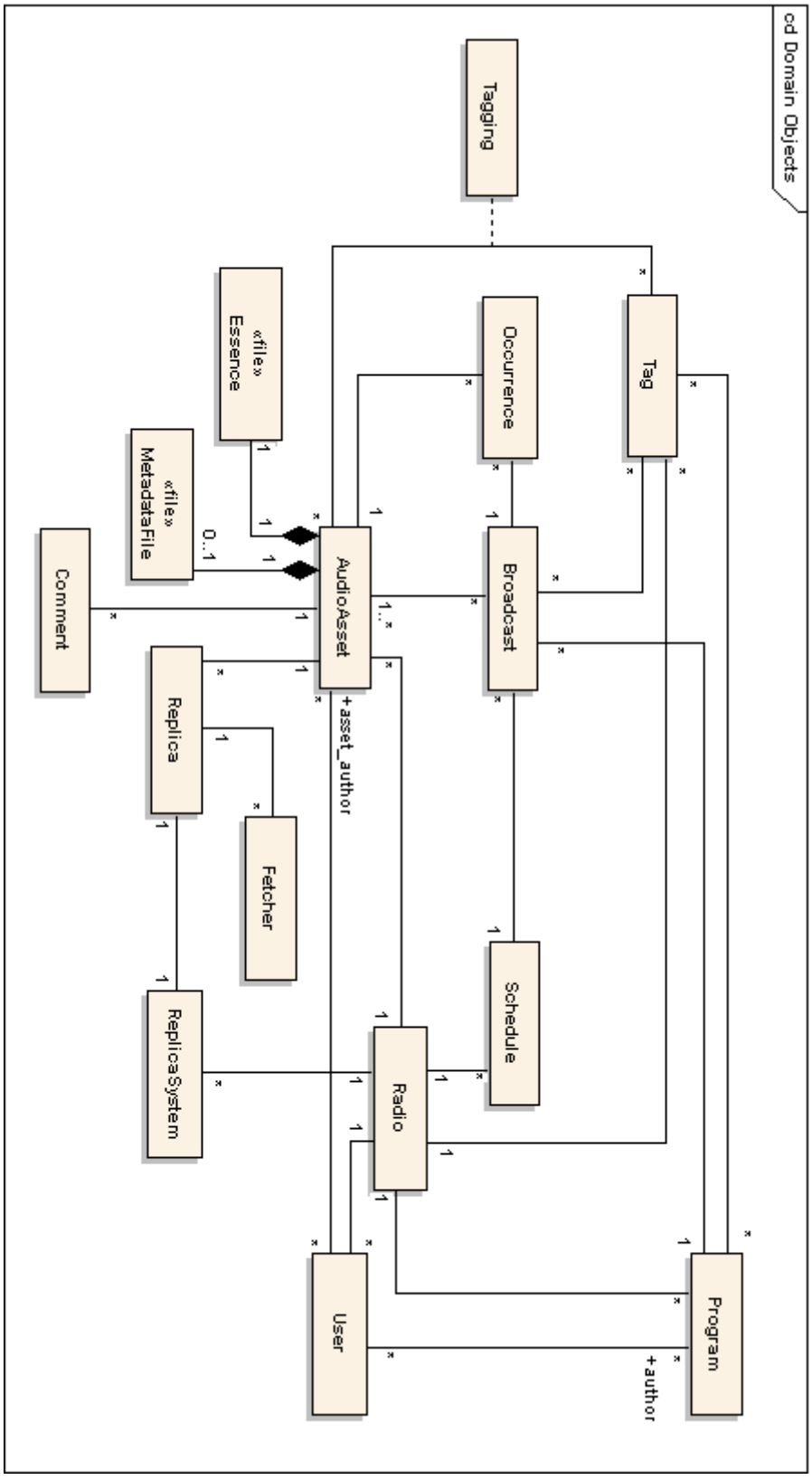

Figura B.1: Modelo de Domínio do componente Aplicação Web *Radiastore*.

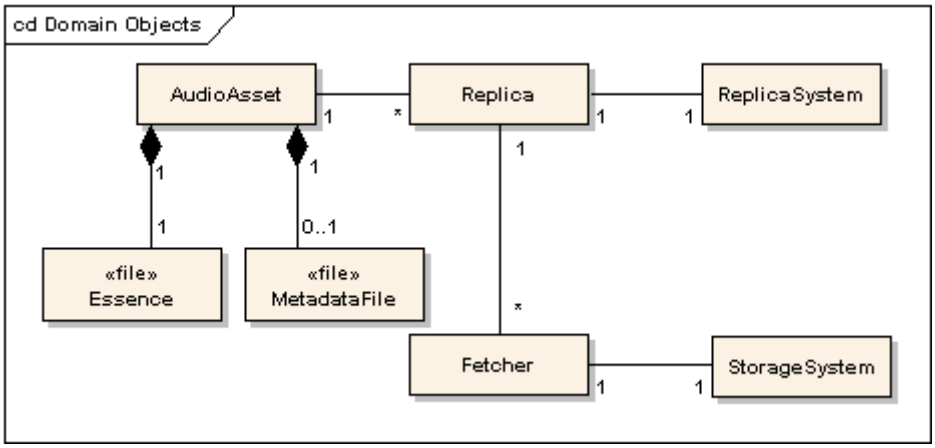

Figura B.2: Modelo de Domínio do componente *Radiastore Daemon*.

C Modelo de Classes

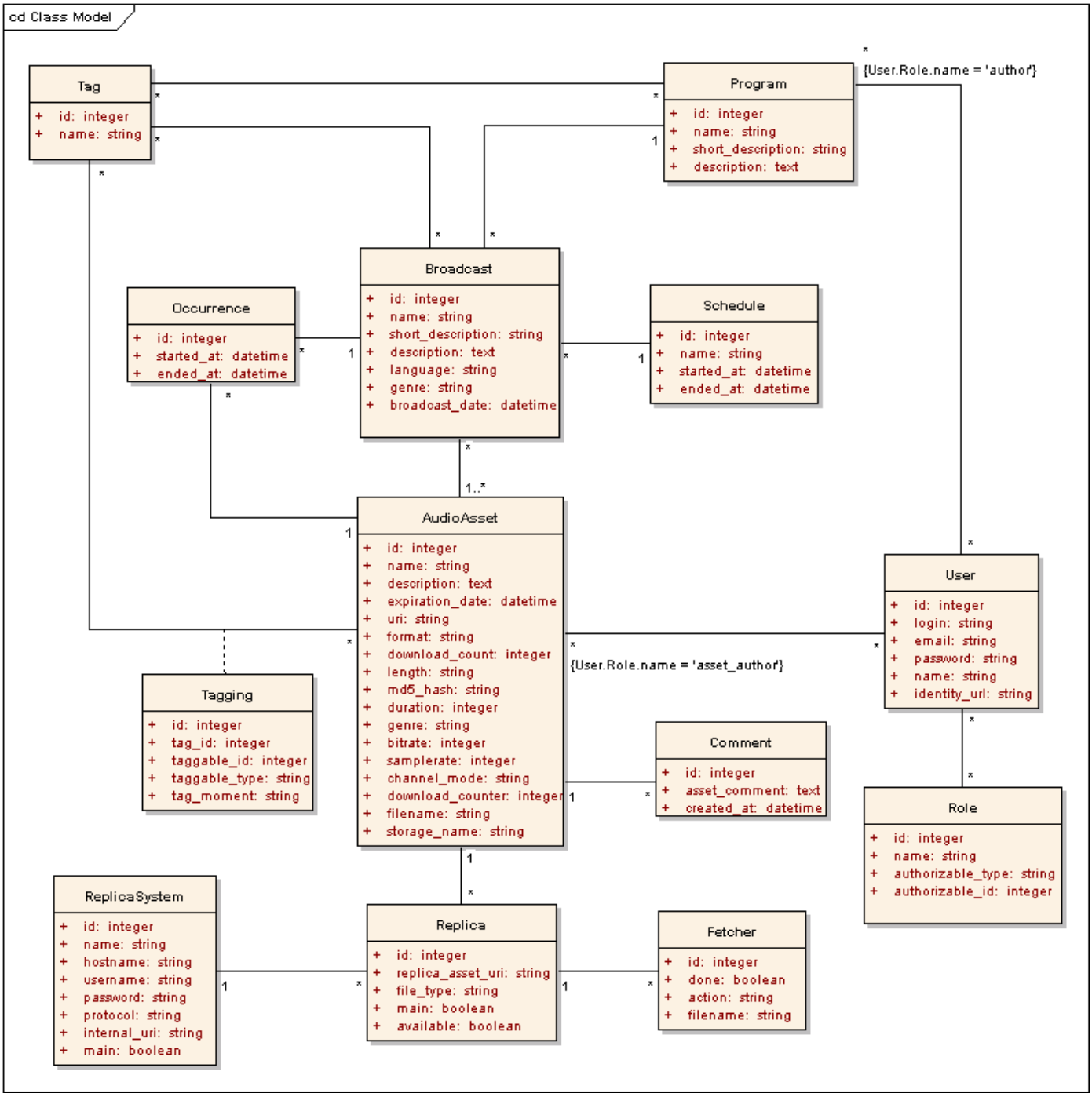

Figura C.1: Modelo de Classes.