FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Modelo de Replicação para a Preservação de Dados em Repositórios

Micael Ferreira Alves de Pinho

Mestrado Integrado em Engenharia Informática e Computação

Orientadora: Doutora Maria Cristina de Carvalho Alves Ribeiro (Professora Auxiliar da FEUP)

17 de Julho de 2012

Modelo de Replicação para a Preservação de Dados em Repositórios

Micael Ferreira Alves de Pinho

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: Doutor José Manuel Magalhães Cruz (Professor Catedrático da FEUP)

Vogal Externo: Doutora Daniela Carneiro da Cruz (Professora Auxiliar da Universidade Lusófona)

Orientadora: Doutora Maria Cristina de Carvalho Alves Ribeiro (Professora Auxiliar da FEUP)

---

17 de Julho de 2012

Resumo

A entrada no mercado das novas tecnologias digitais impulsionou o nosso mundo para uma era em que a criação, manipulação e o armazenamento de informação de forma digital cresceu significativamente. Por outro lado, surgiram alguns problemas relacionados com a preservação a longo prazo dessa mesma informação.

Atualmente, já existem repositórios que permitem o armazenamento e preservação de conteúdos digitais. Contudo, acontecimentos imprevisíveis, como ocorrência de catástrofes naturais, falhas de hardware, ou até mesmo erros humanos, podem provocar danos nos servidores, colocando em risco a informação existente.

Alguns conteúdos existentes nos repositórios digitais poderão ter um elevado valor e importância, e como tal, a sua perda poderá ter consequências graves. Por exemplo, no contexto de investigação, os conjuntos de dados (*datasets*) recolhidos possuem uma grande importância na validação de resultados obtidos em investigação e além disso constituem uma importante fonte de evidência para trabalhos futuros.

Hoje em dia já existem várias plataformas de repositórios disponíveis no mercado, sendo uma delas o DSpace. Este facilita o processo de criação de repositórios institucionais, para a recolha, partilha e preservação digital de conteúdos intelectuais, dos mais diversificados formatos digitais. O DSpace também fornece um vasto conjunto de ferramentas, para uma gestão eficaz dos conteúdos digitais, e em caso de algum formato digital se tornar obsoleto, é possível a sua migração para um dos formatos mais divulgados.

O projeto de dissertação aqui apresentado teve por objetivo a criação de um sistema para a geração de cópias de segurança dos conteúdos existentes numa instância DSpace, para que em caso de necessidade futura, seja possível a sua restauração. Optou-se por um repositório DSpace, visto que os investigadores da Universidade do Porto já estão bastante familiarizados com esta plataforma.

Desenvolveu-se um componente de *backup* para a geração de cópias de segurança e outro para restauração dos conteúdos. Estes componentes foram desenvolvidos utilizando as ferramentas de *backup* disponíveis no DSpace. Efetuaram-se melhorias nas mesmas, através da criação de interfaces mais intuitivas e melhoria dos próprios processos com a redução do processamento e tempo requerido.

A geração das cópias de segurança levantou questões sobre a disponibilidade e integridade das mesmas, visto que é possível que elas também sofram danos. O envio destas para outras localizações remotas foi ponderado, tendo-se analisado a possibilidade de utilização de serviços de armazenamento *cloud*. De modo a tornar abstrata a ligação com diferentes serviços *cloud* optou-se pela utilização do *software open-source* jclouds. Para garantir a consistência, integridade e acessibilidade das cópias de segurança foram desenvolvidas duas componentes, uma para o envio das mesmas para a *cloud* e outra para a obtenção delas da *cloud*.

Abstract

The appearance of new digital technologies in the market has launched our world into an era in which the creation, manipulation and storage of digital information have grown significantly in terms of importance. On the other hand, many problems associated with the preservation and interpretation of such information also rose.

One of the current solutions for this problems are some digital repositories that allow the storage and preservation of digital content. However, the unpredictable can happen, for example, natural disasters, hardware failure or human failure, all of which may cause damage to the server and put the stored information at risk.

Some of the content of the digital repository is of great value and importance, and for that reason, it could lead to serious consequences. For example, in research activities, the datasets are extremely important to validate the results obtained in researches and for reuse in futures researches.

At the present, there is more than one option of digital repository in the market and DSpace is one example. DSpace allows one to easily create an institutional repository, in order to collect, share and preserve intellectual digital content in diverse file formats. DSpace also has many tools for the effective management of digital content. In case of a file format becoming outdated, it is possible to migrate its content to a new file format.

The goal of this project, is the development of a system for the creation of backups from the digital content preserved in a DSpace repository, in order to allow future restoration if necessary. DSpace has been chosen because it is widely used across the University of Porto.

Backup and restoration components have been developed for the creation and restoration of digital content. These components have been developed with the use of the backup tools incorporated in DSpace. Some improvements have been made through the creation of intuitive interfaces and the improvement of processes allowing the reduction of time and processing required.

The backup file creation posed problems regarding the integrity and availability of the information because it was also exposed to the risk damage. The upload to other remote locations was considered and the use of cloud storage services was analysed. In order to abstract the connection with different cloud services, the open-source software jclouds was used. To allow the consistency, integrity and accessibility of the backup files, two components have been developed, one of them to send the backup files to the cloud and the other one to receive the backup files from the cloud.

Agradecimentos

Existem alguns agradecimentos que gostaria de fazer para aqueles que me acompanharam, influenciando assim o meu percurso académico e a realização desta dissertação.

Aos meus pais, Valdemar do Paulo e Maria Carolina, por todo o apoio que me deram e pela paciência que tiveram comigo, principalmente no decorrer destes últimos seis anos. Obrigado por tudo, agradeço-vos do fundo do meu coração.

À minha irmã, Sónia Pinho, por me ter aconselhado a vir para a Universidade e ter estado sempre disponível para me ajudar caso eu precisasse. Obrigada maninha por tudo, sabes que és especial. Um agradecimento também ao Lucas pela ajuda que ele me prestou no final da dissertação.

Um agradecimento muito especial à minha namorada, Carolina Passeira, pela paciência, apoio, ombro amigo, confiança, ajuda, e por muito que me custe a admitir, obrigado também pelos puxões de orelhas. Obrigado pelos nossos momentos, conversas e raspanetes, sem ti poderia ainda não ter chegado aqui.

À minha orientadora, Professora Maria Cristina Ribeiro, que acompanhou o meu trabalho ao longo destes meses agradeço todo o auxílio e disponibilidade prestada.

Agradeço também ao meu colega de laboratório, João Rocha da Silva, por todo o apoio prestado e orientações no meu projeto de dissertação.

Ao meu técnico de IT, colega de curso e amigo Alexandre Rodrigues, muito obrigado pelas nossas longas conversas e por toda a ajuda prestada.

Não poderia deixar de referir a minha segunda casa, durante estes dois últimos anos, a JuniFEUP. Obrigado pelos ensinamentos e lições de vida que me deste para o futuro, mas principalmente obrigado pelos amigos que me deste para a vida.

Sem nunca esquecer os amigos que me rodeiam e os que já partiram, obrigado a todos pelos momentos que tivemos, vocês sabem quem são e o quanto significam para mim.

Um final agradecimento e não menos sentido, aos docentes e funcionários da FEUP, pelos conhecimentos e lições de vida que me transmitiram.

Micael Pinho

"Data is not information,
information is not knowledge,
knowledge is not understanding,
understanding is not wisdom."

Clifford Stoll

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capítulo são apresentados o contexto e enquadramento deste projeto de dissertação. Além disso, é descrita a estrutura da dissertação com referência aos diferentes capítulos existentes e os temas abordados neles.

1.1 Contexto e Enquadramento

Desde que surgiram as tecnologias digitais, a criação, manipulação e armazenamento de informação digital cresceu significativamente. Por outro lado, surgiram problemas com a preservação a longo prazo dessa mesma informação.

Atualmente, já existem repositórios que permitem o armazenamento e preservação de conteúdos digitais. Todavia, existem alguns pormenores nos quais se deve dar especial atenção como por exemplo, garantir que não ocorrem perdas ou danos nos conteúdos armazenados num repositório. Essas perdas ou danos podem ocorrer por vários motivos, nomeadamente erros humanos, ou catástrofes naturais, que provoquem danos físicos no servidor, ou falhas nas componentes de *hardware* do servidor.

No contexto de projetos de investigação, os conteúdos digitais são conjuntos de dados denominados de *datasets*. Estes possuem estruturas e informação bastante diversificadas, pelo facto de existir uma grande variedade de áreas de investigação, podendo assumir a forma de registos textuais, imagens ou vídeos. Os *datasets* existentes em repositórios possuem normalmente um elevado valor e importância, como tal, a ocorrência da sua perda é um acontecimento nefasto, visto que a sua reprodução poderá ser um processo impossível. Por exemplo, no contexto de investigação, os *datasets* possuem uma grande importância na validação de resultados obtidos em investigação, constituindo também uma importante fonte de evidência para trabalhos futuros.

Existem na atualidade várias soluções de repositórios disponíveis no mercado, sendo uma delas o *software open source* DSpace. O DSpace torna fácil o processo de criação de repositórios institucionais, que permitem a recolha, partilha e preservação digital de

conteúdos intelectuais. Esta plataforma permite ainda o armazenamento de uma grande variedade de formatos digitais, como por exemplo: *datasets*, imagens, ficheiros de áudio, ficheiros de vídeo, programas de computador, entre outros. Para além disso, fornece uma vasto conjunto de ferramentas, para uma gestão mais eficiente dos conteúdos digitais. No caso de algum desses formatos se tornar obsoleto, é possível a migração do mesmo para um novo formato digital a definir.

Tendo em conta a possibilidade de ocorrência de danos nos conteúdos existentes num repositório, este projeto de dissertação teve por objetivo a implementação de um sistema para a geração de cópias de segurança dos conteúdos existentes numa dada instância DSpace. Assim, em caso de futura necessidade, será possível o uso das mesmas para se efetuarem restauros. A escolha da utilização da plataforma DSpace prende-se com o facto de os investigadores da Universidade do Porto já estarem bastante familiarizados com a sua utilização.

Inicialmente desenvolveu-se um componente de *backup* para a geração das cópias de segurança dos conteúdos existentes numa instância DSpace. De modo a assegurar a restauração de conteúdos, com recurso às cópias de segurança geradas, procedeu-se ao desenvolvimento de outro componente. Ambos os componentes foram desenvolvidos com recurso às ferramentas de *backup* e restauração existentes no DSpace. Efetuaram-se melhorias das mesmas, através da criação de interfaces mais intuitivas e melhoria dos próprios processos, assegurando deste modo a redução do processamento e tempo requerido.

Contudo, a geração das cópias de segurança, levantou questões sobre a disponibilidade e integridade das mesmas, visto que elas também se encontram sujeitas à ocorrência de danos. De forma a solucionar este problema, ponderou-se o envio de réplicas das cópias de segurança geradas para outras localizações remotas, assegurando assim a sua disponibilidade, consistência e integridade futura. Analisou-se a possibilidade de utilização de serviços de armazenamento *cloud*, tendo-se optado pela utilização do *software open-source* jclouds, uma vez que este permite uniformizar a ligação com diferentes serviços *cloud*.

Resumindo, pretende-se com este projeto de dissertação verificar se é possível assegurar a criação de cópias de segurança dos conteúdos existentes numa instância DSpase, garantindo assim a consistência, integridade e disponibilidade das mesmas, caso no futuro seja necessário a sua utilização para a restauração de conteúdos, que se possam ter perdido ou estejam danificados.

1.2 Estrutura da Dissertação

Esta dissertação, além deste capítulo de introdução, é composta por mais 4 capítulos.

De seguida, no Capítulo 2, é referido a existência de informação digital e importância da sua preservação. Além disso, também é referido a existência de um problema na

preservação de conteúdos digitais existentes em repositórios, sendo de seguida apresentada uma abordagem para a solução. No Capítulo 3, são apresentadas e analisadas as tecnologias utilizadas no decorrer deste projeto de dissertação. No Capítulo 4 é apresentado detalhadamente o sistema desenvolvido. Por fim, no Capítulo 5 apresentam-se as conclusões deste projeto de dissertação.

Capítulo 2

Preservação de Informação Digital

Desde os meados do século XX, com maior incidência nas últimas duas décadas, verificou-se uma completa revolução na geração de informação, como a nível de dimensão e complexidade como de importância. Pode-se afirmar que esta revolução foi fruto do crescimento tecnológico que se verificou nos ramos da informática e da comunicação. Atualmente, estas áreas disponibilizam meios que proporcionam e facilitam a criação, manipulação e armazenamento da informação, fazendo com que ocorra diariamente uma avultada produção de informação [RSRF10, Fer06].

Apesar de ser extremamente importante, a preservação de informação, o acesso de toda a informação existente é uma tarefa inviável, inútil e irrelevante. No entanto, há que assegurar que gerações futuras tenham o acesso a informação significativa e relevante da era atual, tal como hoje é possível consultar informação oriunda de gerações passadas, principalmente dos últimos cinco séculos [Fer06].

A preservação digital é também uma prioridade nas organizações, implicando em alguns casos conversões de formatos analógicos para digitais. Outro objetivo da preservação digital é garantir a integridade da informação preservada, assegurando que não ocorrem danos nem deterioração. Em caso de necessidade, é importante que seja possível reverter um dano ocorrido, ou migrar de formato digital caso o usado se tenha tornado obsoleto [PN03].

A preocupação da preservação afeta também organizações que incorporem atividades de investigação científica, principalmente em áreas onde é necessário o acesso a um grande volume de informação, como por exemplo a genética, medicina, física ou meteorologia. É essencial que esta informação possa ser obtida de modo a que possa ser processada, sempre que surja a necessidade, pois representam um investimento significativo e, em muitos casos, a informação existente é insubstituível. Nestes contextos, a informação é normalmente designada de dados científicos [RSRF10, HBH09, HTT09].

No contexto de investigação, os conjuntos de dados (*datasets*) utilizados são relevantes na validação de resultados obtidos e na comparação de técnicas e abordagens utilizadas, sendo dessa forma úteis para a execução de trabalhos futuros.

Contudo, como qualquer plataforma informática, os repositórios estão sujeitos à ocorrência de acontecimentos que podem provocar sérios danos, ou até mesmo a perda dos conteúdos digitais, alguns deles de elevado valor e importância, como por exemplo os *datasets*, o que pode gerar algumas consequências graves.

2.1 Dados Científicos

Segundo a definição da *Organization for Economic Cooperation and Development* (OECD), dados científicos (*datasets*) são "registos factuais que são usados como fontes primárias na investigação científica, e que geralmente são aceites na comunidade científica como necessários para validar os resultados de uma investigação científica"[1], que contêm estruturas e conteúdos bastante diversificados, devido à existência de uma grande diversidade de áreas de investigação, armazenados em conteúdos textuais, numéricos, imagens ou audiovisuais.

Além disso, os dados científicos também possuem uma dimensão bastante variável, que pode não ultrapassar algumas centenas de kilobytes, no caso de registos de observações individuais ou ensaios de pequenos laboratórios, ou noutro extremo, podem ser gerados várias dezenas de petabytes de dados científicos por dia, como por exemplo no âmbito das atividades científicas no *Large Hadron Collider*[2] do CERN [RSRF10].

Os *datasets* são de extrema importância para uso futuro. A validação de resultados obtidos em investigação científica baseadas nos mesmos e a reutilização dos *datasets* em investigação futuras são os casos de utilização mais predominante, na atualidade. Desta forma, os *datasets* devem ser cuidadosamente preservados, para que, em casos de futura necessidade, possam ser corretamente acedidos e interpretados [RSRF10].

2.2 Preservação e Acessibilidade dos Dados

Nos últimos anos, generalizou-se o conceito e os movimentos a favor dos dados abertos ou *Open Data*, "um conteúdo ou informação é considerado aberto se, qualquer pessoa for livre de o utilizar, manipular e distribuir, sujeito unicamente à identificação da sua

[1] Tradução da definição de "Research data"(pág.13) - *OECD Principles and Guidelines for Acess to Research Data from Public Funding*. Paris, 2007. Disponível em: http://www.oecd.org/dataoecd/9/61/38500813.pdf [consultado em: 5 de Janeiro de 2012]

[2] Mais informações sobre o *Large Hadron Collider* do CERN acessíveis em: http://public.web.cern.ch/public/en/LHC/LHC-en.html

origem", definição segundo o projeto *Open Definition*[3] sobre a alçada da fundação *Open Knowledge Foundation*[4].

Os movimentos em favor de *datasets* abertos defendem que os dados devem ser disponibilizados publicamente de forma gratuita, sem restrições de *copyright*, patentes ou outros mecanismos de controlo. Neste sentido, assemelham-se a outros movimentos de "abertura", tais como o *Open Source*[5] ou o *Open Access*[6], que contudo possuem dinâmicas e objetivos próprios.

Independentemente de qualquer movimento ou conceito existente, atualmente existe um ideal sobre o qual todos os investigadores concordam, a preservação e acessibilidade de *datasets* é um processo que ainda tem que ser muito melhorado, para que se possa retirar o máximo proveito.

Quando se aborda o acesso a *datasets* preservados, é preciso ter em atenção alguns riscos e cuidados, gerados por questões de sigilo, confidencialidade, ou mesmo de direitos de autor. Apesar disso, a preservação e acessibilidade futura dos *datasets*, é de extrema importância, devido principalmente aos seguintes fatores [TR10]:

- Valor da informação: potencial valor dos dados em termos de reutilização, qualidade e importância nacional e/ou internacional, origem, tamanho, escala, custos associados com a sua geração ou o caráter inovador da investigação associada;
- Informação única: os dados contêm informação de observações únicas, que não poderão ser reproduzidas novamente no futuro;
- Importância dos dados para a história, particularmente na história da ciência.

Como se pode verificar, a preservação de *datasets* é de extrema importância, contudo é preciso ter em atenção que nas duas últimas décadas, a quantidade de *datasets* existentes deixou de ser uma raridade, passando a ter uma presença abundante. Este crescimento explosivo na quantidade de *datasets* existentes deveu-se principalmente à introdução de novas tecnologias no mercado informático e de comunicação, que facilitaram a criação, manipulação e o armazenamento de informação digitalmente. Sendo assim, tornou-se inevitável a existência de uma pré-seleção com o intuito de preservar unicamente os *datasets* relevantes [Fer06].

Para que um determinado *dataset* seja um dos elegíveis para o processo de preservação e acessibilidade futura, deve identificar-se com pelo menos um dos seguintes requisitos [TR10]:

- Reutilização: a reutilização dos *datasets* é o motivo mais comum e importante. Um caso de uso de reutilização é a reanálise dos *datasets* sobre uma nova perspetiva

[3] Mais informações sobre o projeto *Open Definition* acessíveis em: http://opendefinition.org/
[4] Mais informações sobre a fundação *Open Knowledge Foundation* acessíveis em: http://okfn.org/
[5] Mais informações sobre o movimento *Open Source* acessíveis em: http://www.opensource.org/
[6] Mais informações sobre o movimento *Open Access* acessíveis em: http://www.eprints.org/openaccess/

de investigação, originada por avanços científicos. Outro caso de uso é a combinação ou recombinação de *datasets*, ou até mesmo comparação com antigos *datasets*, com o intuito de obter ou complementar informação. Esta reutilização pode ser feita dentro ou fora do contexto de investigação no qual o *dataset* foi gerado, denominando-se como uso secundário;

- Verificação: este tipo de utilização é quase sempre originado pela obrigação de cumprimento de códigos de conduta existentes para investigação, como por exemplo, o *Netherlands Code of Conduct for Scientific Practice*[7], que delibera que os *datasets* devem ser mantidos acessíveis para verificação, num determinado período de tempo. Pode-se afirmar que existe uma certa semelhança entre reutilização e verificação, visto que outros cientistas podem querer reanalisar *datasets* antigos, só com o intuito de verificar, ou até mesmo esclarecer dúvidas, sobre investigação científicas baseadas nesses mesmos *datasets*;

- Património: para uso em investigação histórica, no caso particular da história da ciência, ou até mesmo, preservação com o único intuito de criar um património cultural.

Existem outros fatores, sobre os quais se deve também ter uma especial atenção quando se faz o processo de pré-seleção dos *datasets*: documentação, legalidade, infraestrutura e aspetos financeiros.

Relativamente a fatores de documentação, entenda-se os metadados, que não é mais do que informação que permite uma melhor preservação dos *datasets*. O principal intuito da existência de metadados é o de descrever e documentar os dados, processos e atividades relacionadas com a geração dos *datasets*. Estes devem conter informação que descreva a sua origem (tempo ou espaço, métodos, instrumentos de recolha e transformações aplicadas), âmbito, autoria, propriedade e condições de recolha. Além disso, é importante que exista informação sobre os campos existentes nos registos, que será composta por um descritivo de cada um dos campos e referência ao seu tipo de dado [TR10, Fer06, RSRF10].

De salientar, que quanto mais complexos forem os *datasets*, mais complexos terão que ser os detalhes da informação contida nos metadados. A existência de metadados adequados e normalizados é um requisito essencial para o acesso e reutilização dos *datasets*, pois uma interpretação incorreta dos mesmos poderá levar a enormes falhas em investigação científicas. Uma preservação de *datasets* sem que exista a adequada documentação para cada um deles, é um completo desperdício de tempo e recursos [HBH09, RSRF10].

Os fatores legais e as limitações éticas também são de extrema importância, pois poderão existir alguns *datasets*, que por motivos de propriedade intelectual ou permissões,

[7] Mais informações sobre *Netherlands Code of Conduct for Scientific Practice* acessíveis em: http://media.leidenuniv.nl/legacy/netherlands_code_of_conduct_for_scientific_practice.pdf

poderão estar restritos a um determinado grupo de pessoas, e como tal, não possam estar públicos para toda a comunidade [TR10, Fer06].

Por último, os fatores relacionados com as infraestruturas e os fatores financeiros estão intimamente interligados. Neste caso, o problema de financiamento, mas além disso, outros requisitos são necessários: espaço físico, energia, especialistas e infraestruturas. Outro problema pertinente é a mão-de-obra qualificada para fazer a gestão dos *datasets*, em estado de preservação atual ou futura. Além disso, devido à atual incerteza em volta das estratégias de preservação, ainda é impossível calcular os custos associados a longo prazo [TR10].

2.3 Protótipo de Repositório de Dados Científicos

Atualmente, várias instituições de investigação espalhadas pelo mundo, começam a perceber o real valor da preservação digital, no contexto de dados científicos. Desde então têm surgido vários projetos neste âmbito: *Data Asset Framework*[8], *Edinburg DataShare*[9] ou *DANS Data Archive*[10] são bons exemplos desses esforços, com o intuito de assegurar futuramente uma melhor preservação digital de dados científicos.

A Universidade do Porto, tendo também notado este potencial, tem neste momento, os seus serviços centrais, em parceria com um grupo de investigação interno, o desenvolvimento de um projeto, cujo objetivo principal é determinar as principais necessidades de curadoria de dados científicos, usando para caso de estudo os diferentes núcleos de investigação que a Universidade do Porto alberga [dSRL11, dSRL].

No âmbito deste projeto foi desenvolvido um protótipo de repositório de dados científicos, havendo uma grande proximidade entre a equipa de desenvolvimento e os investigadores dos diversos núcleos de investigação, de modo a que os resultados obtidos possam ser validados [dSRL11, dSRL].

Com um variado leque de opções em repositórios *open-source*, a escolha da equipa de desenvolvimento responsável pelo projeto, acabou por recair sobre a plataforma de repositório DSpace. A preferência sobre o DSpace, entre outros motivos, deveu-se ao facto da proximidade que os investigadores da Universidade do Porto já têm com a utilização desta plataforma, devido à existência de dois repositórios operacionais: o Repositório Aberto[11] e o Repositório Temático [dSRL11]. Uma descrição mais elaborada sobre a plataforma DSpace encontra-se na secção 3.1.

[8] Mais informações sobre o projeto *Data Asset Framework* acessíveis em: http://www.data-audit.eu/

[9] Mais informações sobre o projeto *Edinburg DataShare* acessíveis em: http://datashare.is.ed.ac.uk/

[10] Mais informações sobre o projeto *DANS Data Archive* acessíveis em: http://www.dans.knaw.nl/

[11] Mais informações sobre o Repositório Aberto da Universidade do Porto acessíveis em: http://repositorio-aberto.up.pt/

2.4 Perda de Conteúdos Armazenados em Repositórios Digitais

A preservação de informação digital sempre foi um problema bastante complexo, mas desde há algumas décadas atrás, que têm sido feitos esforços no sentido de preservar o máximo de informação possível, para que futuras gerações possam ter acesso a esse conhecimento.

Apesar de todos os cuidados, acidentes e catástrofes acontecem, levando à perda de informação muito valiosa. Um dos momentos mais tristes na história da humanidade, no contexto de perda de informação, terá acontecido no ano de 646, quando a Biblioteca de Alexandria ardeu por completo, perdendo-se para sempre todo o conhecimento ali existente[12].

Atualmente existem repositórios de conteúdos digitais, entre os quais se contam a plataforma DSpace mencionada na secção 3.1, que tem como intuito garantir a preservação e acessibilidade de conteúdos digitais, contudo existem algumas ameaças externas que põem em questão a sua correta preservação e acessibilidade.

Já foram noticiadas catástrofes que aconteceram um pouco por todo o mundo, desde inundações, terramotos, furacões, tornados, maremotos e até casos de acentuada queda de granizo, que provocaram sérios danos, sendo que algumas catástrofes chegaram a atingir universidades e institutos de investigação levando à perda de grandes quantidades de informação. Basta recuar apenas alguns anos e é possível relembrar um vasto leque de exemplos: inundação na biblioteca da Universidade do Havaí em 2004, *tsunami* no Oceano Índico em 2004, furacão Katrina que em 2005 atingiu os Estados Unidos da América, inundações na biblioteca pública de IOWA nos Estados Unidos da América em 2008, terramoto no Haiti em 2010, chuva de granizo que atingiu a Universidade de Calgary no Canadá em 2011, entre outros casos. É possível concluir-se que não existe nenhum lugar no planeta que possa garantir a 100% a proteção dos conteúdos digitais existentes.

Num contexto de conteúdos digitais mais específicos, os *datasets*, a perda dos mesmos pode causar repercussões enormes. Como já foi referido anteriormente, a preservação e acessibilidade futura de *datasets* é um tópico de extrema importância, visto que os mesmos podem possuir informação de elevado valor ou de grande importância histórica. Além disso, existem *datasets* que possuem dados de observações únicas, e como tal será impossível a reprodução dos mesmos no futuro.

A Universidade do Porto, como já foi referido na secção 2.3, encontra-se a desenvolver um protótipo de um repositório de dados científicos com o intuito de garantir a preservação digital de *datasets* e sua acessibilidade futura. Porém, também não consegue garantir que o servidor contendo o repositório de dados científicos, esteja totalmente

[12] Mais informações sobre a Biblioteca de Alexandria acessíveis em: http://www.historiadomundo.com.br/curiosidades/a-biblioteca.htm

isento de todo tipo de acidentes e catástrofes naturais que possam causar a perda da informação armazenada e preservada, que nele se encontra.

Além disso, a ação humana também é susceptível a falhas e, involuntariamente pode causar sérios danos no servidor que contém o repositório de dados. Por outro lado, as componentes de *hardware* do servidor estão suscetíveis a uma enorme variedade de falhas, algumas delas que podem não permitir temporariamente o acesso aos dados preservados, mas noutros casos mais extremos poderá significar a perda de todos os dados contidos no servidor.

Deduz-se que a possível ocorrência destes acontecimentos, coloca em elevado risco a correta preservação a longo prazo dos dados gerados em contexto científico, e como tal, é um problema bastante crítico que deve ser abordado em qualquer processo de curadoria desta área.

2.5 Geração e Preservação de Cópias de Segurança

Com o intuito de solucionar o problema detetado na secção anterior, secção 2.4, procedeu-se à elaboração e implementação da solução abaixo descrita.

Inicialmente procedeu-se à instalação de uma instância de um repositório de conteúdos digitais, tendo-se optado pela utilização de um repositório DSpace, visto que os investigadores da Universidade do Porto já estão familiarizados com esta plataforma, devido à existência atual de dois repositórios ativos, referidos anteriormente na secção 2.3. Além disso, a Universidade do Porto também se encontra a utilizar um repositório DSpace, para a criação de um repositório de dados científicos.

Pretende-se a análise da instância DSpace instalada, de modo a criar-se um processo para a geração de cópias de segurança dos conteúdos existentes na mesma, de forma a assegurar uma via para a preservação dos conteúdos existentes. Uma descrição mais elaborada sobre esta plataforma encontra-se na secção 3.1.

Na eventualidade da ocorrência de algum dano na instância DSpace, e consequentemente também nos conteúdos digitais armazenados, é importante a existência de um meio para a recuperação dos mesmos através das cópias de segurança geradas anteriormente. Deste modo, consegue assegurar-se que um dado conteúdo digital é sempre passível de recuperação caso exista uma cópia de segurança do mesmo.

Contudo, as cópias de segurança geradas também estão sujeitas a sofrerem danos, ou até mesmo perderem-se, caso ocorra algum dano no suporte de armazenamento físico onde as mesmas se encontram guardadas. Tendo em conta a possibilidade de ocorrência deste tipo de situação, nota-se que a geração das cópias de segurança não é suficiente para assegurar uma preservação, uma vez que uma recuperação futura será impossível caso as cópias de segurança necessárias se encontrem danificadas ou perdidas.

Como tal, é vital que também seja assegurado a preservação das cópias de segurança geradas. Uma das soluções existentes é a criação de várias réplicas das cópias de segurança, fazendo a distribuição das mesmas por diferentes localizações geográficas. Assim, pretende-se assegurar que duas localizações geográficas contendo a mesma réplica, não sofram paralelamente danos provenientes do mesmo acontecimento.

Além disso, de modo a que seja assegurada a integridade das réplicas da cópia de segurança, é aconselhado no mínimo a criação de 3 réplicas por cada cópia de segurança, para que se possa garantir a correta deteção e correção da réplica danificada. De notar que a existência de 3 réplicas de segurança só permite uma preservação robusta caso só ocorra uma falha numa das réplicas. Por exemplo, se o resultado da comparação de duas réplicas supostamente idênticas não for positivo, existindo a terceira réplica, poder-se-á identificar qual das réplicas se encontra errada e proceder-se à sua reparação. Por outro lado, no caso de existência de três réplicas distintas, não é possível acontecer uma recuperação, visto que é impossível identificar qual delas é a correta. Como tal, quanto maior for o número de réplicas existentes de um dado *dataset*, maior será a garantia de integridade [SS10, TW11].

Com o intuito de solucionar o problema detetado procedeu-se à análise de algumas tecnologias e ferramentas. Uma descrição mais elaborada pode ser visualizada no Capítulo 3. Por último, procedeu-se à criação do sistema de preservação e a solução adoptada pode ser consultada no Capítulo 4.

Capítulo 3

Tecnologias e Ferramentas

No âmbito deste projeto de dissertação, com vista à resolução dos problemas mencionados na secção 2.4, foram utilizadas diferentes tecnologias e ferramentas.

Todo o sistema desenvolvido foi implementado e testado num cenário de um repositório de conteúdos digitais. Sendo que se encontra em desenvolvimento um protótipo de um repositório de dados científicos, no âmbito de um projeto denominado UPData [dSRL], já anteriormente referido na secção 2.3, optou-se por implementar o sistema na mesma tecnologia que o protótipo do UPData, ou seja, num repositório digital DSpace. Uma descrição mais elaborada sobre esta tecnologia encontra-se na secção 3.1.

Além disso, um dos principais fatores que proporcionou a escolha da plataforma DSpace, e não uma das outras opções existentes no mercado, é que os investigadores da Universidade do Porto já estão familiarizados com a interface desta plataforma, devido à existência atual de dois repositórios implementados em DSpace, denominados de Repositório Aberto da Universidade do Porto[1] e Repositório Temático.

De modo a assegurar-se uma melhor preservação dos conteúdos existentes na instância DSpace, recorreu-se a serviços *cloud*, tendo-se optado pela utilização do serviço de armazenamento *cloud* fornecido pela Amazon. Uma descrição mais promenorizada sobre este serviço encontra-se na secção 3.2.1.

A biblioteca jclouds tornou-se uma prioridade, de modo a uniformizar a ligação com os diferentes serviços de armazenamento *cloud* existentes no mercado. Uma descrição mais pormenorizada sobre a biblioteca jclouds encontra-se na secção 3.3.

[1] Mais informações sobre o Repositório Aberto da Universidade do Porto acessíveis em: http://repositorio-aberto.up.pt/

3.1 DSpace

DSpace é um software *open-source*, que suporta o *Open Archive Initiative Protocol for Metadata Harvesting*[2] (OAI-PMH), e foi projetado de modo a suportar trocas de informação com outros repositórios de conteúdos digitais, implementados neste ou noutro repositório *open-source*. O DSpace também usa a norma *Dublin Core*[3], como formato padrão da representação dos metadados de todos os conteúdos existentes.

O DSpace foi o resultado de uma parceria entre o MIT Libraries (MIT) e a Hewlett-Packard (HP), tendo sido lançado em 2002, com o intuito de disponibilizar sistemas de repositórios, para o armazenamento de documentos digitais, destinados à educação ou provenientes de investigação científica [RSRF10, Sin07].

Em 2007, o MIT e a HP criaram a DSpace Foundation, uma organização sem fins lucrativos para promover a plataforma e suportar os seus utilizadores. Em 2009, o suporte aos utilizadores ficou ao cargo de outra organização sem fins lucrativos, a DuraSpace Foundation[4] [RSRF10].

Atualmente estima-se que a plataforma DSpace contenha cerca de 1000 utilizadores em todo o mundo, sendo alguns deles instituições portuguesas de renome, como por exemplo: Universidade do Porto; Universidade de Lisboa; Universidade de Coimbra; e Universidade de Trás-os-Montes e Alto Douro [Orga].

O DSpace torna fácil o processo de criação de repositórios institucionais, que permitem a recolha, partilha e preservação digital de conteúdos intelectuais. A plataforma permite também o armazenamento de uma grande variedade de formatos digitais, como por exemplo: *datasets*, imagens, ficheiros de áudio, ficheiros de vídeo, programas de computador, entre outros. Para além disso, o DSpace fornece um vasto conjunto de ferramentas, para ajudar as instituições na gestão dos seus conteúdos digitais. Outra vantagem existente é que no caso de um formato digital preservado tornar-se obsoleto, é possível a migração do mesmo para um novo formato digital existente [RSRF10, Pru05].

Por último, um ponto forte da utilização de um repositório DSpace é a enorme capacidade de personalização do mesmo às necessidades das organizações, principalmente nos casos de organizações grandes e complexas, que fazem a submissão de uma grande quantidade de conteúdos digitais, provenientes dos mais diversificados departamentos [Orgb].

Nas próximas secções, serão abordadas algumas questões e funcionalidades do DSpace. Na secção 3.1.1 poderá ser visualizada a organização estrutural dos conteúdos no DSpace, na secção 3.1.2 encontra-se alguma informação sobre a estrutura da base de dados do DSpace. Na secção 3.1.3 poderão ser encontrados diferentes métodos para

[2]Mais informações sobre *Open Archive Initiative Protocol for Metadata Harvesting* acessíveis em: http://www.openarchives.org/

[3]Mais informações sobre *Dublin Core* acessíveis em: http://dublincore.org/

[4]Mais informações sobre a organização DuraSpace Foundation acessíveis em: http://www.duraspace.org/

a realização de cópias de segurança da instância DSpace e dos seus respetivos conteúdos. Por último, na secção 3.1.4 é especificada a API do DSpace existente para *backup* e restauro.

Resumindo, a plataforma DSpace é uma excelente ferramenta *open-source* para a submissão de conteúdos digitais, pois pode ser facilmente adaptada às necessidades existentes de uma instituição ou organização, e dispõe de um vasto conjunto de ferramentas para a gestão da plataforma e dos conteúdos digitais preservados.

3.1.1 Organização Estrutural

A estrutura organizacional do DSpace é constituída fundamentalmente por comunidades, coleções, itens, pacotes e ficheiros, como pode ser observado na figura 3.1.

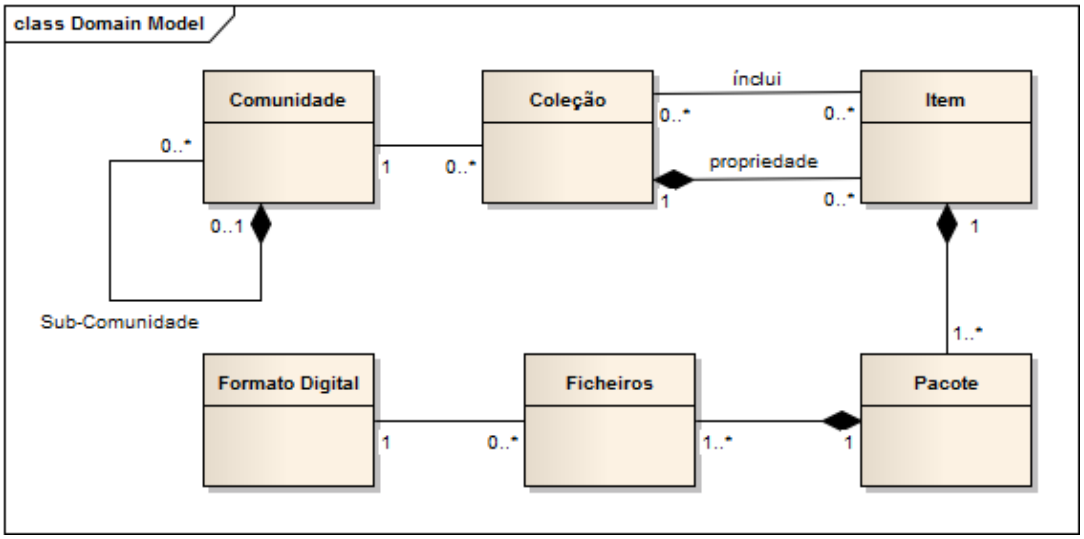

Figura 3.1: Organização estrutural dos conteúdos em DSpace

Cada instância DSpace é organizada em comunidades, que por sua vez poderão ser divididas em subcomunidades, simbolizando deste modo a estrutura de uma dada organização. Por organizações entenda-se por exemplo: departamentos, laboratórios, centros de investigação ou instituições de ensino [Tea11, Orgb].

Cada comunidade é composta por coleções, que poderão ser a representação de um tema, atividade ou projeto inserido na organização [Tea11, Orgb].

Um item é um arquivo com informação associada, facilitando assim a identificação do mesmo e do contexto onde se encontra inserido. Essa informação associada, denominados metadados, encontra-se indexada, para que seja possível fazer-se consultas e pesquisas sobre o item. Um item poderá estar associado a várias coleções, mas só uma delas poderá autodenominar-se proprietária dele. Um item é composto por um ou vários pacotes [Tea11, Orgb].

Entende-se por pacote um conjunto de ficheiros que pertencem a um item. A cada ficheiro existente encontra-se associado um formato digital, para que no futuro, em caso de necessidade, seja possível a migração do ficheiro para um novo formato, caso o formato original se torne obsoleto. De realçar que os conteúdos dos ficheiros também são indexados, caso o formato do ficheiro o permita [Tea11, Orgb].

Numa instância DSpace existem vários utilizadores que poderão aceder e consultar os arquivos digitais armazenados, contudo só poderão aceder aos arquivos presentes nas coleções onde se encontram envolvidos, ou seja, estejam presentes na lista de membros da coleção. Um utilizador pode estar envolvido em uma ou mais coleções e, como tal, as permissões dentro de cada coleção podem variar. Pode-se afirmar que existem três classes gerais de utilizadores [Orgb]:

- *End-user*: pode aceder e consultar os arquivos digitais contidos nas coleções, caso tenha permissões para tal;
- *Collection Curator*: responsável pelos processos de curadoria dos arquivos digitais armazenados, de modo a garantir a sua preservação e acessibilidade futura;
- *Submitter*: utilizador que tem permissões para a inserção de conteúdos digitais em determinadas coleções.

3.1.2 Base de Dados

O DSpace utiliza uma base de dados relacional para o armazenamento da informação existente. Atualmente, o DSpace encontra-se especificado para funcionar com a base de dados PostgreSQL[5] ou Oracle[6]. A utilização de outras bases de dados também é possível. Contudo, poderá existir a necessidade de alguma adaptação ao nível do código *source* do DSpace [Tea11].

A informação armazenada na base de dados é bastante diversificada, desde a estrutura organizacional, conteúdos e respetivos metadados, dados dos utilizadores e permissões, ou o estado dos fluxos de trabalho do sistema ainda por finalizar. Um dos principais motivos da utilização de uma base de dados relacional é a facilidade com que se consegue pesquisar os índices dos registos [Tea11].

O processo de inserção de um conteúdo do tipo item numa instância DSpace é composto por várias fases, não sendo obrigatório que o processo de submissão seja feito na totalidade de uma única vez, como tal, a base dados preserva o estado corrente da submissão, e a isto denomina-se de fluxo de trabalho do sistema por finalizar [Tea11].

[5]Mais informações sobre o software PostgreSQL acessíveis em: http://www.postgresql.org/
[6]Mais informações sobre o software Oracle acessíveis em: http://www.oracle.com/

O pacote `org.dspace.storage.rdbms`[7] disponibiliza um acesso mais simplificado à base de dados, sendo a classe `DatabaseManager`[8] responsável pela execução das chamadas SQL na base de dados e obtenção dos respetivos resultados [Tea11].

A estrutura da base de dados do DSpace pode ser visualizada em pormenor no anexo A.1.

3.1.3 Cópias de Segurança

Como já foi referido anteriormente, um dos principais objetivos do DSpace, além da partilha, é o armazenamento e preservação de uma grande quantidade e diversidade de conteúdos digitais.

Tendo em vista a possibilidade de ocorrência futura de restauros, o DSpace oferece duas soluções para a criação de cópias de segurança: o método tradicional ou o *AIP Backup & Restore*.

Segundo a equipa técnica do DSpace, a escolha do método de criação das cópias de segurança deve ser uma análise interna à organização que possui a instância DSpace, de modo a avaliar as necessidades da mesma. Apesar disso, é aconselhável a utilização dos dois métodos em diferentes períodos, de modo a minimizar as perdas das configurações do DSpace e dos conteúdos armazenados. Por exemplo, uma utilização semanal do método tradicional de modo a evitar perdas nas configurações e personalizações da instância DSpace, enquanto a *AIP Backup & Restore* seria usada diariamente de modo a evitar a perda dos conteúdos armazenados [Tea11].

3.1.3.1 Método Tradicional

O método tradicional é uma das opções existentes, e a mais antiga, de se fazer uma cópia de segurança de uma instância DSpace.

A aplicação deste método implica que sejam realizadas duas cópias de segurança. A primeira delas é relativa a todos os ficheiros associados ao DSpace, existentes no sistema. A outra cópia deve-se à necessidade de realização de um *backup* da informação existente na base de dados do DSpace [Tea11].

A grande vantagem da utilização deste método é que em caso de compatibilidade de *hardware* e *software*, o processo de restauro de toda a instância DSpace é bastante fácil de realizar. Por outro lado, se for necessário fazer o restauro noutro sistema, podem ocorrer incompatibilidades que tornem o restauro difícil ou até mesmo impossível [Tea11].

Apesar de ser aconselhado uma cópia de todo o conteúdo associado com a instância DSpace, ficheiros e base de dados, alguns conteúdos podem ser ignorados. Porém, os

[7] Mais informações sobre a API do pacote *org.dspace.storage.rdbms* acessíveis em: http://maven.mse.jhu.edu/embargo/dspace-1.4.2-api/org/dspace/storage/rdbms/package-summary.html

[8] Mais informações sobre a API da classe `DatabaseManager` acessíveis em: http://maven.mse.jhu.edu/embargo/dspace-1.4.2-api/org/dspace/storage/rdbms/DatabaseManager.html

conteúdos a seguir listados devem estar sempre presentes na cópia de segurança gerada [Tea11]:

- Código *source* do DSpace caso alguma customização tenha sido realizada;
- Diretório `assetstore`, visto que este diretório contém os conteúdos digitais armazenados na instância DSpace;
- Diretório `config`, visto que o mesmo contém os ficheiros com as configurações da instância DSpace;
- Diretório `history` e `log` caso se pretenda a preservação dos registos das atividades;
- *Script* da estrutura e conteúdo da base de dados da instância DSpace.

3.1.4 *AIP Backup & Restore*

O *AIP Backup & Restore* é uma API que vem incorporada no DSpace, desde o lançamento da versão 1.7, disponibilizando ferramentas para a criação de cópias de segurança dos diversos conteúdos existentes numa instância DSpace: *site*, comunidades, coleções e itens [Tea11].

O principal objetivo desta API é garantir a exportação e importação de todos os conteúdos existentes e da sua respetiva informação, tal como, ficheiros, metadados e as relações existentes entre si. A exportação de conteúdos ocorre para ficheiros físicos externos à plataforma, de modo a garantir que em caso de necessidade futura, os restauros possam acontecer sem que existam perdas de informação. O ficheiro criado aquando da exportação de um conteúdo é denominado de AIP, sigla proveniente de *Archival Information Packages* [Tea11].

O ficheiro AIP é um ficheiro que contém o descritivo de um único conteúdo DSpace: comunidade, coleção, item ou *site*. Para se verificar que o conteúdo de dois ficheiros AIP é igual, ficheiros esses provenientes do mesmo arquivo da instância DSpace deve-se comparar os seus valores de *checksum*, e caso sejam iguais significa que não houve alterações no arquivo durante o período em que as cópias de segurança foram efetuadas. O *checksum* é a impressão digital do ficheiro que depende do conteúdo presente nele, calculado através de uma função de dispersão [Tea11].

Normalmente um ficheiro AIP é um ficheiro compactado de formato zip, contendo toda a informação sobre o conteúdo, ficheiros e licenças relacionadas. Toda a informação sobre o conteúdo se encontra num ficheiro XML, estruturado no formato METS, que segundo a definição de *The Library of Congress* é um "padrão para a codificação descritiva,

administrativa e estrutural de metadados relacionados com objetos presentes numa biblioteca digital, usando para isso a linguagem XML"[9]. De realçar que o conteúdo existente num ficheiro AIP varia consoante o tipo de arquivo representado [Tea11].

No caso de o ficheiro AIP representar um *site*, o mesmo só irá conter um ficheiro XML. No conteúdo do ficheiro XML, existirão METS contendo metadados básicos sobre a instância DSpace e identificadores das comunidades de alto-nível, ou seja, comunidades que não sejam subcomunidade de outra comunidade existente. Além disso, o ficheiro XML também contém METS com a informação de todos os utilizadores e grupos existentes na instância DSpace [Tea11].

Se o ficheiro AIP representar uma comunidade também irá conter um ficheiro XML, porém a informação armazenada neste será muito diferente do caso anterior. O conteúdo do XML conterá METS com os metadados da comunidade, e em caso de existência de relações de parentesco com outras comunidades ou coleções, terá os seus identificadores. O XML também terá METS para a identificação de grupos específicos daquela comunidade, e além disso, METS para a correta identificação das permissões existentes. Caso a comunidade contenha um logo, o ficheiro do mesmo existirá no ficheiro AIP [Tea11].

Se o arquivo representado pelo ficheiro AIP for uma coleção também irá conter um ficheiro XML. O ficheiro XML terá METS contendo os metadados da coleção e referências aos itens com ela relacionados, em caso de existência dos mesmos. No XML também estarão METS com informação sobre os grupos específicos daquela coleção e METS contendo as permissões existentes. Em caso de existência de templates de itens, o ficheiro XML também terá METS contendo metadados sobre essas templates. Tal como ocorreu na comunidade, caso a coleção contenha um logo, o ficheiro do mesmo também estará presente no ficheiro AIP [Tea11].

Por último, temos o caso em que o ficheiro AIP representa um item. Neste caso o ficheiro AIP vai conter um ficheiro XML e todos os ficheiros existentes no item, que se encontram armazenados na instância DSpace. O XML vai ser composto por METS com os metadados do item e referências a todos os pacotes e ficheiros armazenados, associados a esse item. O ficheiro XML também vai conter os METS com as permissões do item, e de cada pacote e ficheiro referenciado [Tea11].

A utilização desta API tem como principal vantagem a resolução de muitas complexidades existentes aquando da utilização do método tradicional, como por exemplo a necessidade de sincronização das duas cópias de segurança requeridas. Além disso, a sua utilização traz algumas vantagens à comunidade de utilizadores do DSpace, como por exemplo [Tea11]:

- Facilidade de movimentação de conteúdos entre instâncias DSpace;

[9] Tradução da definição de METS. Disponível em: http://www.loc.gov/standards/mets/ [consultado em: 17 de Junho de 2012]

- Realização de cópias de segurança mais consistentes sobre uma dada hierarquia, em vez de se estar dependente da sincronização da cópia de segurança da base de dados com a cópia de segurança dos ficheiros da instância DSpace;
- Método de extração mais fácil da informação contida na instância DSpace, independentemente do propósito final;
- Fornece um formato normalizado para a migração de hierarquias inteiras (comunidades ou coleções).

3.1.4.1 Exportação

Uma das funcionalidades existentes na *AIP Backup & Restore* é a exportação de conteúdos, ou seja, a criação de cópias de segurança dos diversos conteúdos existentes numa dada instância DSpace: comunidades, coleções, itens e *site*.

Existem dois tipos de exportação possíveis sobre um dado conteúdo DSpace: individual ou hierárquico. No caso de uma exportação individual, só será gerado um ficheiro AIP correspondente ao conteúdo pretendido. No caso da ocorrência de uma exportação hierárquica, além da geração do ficheiro AIP correspondente ao conteúdo pretendido, também são gerados os ficheiros AIP dos conteúdos que se encontram hierarquicamente abaixo desse conteúdo [Tea11].

De realçar, que sempre que ocorre a exportação de um ou vários conteúdos existentes numa instância DSpace, é sempre criado um ficheiro AIP individual para cada um dos conteúdos.

Por exemplo, no caso de exportação hierárquica do *site*, serão exportados para ficheiros AIP todas as comunidades, coleções e itens presentes na instância DSpace. Em caso de uma exportação hierárquica de uma comunidade, será exportada essa comunidade e todas as subcomunidades, coleções e itens com ela relacionada. Relativamente a uma coleção, sempre que ocorra uma exportação hierárquica serão exportados todos os itens presentes nessa coleção e a própria coleção. Por último, em caso de exportação hierárquica de um item, o comportamento será idêntico à exportação individual, ocorrendo unicamente a criação de um ficheiro AIP correspondente a esse item [Tea11].

3.1.4.2 Importação

A outra funcionalidade existente na *AIP Backup & Restore* é a importação de conteúdos para uma instância DSpace, através das cópias de segurança criadas, na forma de ficheiros AIP.

À semelhança da funcionalidade de exportação, é possível a importação individual ou hierárquica sobre um dado ficheiro AIP. No caso de ocorrência de uma importação individual, só será importado para a instância DSpace o conteúdo representado no ficheiro

AIP. Por outro lado, no caso de ocorrência de uma importação hierárquica, será importado para a instância DSpace o conteúdo representado no ficheiro AIP, e o conteúdo de todos os ficheiros AIP relacionados hierarquicamente, caso eles estejam presentes [Tea11].

Por exemplo, na importação hierárquica de um ficheiro AIP de um *site*, será importado o conteúdo desse ficheiro e de todos os ficheiros AIP que representem comunidades, coleções e itens relacionados hierarquicamente. No caso de importação hierárquica de um ficheiro AIP de uma comunidade, será importado o conteúdo desse ficheiro, e de todos os ficheiros AIP das subcomunidades, coleções e itens relacionados hierarquicamente. No caso de importação hierárquica de um ficheiro AIP de uma coleção, será importado o conteúdo desse ficheiro e de todos os ficheiros AIP dos itens relacionados com essa coleção. Por último, em caso de importação hierárquica de um item, só será importado o conteúdo do ficheiro, visto que um item não contém outros conteúdos DSpace hierárquicos abaixo [Tea11].

Para que a importação hierárquica de um dado ficheiro AIP funcione na totalidade é necessário que estejam presentes todos os ficheiros AIP necessários, ou seja, os ficheiros AIP de conteúdos que sejam hierarquicamente descendentes do conteúdo representado [Tea11].

O processo de submissão de ficheiros AIP numa instância DSpace pode ocorrer de vários modos, sendo sempre obrigatório a sua identificação: submissão, restauro ou substituição. O modo submissão faz a inserção do conteúdo de um dado ficheiro AIP, através da criação de um novo registo na instância DSpace. O modo restauro tem como objetivo o restauro de conteúdos que foram eliminados da instância DSpace. Por último, a substituição, altera a informação de conteúdos existentes na instância DSpace, sobrepondo a nova informação recebida através dos ficheiros AIP [Tea11].

3.1.4.3 *AIP Backup & Restore* vs. Método Tradicional

Como já foi salientado anteriormente, é possível a criação de cópias de segurança e o consequente restauro através de dois métodos, utilização da *AIP Backup & Restore* ou através do método tradicional.

Existem algumas diferenças substanciais na utilização destes dois métodos e consequentemente nos resultados obtidos, gerando assim um vasto leque de prós e contras que deverão ser analisados aquando da escolha do método de criação das cópias de segurança, com intuito de assegurar futuros restauros.

Ambos os métodos permitem facilmente a criação de cópias de segurança e restauro de todo o conteúdo de uma instância DSpace, mas ambos têm inconvenientes. O método tradicional obriga a que sejam efetuadas duas cópias de segurança sobre a instância DSpace, uma dela relativa à base de dados e a outra relativa aos ficheiros existentes. Por

outro lado, através da *AIP Backup & Restore* é possível criar uma única cópia de segurança, tendo como único senão que itens que ainda não estejam oficialmente em arquivo, ou seja, estejam em processo de submissão, não serão adicionados à cópia de segurança [Tea11].

Quando não se pensa na geração de uma cópia de segurança global e no seu consequente restauro, mas sim, no contexto de um ou vários conteúdos presentes numa instância DSpace, já temos uma situação completamente diferente. A *AIP Backup & Restore* consegue criar uma cópia de segurança de um ou vários conteúdos presentes na instância DSpace, de modo a que eles possam ser restaurados no futuro sem perda de informação. Pelo contrário o método tradicional é confrontado com um esforço extra nesta tarefa, sendo requerido um grande conhecimento da estrutura da base de dados do DSpace e da organização das pastas, de modo a que se aceda unicamente aos dados correspondentes aos conteúdos DSpace em questão, tanto no processo de criação da cópia de segurança como no do próprio restauro [Tea11].

No DSpace é possível a configuração de recolhas de coleções oriundas de repositórios remotos. Através do método tradicional é possível a preservação das configurações definidas, mas através da *AIP Backup & Restore* tal já não é possível. De realçar, que apesar de não ser possível através da *AIP Backup & Restore* a preservação das configurações definidas, os itens já anteriormente recolhidos serão preservados e restaurados em caso de necessidade [Tea11].

Relativamente a itens que usam esquemas de metadados e campos de dados personalizados, o método tradicional consegue assegurar a correta criação da cópia de segurança e restauro futuro em caso de necessidade. Através do uso da *AIP Backup & Restore* não é possível a preservação dos esquemas de metadados e campos de dados personalizados, e como tal, em caso de necessidade futura de restauro, os esquemas de metadados deverão ser primeiro criados de modo manual na instância DSpace, de modo a que a instância DSpace possa ser capaz de criar os campos de dados personalizados pertencentes a um dado esquema [Tea11].

Por último, com o método tradicional é possível a criação da cópia de segurança contendo as configurações e customizações da instância DSpace, enquanto tal não é possível com a *AIP Backup & Restore* [Tea11].

3.2 Serviços *Cloud*

Os serviços *cloud* disponibilizam recursos a nível de armazenamento e computação.

Estes serviços têm tido um aumento de utilização nos últimos anos, crescimento fomentado por várias motivos. Um dos principais motivos é ocultar a complexidade das infraestruturas necessárias ao cumprimento das necessidades de uma organização, a nível

de armazenamento e computação. Além disso, o acesso aos serviços *cloud* é bastante simples, bastando para isso um computador com acesso à *internet* [SMM09, Car10].

Uma vez que todos os recursos estão disponíveis na *cloud*, não há necessidade da existência de computadores pessoais com grandes recursos computacionais, ou seja, existe uma redução no custo de aquisição de equipamento. Além disso, é possível a qualquer momento adicionar ou retirar recursos computacionais, adaptando assim os recursos disponíveis às necessidades reais da empresa. Por último, os custos associados a um serviço *cloud* não são muito elevados [SMM09].

Existem várias opções de serviço *cloud* disponíveis no mercado, sendo uma delas a Amazon S3, serviço descrito na secção 3.2.1.

3.2.1 Amazon S3

O Amazon S3 é um serviço de armazenamento na *cloud*, que possui uma interface *web* simples, para o armazenamento e obtenção de qualquer ficheiro, independentemente do momento e do local do qual se está a aceder. Este serviço permite que os programadores deixem de lado as preocupações como o armazenamento dos dados, podendo assim se focar na resolução de outros problemas existentes [Ser].

Com vista a que o serviço seja utilizado por qualquer ferramenta de desenvolvimento *web*, o mesmo utiliza interfaces REST e SOAP [Ser].

Este serviço permite o armazenamento, leitura e remoção de objetos de tamanho superior a 1 *byte* e inferior a 5 *terabytes*, sendo ilimitado o número de objetos que podem ser armazenados. Cada objeto encontra-se num dado *container* e contém uma identificação única [Ser].

Um *container* pode estar armazenado em diferentes localizações, como por exemplo: EUA *Standard*; Oregon no Oeste dos EUA; Norte da Califórnia no Oeste dos EUA; Irlanda na Europa; Singapura na Ásia; Tóquio na Ásia; São Paulo na América do Sul. A escolha da localização do *container* deverá ter em conta as necessidades de otimização da latência e a minimização dos custos ou licenças existentes. A não ser que o cliente dê uma ordem direta de transferência, os objetos armazenados numa dada região nunca se movem para outra [Ser].

O serviço Amazon S3 foi construído sobre os seguintes requisitos [Ser]:

- Proteção: o cliente deve conseguir de forma intuitiva assegurar o controlo de quem pode aceder aos seus dados. Também deve ser assegurado a proteção dos dados em trânsito ou em repouso;
- Confiabilidade: dados armazenados com 99.999999999% de durabilidade e 99.99% de disponibilidade. Não podem existir pontos críticos em que possa ocorrer uma falha, sendo que todas as falhas são toleradas ou recuperadas pelo sistema sem que haja algum tempo de inatividade;

- Escalabilidade: o serviço pode ser dimensionado em termos da capacidade de armazenamento, velocidade das solicitações e usuários;
- Rapidez: o serviço deve ser rápido podendo assim oferecer suporte a aplicações de elevado desempenho;
- Económicos: o serviço foi construído com componentes de *hardware* de baixo custo. Todo o *hardware* irá eventualmente falhar num dado momento, porém isso não deve comprometer o funcionamento do sistema;
- Simplicidade: este serviço deve permitir um armazenamento altamente escalável, confiável, rápido e acessível, de uma forma que facilite a utilização para qualquer aplicação, independentemente da localização.

O serviço Amazon S3 é utilizado para oferecer suporte a uma grande variedade de casos de utilização, como por exemplo: armazenamento e distribuição de conteúdo; armazenamento para análise de dados; e *backup*, arquivo e recuperação de desastres [Ser].

3.2.1.1 Segurança dos Dados

Os dados podem estar públicos ou privados e com diferentes direitos atribuídos consoante o utilizador especificado.

O Amazon S3 fornece vários mecanismos de modo a garantir que os dados estão protegidos contra acessos não autorizados: *Identity and Access Management* (IAM); *Access Control Lists* (ACLs); políticas do *container*; e *query string authentication* [Ser].

O IAM permite que organizações possam ter uma única conta Amazon o que possibilita a associação de diferentes utilizadores a uma conta só. Além disso, com o IAM é possível definir quais utilizadores podem aceder a um determinado *container* existente. O ACLs permite definir individualmente permissões sobre objetos existentes na *cloud*. As políticas do *container* permitem adicionar ou remover permissões sobre a totalidade ou parte dos objetos existentes num dado *container*. Com a *query string authentication* é possível compartilhar objetos que estejam presentes na *cloud* através da disponibilização do respetivo URL [Ser].

É possível o *upload* e *download* seguro de dados através da uma ligação SSL encriptada com o uso do protocolo HTTPS. Este serviço fornece ainda várias opções para a criptografia de dados em repouso. Pode-se recorrer ao uso da *Amazon S3 Encryption Client* para uma gestão pessoal das chaves criptográficas, ou uso da *Amazon S3 Server Side Encryption* (SSE) de modo a que o Amazon seja o responsável pela sua gestão [Ser].

Por último, é possível guardar em registos, *logs*, as solicitações efetuadas a um dado *container* ou objetos nele contido, permitindo assim que em caso de necessidade futura, possam ser feitas auditorias sobre as operações realizadas [Ser].

Tabela 3.1: Preços para o armazenamento de dados na *cloud* - *container* na Irlanda

	Armazenamento padrão	**Redundância de armazenamento reduzida**
Primeiro 1 TB/mês	0.0992€ por GB	0.0738€ por GB
Próximos 49 TB/mês	0.0873€ por GB	0.0659€ por GB
Próximos 450 TB/mês	0.0754€ por GB	0.058€ por GB
Próximos 500 TB/mês	0.0715€ por GB	0.05€ por GB
Próximos 4000 TB/mês	0.0635€ por GB	0.0421€ por GB
Mais de 5.000 TB/mês	0.0437€ por GB	0.0294€ por GB

3.2.1.2 Durabilidade e Confiabilidade dos Dados

A Amazon S3 fornece uma infraestrutura de armazenamento extremamente durável, projetada para o armazenamento de dados críticos ou primários. Os objetos são armazenados redundantemente em vários dispositivos de diversas instalações de uma dada região do Amazon S3 [Ser].

De modo a assegurar a durabilidade dos dados, em qualquer operação de colocação de um objeto num dado *container*, o serviço armazena o respetivo objeto nas diversas localizações antes de retornar que a operação foi realizada com sucesso [Ser].

Visto que os objetos já se encontram armazenados na *cloud*, o serviço mantém a sua durabilidade ao detetar e reparar rapidamente qualquer redundância perdida. Além disso, este serviço também verifica a integridade dos dados armazenados recorrendo ao uso do *checksum*. Caso seja detetado que um dado se encontra corrompido, ele é automaticamente reparado através das redundâncias existentes [Ser].

O armazenamento padrão do Amazon S3 é projetado de modo a garantir 99.999999999% de durabilidade e 99.9% de disponibilidade dos objetos. Para além disso, foi pensado de modo a suportar a falha simultânea de duas instalações [Ser].

É possível o armazenamento de dados na *cloud* com redundância reduzida, o denominado armazenamento de redundância reduzida (RRS). A opção RSS armazena os objetos em vários dispositivos de diversas instalações, mas não replicará na mesma quantidade que o armazenamento padrão da Amazon, fazendo com que os custos sejam mais baixos. Esta é uma excelente opção para o armazenamento de dados reproduzíveis e não críticos [Ser].

O armazenamento com redundância reduzida é projetado de forma a garantir 99.9% de durabilidade e 99.9% de disponibilidade dos objetos. Este tipo de armazenamento foi projetado de modo a suportar a falha em uma única instalação [Ser].

3.2.1.3 Custos

Em termos de custos, no Amazon S3 só é cobrado pelo que se usa, e como tal não existem taxas mínimas [Ser]. Na tabela 3.1, pode visualizar-se os preços praticados no

armazenamento de dados sendo que se considerou que o *container* estaria localizado na Irlanda.

Os preços praticados consoante o número de solicitações efetuadas à *cloud* podem ser visualizados na tabela 3.2, tendo-se também considerado que o *container* estaria localizado na Irlanda. De realçar que solicitações para a remoção de ficheiros da *cloud* não são cobráveis.

Tabela 3.2: Preços das solicitações à *cloud* - *container* na Irlanda

	Preços
Solicitações PUT, COPY, POST ou LIST	0.0079€ por 1000 solicitações
Solicitações GET e todas as outras	0.0079€ por 1000 solicitações

Na tabela 3.3, podem visualizar-se os preços praticados nas transferências consoante o volume de dados transferidos. Neste caso, também se considerou que o *container* estaria localizado na Irlanda.

Tabela 3.3: Preços das tranferências de dados para a *cloud* - *container* na Irlanda

	Preços
Transferência de dados para dentro	
Transferência de todos os dados para dentro	0.0000 por GB
Transferência de dados para fora	
Primeiro 1 GB/mês	0.0000€ por GB
Até 10 TB/mês	0.0953€ por GB
Próximos 40 TB/mês	0.0715€ por GB
Próximos 100 TB/mês	0.0556€ por GB
Próximos 350 TB/mês	0.0397€ por GB
Próximos 524 TB/mês	—
Próximos 4 PB/mês	—
Maior que 5 PB/mês	—

3.3 jclouds

O jclouds é uma biblioteca *open-source* sob a licença da Apache[10], que facilmente pode ser incorporada em aplicações [JCLa].

Esta biblioteca permite a ligação com diversos serviços *cloud*, que proporcionam tanto recursos de armazenamento como de computação, provenientes de diferentes fornecedores. Através da utilização desta biblioteca é possível uniformizar a ligação, contudo sem que se percam as potencialidades e flexibilidade características de cada um dos serviços [Clo, Cha12].

[10]Mais informações sobre a fundação Apache acessíveis em: http://www.apache.org/

A biblioteca jclouds é conhecida por ter as seguintes características: interface simples, portabilidade, capacidade de lidar com complexidade *web*, testabilidade; performance aceitável; e cobertura de vários serviços *cloud*, atualmente mais de 30, provenientes de diversos fornecedores [Cha12, Haz10].

O jclouds é atualmente utilizado no suporte de plataformas de grandes empresas, como por exemplo: Adobe, CloudBees, enStratus e Red Hat [Clo].

Na secção 3.3.1 encontra-se a descrição de uma das API's presentes no jclouds, denominada Blobstore.

3.3.1 BlobStore

A BlobStore é uma das API's existentes na biblioteca jclouds, responsável pela manipulação de conteúdos armazenados na *cloud* de vários fornecedores. Esta API oferece a possibilidade de ligação assíncrona ou síncrona, assim como o acesso aos conteúdos armazenados através de uma estrutura do tipo `map` [Haz10, JCLb].

A API BlobStore é bastante simples, mas como em outras componentes do jclouds, em caso de necessidade, existem meios para se ganhar o acesso a ferramentas específicas do serviço [JCLb].

Em suma, a API BlobStore permite a manipulação de conteúdos existentes na *cloud*, tendo como principais funções: envio de conteúdos para a *cloud*; receção de conteúdos da *cloud*; receção e manipulação da informação dos conteúdos existentes na *cloud* [JCLb].

Um objeto *blobstore* é sempre necessário quando se pretende manipular um conteúdo para, ou da *cloud*, sendo que uma representação do mesmo pode ser visualizada na figura 3.2. Como se pode observar na figura, um objeto *blobstore* é composto pelos seguintes atributos: serviço, *container* e conteúdo.

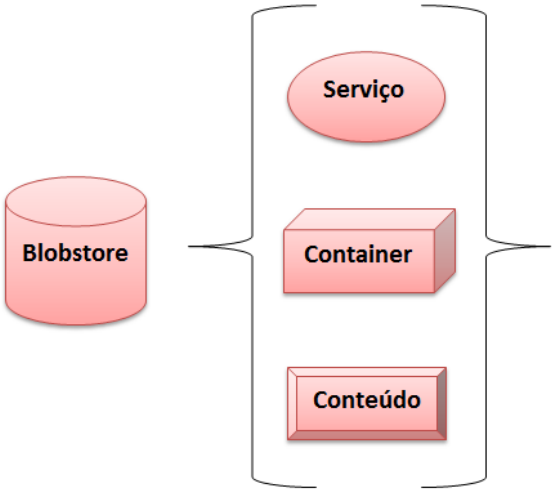

Figura 3.2: Estrutura do objeto `blobStore`

O atributo serviço representa o fornecedor a que o objeto *blobstore* está associado. Existem várias opções existentes, entre elas: Amazon; VMWare, AZure; e Rackspace [JCLb].

O atributo *container* é o endereço da localização do conteúdo. Por exemplo, no Amazon S3 um *container* é denominado de *bucket* e o seu nome deverá ser único. Em outros serviços existentes, a atribuição do nome de um *container* é menos restrito [JCLb].

Através do objeto *blobstore*, é possível obter-se a listagem de todos os *containers* existentes em dado serviço e dos conteúdos existentes neles. Esses conteúdos podem ser de vários tipos: *blob*, pasta ou *virtual path*;

O objeto *blob* é a representação de um dado ficheiro armazenado num *container* de um dado serviço de armazenamento *cloud* e a sua respetiva informação. A pasta é uma subdivisão de um *container*, que por sua vez pode conter outras pastas e ficheiros. Um *virtual path* pode ser um marcador ou prefixo, que é usado unicamente para dar a aparência de uma estrutura hierárquica [JCLb].

Capítulo 4

Sistema de Preservação

Como já foi salientado anteriormente, os repositórios de conteúdos digitais permitem a preservação e acessibilidade futura de conteúdos dos mais diversificados formatos digitais. Contudo, acontecimentos imprevistos de origem natural ou até mesmo humana podem provocar a ocorrência de danos ou até mesmo a perda dos conteúdos digitais existentes num dado repositório. Sendo assim, com o intuito de resolução do problema identificado na secção 2.4, procedeu-se à elaboração de um sistema de preservação.

Como já foi mencionado na secção 2.5, para que fosse possível testar o sistema de perservação implementado, procedeu-se à instalação de um repositório de conteúdos digitais, tendo-se optado pela instalação de uma instância DSpace, devido à grande proximidade que os investigadores da Universidade do Porto têm com esta plataforma.

O problema mencionado na secção 2.4 é relativo à necessidade de geração das cópias de segurança dos conteúdos existentes numa dada instância DSpace, para que em caso de necessidade futura, seja possível a ocorrência da restauração de conteúdos, através das cópias de segurança geradas. Com esse fim, desenvolveu-se o componente de *backup* para a geração das cópias de segurança, e o componente de importação para a restauração dos conteúdos, componentes essas explicadas ao pormenor nas secções 4.1 e 4.4, respetivamente.

Com a geração das cópias de segurança, surgiu outra necessidade, relativa à preservação das cópias de segurança geradas, de modo a garantir-se a sua disponibilidade e integridade, para que as mesmas possam ser utilizadas no futuro em caso de necessidade. Uma das soluções existentes consistia na criação de várias réplicas das cópias de segurança, para que as mesmas fossem distribuídas por diferentes localizações geográficas. Além disso, seria necessário a implementação de um mecanismo que assegura-se a integridade das réplicas existentes, fazendo a comparação das mesmas entre si, com o intuito

de identificar as réplicas danificadas, para que as mesmas pudessem ser substituídas por uma réplica fiel.

Sendo que os serviços de armazenamento *cloud* conseguem responder às necessidades acima referidas, relativas à preservação das cópias de segurança geradas, optou-se pela utilização de um dos serviços de armazenamento *cloud* existentes atualmente no mercado. Sendo assim, foram desenvolvidas mais dois componentes, um componente responsável pelo envio das cópias de segurança para a *cloud* e outro componente responsável pela obtenção da *cloud* das cópias de segurança preservadas. Os componentes encontram-se detalhados na secção 4.2 e 4.3, respetivamente.

Através deste sistema de preservação, consegue-se assegurar a preservação dos conteúdos existentes numa dada instância DSpace, com o recurso a cópias de segurança, assegurando assim que é possível a qualquer momento a utilização dessas cópias de segurança para o restauro de conteúdos.

4.1 Componente de *Backup*

Um dos componentes do sistema de preservação é o componente de *backup*, responsável pela criação das cópias de segurança dos conteúdos existentes numa instância DSpace: comunidades, coleções, itens ou *site*.

Inicialmente foram analisados os métodos para a criação de cópias de segurança dos conteúdos existente numa instância DSpace, tendo-se optado pela utilização da API *AIP Backup & Restore*. Uma descrição mais elaborada pode ser visualizada na secção 4.1.1.

Tendo sido escolhida a API *AIP Backup & Restore* procedeu-se à análise das funcionalidades disponibilizadas pela mesma, no contexto de criação de cópias de segurança dos conteúdos de uma instância DSpace. Para uma análise mais detalhada dos pormenores da API consultar a secção 4.1.2.

A criação da cópia de segurança, de um conteúdo existente numa instância DSpace, só deverá ocorrer sobre as seguintes condições: cópia de segurança inexistente; ocorrência de alguma modificação no conteúdo; ficheiro AIP proveniente da geração da cópia de segurança encontra-se danificado. A verificação da existência e consistência do ficheiro AIP é bastante simples, contudo a verificação da ocorrência de uma modificação num dado conteúdo já não é tão elementar, visto que a base de dados do DSpace não preserva a data da última modificação para todo o tipo de conteúdo do DSpace. Para mais detalhes sobre este assunto, visualizar a secção 4.1.2.

A propósito de uma gestão mais eficaz das cópias de segurança geradas, é importante o registo das mesmas, como tal, foi criada uma classe que permite fazer a inserção, atualização e eliminação de registos relativos à geração das cópias de segurança. Mais detalhes sobre esta funcionalidade podem ser encontrados na secção 4.1.4.

Com o intuito de evitar a criação de cópias de segurança desnecessárias, foi elaborado um algoritmo para a verificação da existência de cópias de segurança, e se as mesmas ainda se encontram atualizadas e íntegras. Mais detalhes sobre este algoritmo podem ser encontrados na secção 4.1.5.

Por último, procedeu-se à implementação do sistema de *backup*, tendo em conta todas as análises efetuadas e a novas funcionalidades implementadas. Mais detalhes sobre o sistema de *backup* implementado podem ser encontrados na secção 4.1.6.

4.1.1 Escolha do Método de *Backup*

Como já foi referido anteriormente, na secção 3.1.3, existem duas opções para a realização de cópias de segurança de conteúdos existentes numa instância do DSpace: modo tradicional ou uso de uma API denominada de *AIP Backup & Restore*.

Após a análise das diferenças existentes entre os dois métodos, e enquadramento das respetivas características no âmbito deste projeto de dissertação, optou-se pela utilização da *AIP Backup & Restore*.

O principal motivo de a escolha ter recaído sobre a API, deve-se ao fato de a mesma permitir manipular os conteúdos de uma instância DSpace, como individualidades, sem ser necessário conhecer muito pormenorizadamente a estrutura e organização do DSpace e da própria base de dados, além de que, esta API exporta os metadados dos conteúdos num formato padrão METS, facilitando assim possíveis importações futuras para instâncias DSpace, como para outras plataformas de armazenamento e preservação de conteúdos digitais.

Contudo, é aconselhável que pelo menos uma vez, de preferência após a instalação e configuração de uma instância DSpace, se proceda à criação de uma cópia de segurança pelo método tradicional, evitando assim futuramente possíveis perdas de informação sobre as configurações. Assim, a recuperação de uma instância DSpace será um processo mais rápido e simples, pois não será necessário voltar a efetuar todo o processo de configuração e instalação. Para mais pormenores, sobre as diferenças entre os dois métodos para a criação de cópias de segurança, consultar a secção 3.1.4.3.

4.1.2 Análise da *AIP Backup & Restore*

Como já foi referido anteriormente, na secção 3.1.4, esta API permite a exportação e importação de conteúdos de uma instância DSpace: comunidades, coleções, itens ou *site*. Por exportação, entenda-se a criação de cópias de segurança de conteúdos existentes na instância DSpace, para que as mesmas possam ser utilizadas no futuro, em caso de necessidade de recuperação de conteúdos que se encontrem danificados ou perdidos.

No contexto de criação de cópias de segurança, foi notada a existência de duas variantes de exportação dos conteúdos: individual e hierárquica. A variante individual faz

unicamente a exportação de um dado conteúdo existente na instância DSpace, enquanto que a variante hierárquica, além de exportar o conteúdo, também faz a exportação de todos os outros conteúdos que estejam hierarquicamente abaixo desse conteúdo, ou seja, conteúdos que na hierarquia do DSpace sejam considerados filhos de um dado conteúdo. A função da API responsável por uma exportação individual encontra-se denominada de `disseminate()`, enquanto a responsável pela exportação hierárquica é denominada de `disseminateAll()`. Estas funções além de gerarem as cópias de segurança dos conteúdos presentes, também asseguram que os ficheiros gerados possam ser utilizados para a partilha desses conteúdos com outras instâncias DSpace.

De realçar que aquando da exportação de um dado conteúdo existente numa instância DSpace, independentemente de ser uma exportação individual ou hierárquica, é criado um ficheiro AIP contendo os metadados desse conteúdo e os respetivos ficheiros. Para mais informações sobre a estrutura de um ficheiro AIP e sobre as diferenças existentes consoante o tipo de conteúdo do DSpace sobre o qual foi gerado o ficheiro, consultar a secção 3.1.4.

A análise de diferentes ficheiros AIP, resultantes de diferentes chamadas à função `disseminate()`, comprovou que a mesma efetua uma exportação correta dos metadados e, em caso de existência, também dos respetivos ficheiros, de um dado conteúdo existente numa instância DSpace.

A análise realizada sobre a função `disseminateAll()` comprovou que a mesma também funciona como especificado, percorrendo todos os conteúdos hierarquicamente e fazendo a criação individual de cada um dos respetivos ficheiros AIP. Porém, foi notado que caso já exista uma cópia de segurança atualizada de um dado conteúdo, a mesma é ignorada, pois a função simplesmente faz a exportação de toda a hierarquia sem considerar as cópias de segurança já realizadas. Mesmo que a exportação ocorra para um diretório onde já exista o respetivo ficheiro AIP, o ficheiro existente é ignorado, sendo recriada a cópia de segurança.

Em instâncias DSpace com poucos conteúdos, este processo poderá não ser muito problemático, contudo em instâncias com bastantes conteúdos, centenas ou até mesmo milhares, alguns conteúdos poderão atingir *gigabytes* em termos de tamanho, o tempo de processamento poderá ser muito longo. Esta situação não pode ser evitada, mas pode-se reduzir substancialmente o tempo de processamento, não se fazendo a criação de algumas cópias de segurança, associados a conteúdos existentes numa instância DSpace, sobre os quais já existe a cópia de segurança atualizada.

Com o intuito de redução do processamento e tempo exigido na criação das cópias de segurança, a geração de uma cópia de segurança de um dado conteúdo existente numa instância DSpace, só deverá ocorrer quando:

- Não existe a cópia de segurança;

- Ocorreu alguma modificação do conteúdo;
- O ficheiro AIP da cópia de segurança encontra-se danificado.

O processo de verificação da existência de uma cópia de segurança e integridade da mesma é bastante fácil, bastando para isso conferir a existência do ficheiro localmente e comparação do MD5 atual com o MD5 gerado aquando da criação do ficheiro AIP.

Por outro lado, a verificação da ocorrência de uma modificação num conteúdo, já se torna um processo mais elaborado, visto que o DSpace só guarda a data da última modificação se o tipo de conteúdo for um item. Sendo assim, de modo a certificar-se da existência de modificações em conteúdos do tipo comunidade ou coleção, tornou-se necessário a existência de registos que identifique em que conteúdos ocorreram modificações. Mais detalhes sobre a solução implementada para esta necessidade, consultar o capítulo 4.1.3.

4.1.3 Registo de Modificações de Conteúdos da Instância DSpace

A base de dados do DSpace, no contexto de conteúdos existentes na instância DSpace, só preserva a data da última modificação, se o tipo do conteúdo dado for um item, como tal, tornou-se prioritário a implementação de um método que permita a preservação da data da última modificação nas comunidades e coleções. Esta preservação é essencial, facilitando a deteção da ocorrência de modificações em comunidades e coleções, podendo-se assim assegurar se uma dada cópia de segurança ainda se encontra atualizada.

Como não é guardado a data da última modificação ocorrida, isto limita a identificação dos conteúdos modificados e como tal, pretende-se que existam registos com as comunidades e coleções que sofreram modificações. Para evitar modificações das tabelas existentes, optou-se pela criação de uma nova tabela na base de dados do DSpace, tabela essa denominada de `logModifications`.

A tabela `logModifications` tem como missão registar quais são as comunidades e coleções que sofreram modificações, de modo a controlar que cópias de segurança poderão estar desatualizadas, ou seja, comunidades e coleções onde ocorreu uma modificação e nenhuma cópia de segurança foi criada posteriormente. Esta tabela será composta pelos seguintes campos:

- `id`: identificador único do registo da tabela;
- `object_id`: identificador do conteúdo DSpace;
- `type`: tipo de conteúdo DSpace (comunidade ou coleção);
- `handler`: identificador universal do conteúdo DSpace;
- `action`: tipo de modificação ocorrida (inserção, atualização, eliminação);
- `last_modification`: data em que ocorreu a última modificação.

Sempre que ocorrer alguma modificação numa determinada comunidade ou coleção, esse acontecimento deverá ficar registado na tabela `logModifications`. Caso já exista na tabela um registo associado a esse conteúdo, acontecerá uma atualização do mesmo, caso contrário, deverá ser inserido um novo registo. Todos os campos acima referidos devem ser preenchidos, sendo que o campo `action` deverá conter a operação ocorrida: inserção, atualização ou remoção.

Quando ocorrer a geração da cópia de segurança de um determinado conteúdo DSpace, caso exista um registo na tabela `logModifications` associado a esse conteúdo, o registo deve ser removido, visto que o objetivo desta tabela é manter o registo de comunidades e coleções sobre as quais ocorreu uma modificação e ainda não foi gerada posteriormente a respetiva cópia de segurança.

Para que se possa registar, consultar e gerir de forma intuitiva as modificações ocorridas em comunidades e coleções, foi criada uma classe denominada `modificationsRegistry`. Esta classe tem as seguintes funções principais:

- `existLog()`: responsável por verificar se na tabela `logModifications` existe algum registo que se encontre associado a uma dada comunidade ou coleção;
- `updateLog()`: responsável por fazer a inserção ou atualização de um registo na tabela `logModifications`, registo esse que se encontra associado a uma certa comunidade ou coleção;
- `deleteLog()`: responsável por apagar um registo da tabela `logModifications` que se encontre associado a uma dada comunidade ou coleção.

Com o intuito de minimizar as alterações ao código, aquando da integração desta nova funcionalidade numa instância DSpace, tentou-se efetuar as mudanças do código *source* ao nível da classe `DatabaseManager`[1], classe responsável pela execução das chamadas SQL na base de dados.

O objetivo é efetuar o registo das modificações, aquando da ocorrência de uma chamada SQL à base de dados do DSpace, do tipo *insert*, *delete*, ou *update*, que correspondem respetivamente a chamadas de inserção, remoção e atualização. Esse registo só ocorreria caso as chamadas fossem efetuadas sobre as tabelas *community* ou *collection*, tabelas essas responsáveis pelo armazenamento da informação geral de comunidades e coleções, respetivamente. Esta abordagem teve que ser abandonada, porque ao nível da classe `DatabaseManager` não ocorre nenhuma atualização da base de dados. Por exemplo, durante o processo de criação de uma comunidade, são introduzidos registos em várias tabelas do DSpace, mas a atualização da base de dados só ocorre verdadeiramente no

[1] Mais informações sobre a API da classe `DatabaseManager` acessíveis em: http://maven.mse.jhu.edu/embargo/dspace-1.4.2-api/org/dspace/storage/rdbms/DatabaseManager.html

final de todo o processo, como tal, é impossível aceder a um dos campos necessários ao registo, o campo `handler`, visto que o mesmo só será realmente criado na base de dados do DSpace aquando da atualização da mesma.

Tendo sido detetado o problema acima referido, teve-se que optar por efetuar as modificações do código ao nível dos *controllers*, após ter sido efetuada a chamada de atualização da base de dados do DSpace. Ou seja, sempre que ocorra a chamada de uma função com o intuito de inserção, atualização ou remoção de uma comunidade ou coleção da instância DSpace, logo após ter sido efetuado a atualização dessa operação na base de dados do DSpace, deverá ser efetuada uma chamada à base de dados, através da classe `modificationsRegistry`, com o intuito de atualizar ou inserir o registo da ocorrência de modificação num dado conteúdo da instância DSpace.

4.1.4 Registo das Cópias de Segurança

O registo da geração das cópias de segurança é essencial para uma gestão mais eficaz das mesmas, tanto para efeitos de verificação da sua existência, como para efeitos de validação da integridade dos ficheiros AIP existentes.

De modo a assegurar a preservação dos registos referentes à geração das cópias de segurança, foi criada uma nova tabela na base de dados do DSpace, denominada de `logBackup`. Esta tabela tem como principal objetivo, o armazenamento da informação proveniente da última operação de geração da cópia de segurança, para cada um dos conteúdos da instância DSpace. A tabela será composta pelos seguintes campos:

- `id`: identificador único do registo da tabela;
- `object_id`: identificador do conteúdo DSpace;
- `type`: tipo de conteúdo DSpace (comunidade, coleção ou item);
- `handler`: identificador universal do conteúdo DSpace;
- `last_backup`: data em que ocorreu a criação da cópia de segurança que se está a registar;
- `md5`: MD5 da cópia de segurança aquando da sua geração.

Sempre que ocorrer um processo de geração da cópia de segurança de um dado conteúdo existente numa instância DSpace, esta ação ficará registada na tabela `logBackup`. Caso já exista um registo associado ao conteúdo, o registo será atualizado, senão será inserido um novo registo, mantendo-se assim um registo único por cada conteúdo existente.

Para que se possa registar, consultar e manipular de forma intuitiva os registos existentes dos processos de geração das cópias de segurança, foi criada uma classe denominada `backupProcess`. Esta classe tem as seguintes funções principais:

- `updateProcessBackup()`: responsável pela inserção ou atualização de um registo na tabela `logBackup`, registo esse que se encontra associado a um conteúdo da instância DSpace;
- `existRegist()`: verifica se um dado conteúdo tem um registo associado na tabela `logBackup`;
- `getSavedMD5()`: devolve o MD5 do ficheiro AIP gerado na última cópia de segurança efetuada, correspondente a um dado conteúdo. Este MD5 foi preservado mal se finalizou o processo de geração do respetivo ficheiro;
- `getLastBackupDate()`: devolve a data da criação da última cópia de segurança de um dado conteúdo, caso o registo exista.

Deste modo, é fácil e simples manter e controlar o registo e consequentemente as cópias de segurança dos conteúdos de uma instância DSpace: comunidades, coleções e itens.

4.1.5 Algoritmo de Confirmação de *Backup*

A geração de uma cópia de segurança de um dado conteúdo existente numa instância DSpace, só deverá ocorrer perante a ocorrência das seguintes situações: inexistência da cópia de segurança; cópia de segurança desatualizada; ficheiro da cópia de segurança encontra-se danificado. Com o intuito de evitar a criação desnecessária de cópias de segurança dos conteúdos de uma instância DSpace procedeu-se à elaboração de um algoritmo.

Como já foi referido, na secção 4.1.3, só o conteúdo do tipo item é que guarda a data da última modificação. Como tal, foi necessário a criação da tabela `logModifications`, para que se possa controlar a ocorrência de modificações em comunidades e coleções. Com esse intuito, nesta tabela só estarão presentes as comunidades e coleções em que ocorreram modificações, não tendo havido posteriormente a geração de nenhuma cópia de segurança. Sendo assim, o processo de verificação da existência de uma cópia de segurança atualizada de um dado conteúdo será igual caso o conteúdo seja do tipo comunidade ou coleção, mas diferente no caso de ser do tipo item.

Na figura 4.1, pode-se visualizar a representação do algoritmo que verifica se o ficheiro da cópia de segurança, de um dado conteúdo existente na instância DSpace, existe e se encontra atualizado, correto e íntegro.

Inicialmente verifica-se se o conteúdo existente na instância DSpace é do tipo comunidade ou coleção, ou se por outro lado é do tipo item. Caso o conteúdo DSpace seja uma comunidade ou coleção, verifica-se se existe algum registo na tabela `logModifications` associado com conteúdo. Caso exista, isto significa que ocorreu uma modificação desse conteúdo, e como tal, pode ser efetuada uma cópia de segurança.

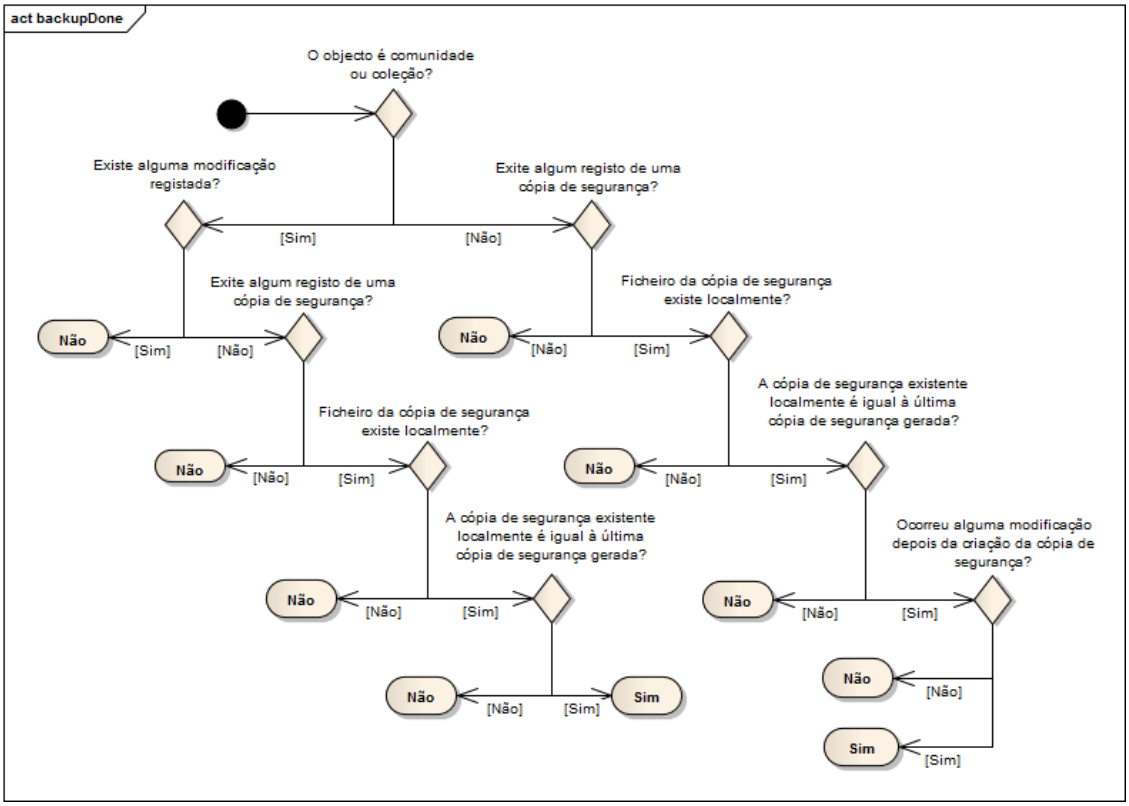

Figura 4.1: Algoritmo de confirmação da existência de um dado *backup*.

Por outro lado, caso não exista um registo associado na tabela `logModifications`, significa que não ocorreu nenhuma modificação, mas isso não assegura que a cópia de segurança exista e que esteja íntegra.

A seguir, independentemente do tipo do conteúdo dado é verificado se existe algum registo na tabela `logBackup` associado com o conteúdo. Se existir o registo da criação da cópia de segurança, é verificado se o ficheiro AIP ainda existe localmente, e se o MD5 do ficheiro é igual ao MD5 preservado aquando da geração do ficheiro AIP. Caso não exista o registo da criação da cópia de segurança ou não exista localmente o ficheiro AIP gerado pode-se efetuar uma cópia de segurança do conteúdo. Além disso, também se pode efetuar uma cópia de segurança do conteúdo, caso o respetivo ficheiro AIP exista localmente e o MD5 do mesmo seja diferente do MD5 guardado aquando da geração da última cópia de segurança daquele conteúdo. A situação descrita anteriormente pode ocorrer nos seguintes casos: o ficheiro AIP existente localmente encontra-se danificado ou corresponde a um *backup* antigo.

Se o conteúdo dado for do tipo comunidade ou coleção, pode-se afirmar que se toda a verificação anterior estava correta, ou seja, existe um registo de criação da cópia de segurança, o ficheiro ainda existe localmente e encontra-se íntegro, então não é necessário a geração de uma nova cópia de segurança. Por outro lado, caso o conteúdo dado seja do

tipo item, existe uma verificação extra a ser efetuada.

A última verificação a ser efetuado, no caso de o conteúdo ser do tipo item, é verificar se o item sofreu alguma modificação após a criação da última cópia de segurança. Com esse intuito, obtêm-se a data da última modificação ocorrida no item, compara-se com a data de criação da última cópia de segurança, e caso a data da última modificação seja posterior, é porque ocorreu uma modificação após ter sido gerada a última cópia de segurança, e como tal, é necessário a ocorrência da geração da cópia de segurança, caso contrário, não.

Através destas consecutivas verificações, é assegurado que só ocorre, em caso de necessidade, a criação de uma cópia de segurança de um dado conteúdo existente numa instância DSpace.

4.1.6 Sistema de *Backup*

O sistema de *backup* tem por objetivo a criação de cópias de segurança dos conteúdos existentes numa instância DSpace, caso as mesmas ainda não existam, se encontrem desatualizadas ou estejam corrompidas.

Através da análise das funcionalidades existentes na *AIP Backup & Restore* pode-se concluir que a API permite a exportação dos conteúdos de uma instância DSpace através de duas variantes: individual e hierárquica. A variante individual provou funcionar como especificada, fazendo a exportação completa e correta de um dado conteúdo existente na instância DSpace. A variante hierárquica, também provou funcionar como especificado, mas em termos de eficiência deixou muito a desejar, visto que esta variante ignora a existência de cópias de segurança já realizadas, refazendo-as novamente por vezes. Para rever os detalhes, consultar a secção 4.1.2.

Além disso, de modo a evitar-se a realização desnecessária de cópias de segurança, procedeu-se à elaboração de um algoritmo. Este algoritmo verifica se é necessário a realização da cópia de segurança de um dado conteúdo existente na instância DSpace, verificando se já existe alguma cópia de segurança associado ao conteúdo, e em caso de existência, certifica-se que a mesma se encontra atualizada e não se encontra danificada. Este algoritmo encontra-se detalhado ao pormenor na secção anterior, secção 4.1.5.

O sistema de *backup* foi implementado numa classe denominada de `Backup`. Esta classe permite a realização de cópias de segurança dos conteúdos existentes numa instância DSpace, certificando-se que as mesmas só ocorrem quando são inexistentes, estejam desatualizadas ou danificadas. Esta classe contém as seguintes funções principais:

- `exportCommunity()`: o objetivo desta função é a criação de uma cópia de segurança de uma dada comunidade, ou seja, a geração do respetivo ficheiro AIP. A geração do ficheiro AIP só ocorrerá caso se verifique que atualmente ainda não existe

nenhuma cópia de segurança sobre dessa comunidade, ou se a cópia de segurança existente encontra-se desatualizada ou corrompida;

- `exportCollection()`: esta função faz a geração de uma cópia de segurança de uma dada coleção, ou seja, a criação do respetivo ficheiro AIP. A criação do ficheiro AIP só acontecerá caso se verifique que atualmente ainda não existe nenhuma cópia de segurança da respetiva coleção, ou se a cópia de segurança existente se encontra desatualizada ou corrompida;

- `exportItem()`: função responsável pela criação da cópia de segurança de um determinado item, ou seja, a criação do respetivo ficheiro AIP. A criação do ficheiro AIP só acontecerá caso se verifique que atualmente ainda não existe nenhum ficheiro AIP do respetivo item, ou caso esse ficheiro esteja desatualizado ou corrompido;

- `exportCommunityAndChilds()`: função responsável pela criação da cópia de segurança de uma dada comunidade e das cópias de segurança de todos os conteúdos existentes na instância DSpace, hierarquicamente relacionados com a dada comunidade. Basicamente, esta função recebe uma certa comunidade e faz a chamada da função `exportCommunity()` para que ocorra a exportação da comunidade dada. De seguida, vê se a comunidade contém subcomunidades, e em resultado positivo, para cada uma das subcomunidades, faz a chamada recursiva desta função, para que seja criada a cópia de segurança da subcomunidade e dos seus respetivos filhos. Por último, verifica se a comunidade também contém coleções, e caso as contenha, faz a chamada da função `exportCollectionAndChilds()`, de modo a que a coleção e os seus respetivos filhos sejam exportados;

- `exportCollectionAndChilds()`: função que além de fazer a geração da cópia de segurança de uma dada coleção, também faz a geração das cópias de segurança dos conteúdos existentes na instância DSpace, que estão hierarquicamente relacionados. Em suma, esta função recebe uma coleção e faz a chamada da função `exportCollection()`, para que ocorra a geração da sua cópia de segurança. De seguida, vê se a coleção contém itens, e em caso positivo, para cada um dos itens, faz a chamada da função `exportItem()`, de modo a que ocorra a criação da cópia de segurança de cada um dos itens;

- `backupDone()`: função que verifica se para um dado conteúdo existente na instância DSpace já existe uma cópia de segurança criada, e em caso afirmativo, se a mesma está atualizada e se o ficheiro AIP se encontra correto. Esta verificação é feita através do algoritmo exemplificado na secção 4.1.5.

De realçar que exportação dos conteúdos do DSpace, ou seja, a criação das cópias de segurança, utiliza a função `disseminate()` proveniente da API *AIP Backup & Restore*.

A outra função da mesma API, a `disseminateAll()`, não foi utilizada visto que ignora a existência de cópias de segurança já realizadas.

Para uma melhor identificação do conteúdo de um dado ficheiro AIP, o nome dos ficheiros é gerado por omissão pelo seguinte padrão: `<tipo-conteúdo>@<handler-conteúdo>.zip`. Por exemplo, para um conteúdo do DSpace, com o `handler` 123456789-25, o nome do ficheiro AIP consoante o seu tipo, seria:

- Comunidade: COMMUNITY@123456789-25.zip;
- Coleção: COLLECTION@123456789-25.zip;
- Item: ITEM@123456789-25.zip.

De realçar que o padrão acima referido é utilizado pelo DSpace e também foi o adoptado por este sistema, visto que através dele é fácil a identificação do conteúdo presente na cópia de segurança.

Através deste sistema, assegura-se a correta geração das cópias de segurança dos conteúdos existentes na instância DSpace, e a sua geração só ocorre em caso de necessidade.

4.2 Componente de Envio para a *Cloud*

Um dos componentes do sistema de preservação é o componente de envio para a *cloud*, responsável pelo envio de cópias de segurança dos conteúdos de uma dada instância DSpace, ou seja, o envio para a *cloud* dos ficheiros AIP gerados pelo sistema de *backup* exemplificado na secção 4.1.6.

Este componente tem um valor acrescentado no sistema, visto que irá garantir a preservação das cópias de segurança em serviços *cloud*. Deste modo evita-se que as cópias de segurança geradas se percam ou fiquem corrompidas, assegurando assim, que as mesmas estão em bom estado e acessíveis no futuro em caso de necessidade.

Optou-se pela utilização do serviço de armazenamento na *cloud* disponibilizado pela Amazon S3, testando-se a ligação com o serviço através da biblioteca jclouds, biblioteca que uniformiza a ligação a diferentes serviços *cloud*, provenientes de vários fornecedores.

Com o intuito de um melhor controlo e gestão dos envios realizados para a *cloud*, é essencial que eles fiquem registados, como tal, foram criadas algumas funções de modo a permitir a inserção e atualização de registos dos envios para a *cloud*, dos ficheiros de *backup* existentes.

Implementou-se uma classe de nome `cloudConnection`, com o intuito de centralizar todos os pedidos que eram efetuados a um dado serviço *cloud*. Esses pedidos podem ser de variados tipos, como por exemplo, o estabelecimento da ligação ou o envio de um ficheiro AIP para a *cloud*.

De modo a evitar-se o envio desnecessário de cópias de segurança para a *cloud*, ou até mesmo o envio de ficheiros AIP desatualizados ou danificados, foram desenvolvidos dois algoritmos. Através destes algoritmos fica assegurado que um ficheiro AIP só é enviado para a *cloud* caso corresponda a uma cópia de segurança atualizada, não se encontre corrompido e não esteja de momento presente na *cloud*.

Por último, procedeu-se à implementação do sistema de envio de ficheiros AIP para a *cloud*, tendo em conta todas as análises efetuadas e a novas funcionalidades implementadas.

4.2.1 Testes de Envio de Ficheiros para a Cloud

Com o intuito de preservação das cópias de segurança dos conteúdos do DSpace, optou-se pela utilização de um serviço de armazenamento *cloud*. Deste modo, assegura-se que as cópias de segurança são preservadas corretamente, estando sempre acessíveis e íntegras, para o caso de necessidade futura de recuperação de conteúdos de uma dada instância DSpace, que se perderam ou ficaram danificados.

De modo a assegurar-se uma ligação uniforme com diferentes serviços de armazenamento *cloud*, optou-se pela utilização da biblioteca jclouds. Como já foi referido anteriormente na secção 3.1, esta biblioteca permite uniformizar a ligação com diversos serviços *cloud* provenientes de diferentes fornecedores, sem que no entanto se perca as potencialidades e flexibilidades características do serviço em causa.

A escolha do serviço de armazenamento *cloud* recaiu sobre o Amazon S3, visto tratar-se de um serviço com uma excelente reputação *online*, e além disso, os custos associados com os testes necessários aquando da implementação do sistema, não são muito elevados. Mais informação sobre o serviço Amazon S3 podem ser encontradas na secção 3.2.1.

Como já foi referido anteriormente, na secção 3.3.1, a BlobStore é uma das API's do jclouds, sendo responsável pela manipulação dos conteúdos armazenados na *cloud*, tendo como principais funções: envio de ficheiros para a *cloud*; receção de ficheiros da *cloud*; receção e manipulação da informação dos ficheiros existentes na *cloud*. Esta API oferece a possibilidade de ligação assíncrona ou síncrona, assim como o acesso aos ficheiros usando uma estrutura do tipo `map`.

De modo a assegurar uma melhor distribuição dos conteúdos na *cloud*, dentro do *container* definido, foram criados três pastas: communities, collections e items. O objetivo é que cada uma das pastas faça unicamente o armazenamento e preservação dos ficheiros AIP das comunidades, coleções e itens, respetivamente.

Inicialmente optou-se só pela utilização da ligação assíncrona da API BlobStore, visto que a mesma permitia envios paralelos, e deste modo assegurava-se velocidades de *upload* mais elevadas. Além disso, com a utilização da ligação assíncrona assegura-se o envio de ficheiros para *cloud* da Amazon com tamanho superior a 5GB, ação que seria impossível

de realizar através da ligação síncrona. Lamentavelmente foi detetado que o envio de ficheiros de tamanho inferior a 35MB, através da ligação assíncrona, não ocorria como o esperado, pois os ficheiros nunca chegavam à *cloud*. Como tal, todos os ficheiros de tamanho igual ou inferior a 35MB passaram a ser enviados através de uma ligação síncrona, e para os restantes seria utilizada uma ligação assíncrona.

Quando um ficheiro é enviado com sucesso para a *cloud*, o serviço de armazenamento *cloud* retorna o seu ETag, que é basicamente o *checksum* do ficheiro presente na cloud. Se o envio do ficheiro ocorreu por uma ligação síncrona, o ETag é igual ao valor MD5 do ficheiro enviado, contudo, se o ficheiro foi enviado através de uma ligação assíncrona, o ETag vai corresponder ao somatório dos *checksum* de cada uma das partes do ficheiro enviadas, e por consequência não será igual ao MD5 do ficheiro. De realçar, que aquando do envio de um ficheiro para a *cloud*, se o serviço *cloud* retornar o ETag, é porque o envio do mesmo ocorreu com sucesso, caso contrário, é retornada uma exceção contendo o erro.

Resumindo, pode-se afirmar que a API BlobStore da biblioteca jclouds permite uniformizar a ligação com vários serviços de armazenamento *cloud*, para o envio de ficheiros de variados tamanhos, sem que se perca as potencialidades e flexibilidades do sistema, concluindo-se assim, que a mesma cumpre as necessidades requeridas pelo sistema de envio de ficheiros para a *cloud*.

4.2.2 Registo dos Envios para a Cloud

O registo dos envios para a *cloud* das cópias de segurança geradas, os ditos ficheiros AIP, é essencial para uma gestão mais eficaz dos envios realizados, e dos próprios conteúdos enviados.

Como já foi referido anteriormente, na secção 4.1.4, já existe uma tabela denominada `logBackup`, que contém informação vital sobre a geração de cópias de segurança dos conteúdos existente numa instância DSpace, ou seja, os ficheiros AIP. Como se pretende enviar unicamente para a *cloud*, os ficheiros AIP gerados, não se viu a necessidade da criação de uma nova tabela, com o intuito de se assegurar um melhor controlo dos ficheiros AIP enviados. Como tal, foram adicionados alguns campos à tabela `logBackup`:

- `last_sendcloud`: data em que ocorreu o último envio de um dado ficheiro AIP para a *cloud*;
- `md5_sent`: MD5 do ficheiro AIP enviado para a *cloud*;
- `etag`: valor ETag retornado pelo serviço de armazenamento *cloud*.

Sendo assim, a tabela `logBackup`, além de conter a informação existente sobre as cópias de segurança geradas, também vai fazer manter o registo do envio das mesmas para a *cloud*.

Sempre que ocorra o envio para a *cloud*, de um dado ficheiro AIP, este acontecimento ficará registado na tabela `logBackup`. Caso já exista um registo associado a esse ficheiro AIP, o registo será atualizado, senão será inserido um novo registo, mantendo-se assim um registo único por cada ficheiro AIP.

Para que se possa registar, consultar e manipular de forma intuitiva, os registos das atividades de envio dos ficheiros AIP para a *cloud*, foram adicionadas algumas funções à classe `backupProcess`, a seguir listadas:

- `updateProcessSendCloud()`: responsável pela inserção ou atualização de um registo na tabela `logBackup`, registo esse relacionado com o envio para a *cloud* de um dado ficheiro AIP;
- `getLastSendCloudDate()`: retorna a data do último envio para a *cloud* de um dado ficheiro AIP, caso o envio tenha ocorrido;
- `getSentMD5()`: devolve o MD5 de um dado ficheiro AIP enviado para a *cloud*, caso o envio tenha acontecido;
- `getETag()`: devolve o último ETag retornado pelo serviço de armazenamento *cloud*, aquando da recepção de um dado ficheiro AIP;
- `equalFiles()`: verifica para um dado conteúdo DSpace, se o último ficheiro enviado para a *cloud* é igual ao ficheiro AIP gerado na última cópia de segurança criada.

As funções acima descritas têm duas assinaturas diferentes possíveis de utilizar. A primeira delas recebe como parâmetros o `type_object` e o `object_id`, campos existentes na tabela `logBackup`, com o intuito de identificação do registo a manipular, sendo esta assinatura destinada à manipulação de conteúdos ainda existentes na dada instância DSpace. A outra assinatura recebe como parâmetro o `handle`, campo também existente na tabela `logBackup`, com o intuito de identificar o registo a manipular, mas neste caso, esta assinatura é destinada à manipulação de registos que correspondam a conteúdos que existiam, mas já foram removidos da instância DSpace.

Deste modo, através da classe `backupProcess`, além de ser possível o controlo intuitivo e fácil de registos associados ao processo de criação das cópias de segurança, também é possível consultar e manipular intuitivamente os registos dos envios de ficheiros AIP para um serviço de armazenamento *cloud*.

4.2.3 Pedidos a um Serviço de Armazenamento *Cloud*

Com o intuito de centralizar todos os pedidos estabelecidos a um dado serviço de armazenamento *cloud*, procedeu-se à criação de uma classe, denominada `cloudConnection`.

Estes pedidos podem ser dos mais variados tipos, desde o estabelecimento da ligação, término da mesma, envio de ficheiros, ou até a recepção de informação sobre os ficheiros existentes na *cloud*.

Neste contexto, do envio de ficheiros AIP para a *cloud* e acesso à sua informação, foram criadas várias funções, a seguir listadas:

- `makeConnection()`: função responsável por estabelecer a ligação com um determinado serviço de armazenamento *cloud*;
- `closeConnection()`: função responsável pelo término da ligação existente com um dado serviço de armazenamento *cloud*;
- `getConnection()`: esta função retorna a ligação existente atualmente com um dado serviço de armazenamento *cloud*;
- `getInfoAllFilesInCloud()`: esta função retorna uma estrutura do tipo `map`, contendo na *key* os *handlers* dos ficheiros AIP preservados na *cloud*, e contendo no campo *value* o respetivo valor `ETag`;
- `getInfoFilesIn()`: esta função retorna uma estrutura do tipo `map`, contendo na *key* todos os `handlers` dos ficheiros AIP de um dado tipo, ficheiros esses preservados na *cloud*, e contendo no *value* o valor `ETag` dos respetivos ficheiros AIP;
- `sendFile()`: função responsável pelo envio para a *cloud* de um dado ficheiro AIP.

Resumidamente, esta API permite estabelecer a ligação com um serviço de armazenamento *cloud*, para que possa enviar as cópias de segurança geradas numa instância DSpace, e além disso ter acesso à informação dos ficheiros AIP preservados no dado serviço.

4.2.4 Algoritmos para o Envio de Ficheiros AIP para a *Cloud*

O envio desnecessário de ficheiros AIP para a *cloud* provoca um aumento do tráfego, como tal, sempre que se pretende enviar um dado ficheiro AIP, deve-se questionar a necessidade do seu envio.

Sendo assim, tornou-se prioritário o desenvolvimento de um algoritmo que controle quais ficheiros AIP devem ou não ser enviados para a *cloud*, tendo em conta que o envio de um dado ficheiro AIP, só deverá ocorrer sobre as seguintes condições:

- ficheiro AIP inexistente na *cloud*;
- ficheiro AIP existente na *cloud* encontra-se desatualizado;
- ficheiro AIP existente na *cloud* não corresponde ao último enviado.

Os conteúdos de uma instância DSpace estão em constante alteração, sendo possível a ocorrência da remoção de alguns deles, e nesse caso, poderão existir ficheiros AIP que não terão nenhum conteúdo associado na instância DSpace, apesar de terem sido gerados a partir dela. É importante a preservação desses ficheiros AIP, caso no futuro se pretenda fazer a restauro dos conteúdos removidos, como tal, é importante o envio desses ficheiros para a *cloud*, de modo a assegurar a sua correta preservação.

O algoritmo de verificação da presença de um dado ficheiro AIP na *cloud*, vai variar consoante a existência ou não do conteúdo por ele representado na instância DSpace, derivado da necessidade de verificação se um dado ficheiro AIP corresponde a uma cópia de segurança atualizada. Como tal, para o envio de ficheiros AIP que representem conteúdos ainda existentes na instância DSpace, será utilizado o algoritmo descrito na secção 4.2.4.1. Para o outro caso, de ficheiros AIP que representam conteúdos removidos da instância DSpace, será utilizado o algoritmo descrito na secção 4.2.4.2.

Através destes algoritmos certifica-se que todos os ficheiros AIP enviados para a *cloud* encontram-se atualizados e não corrompidos, a além disso, também se evita envios repetidos.

4.2.4.1 Algoritmo I

Este algoritmo destina-se à verificação da existência da cópia de segurança de um dado conteúdo ainda existente na instância DSpace. É verificado se uma cópia de segurança atualizada já foi enviada para a *cloud*, e se a mesma ainda existe lá e se continua íntegra. Na figura 4.2 pode-se visualizar a representação do algoritmo.

Inicialmente é verificado se o conteúdo é do tipo comunidade ou coleção, ou se por outro lado, é um conteúdo do tipo item.

Caso o resultado da verificação anterior confirme que o conteúdo é do tipo comunidade ou uma coleção, verifica-se se existe algum registo na tabela `logModifications` associado com o conteúdo. Caso exista, isto significa que ocorreu uma modificação desse conteúdo, e como tal, deverá ser efetuado uma cópia de segurança, antes que seja possível enviar o ficheiro AIP para a *cloud*. Por outro lado, caso não exista um registo associado na tabela `logModifications`, é verificado se existe algum registo na tabela `logBackup` associado também com o conteúdo. Caso não exista nenhum registo associado na tabela `logBackup`, isto significa que nenhuma cópia de segurança foi gerada do conteúdo, comunidade ou coleção, e como tal, a mesma deve ser gerada antes de se questionar o seu envio para a *cloud*. No entanto, caso exista o registo associado na tabela `logBackup`, significa que foi gerado uma cópia de segurança do respetivo conteúdo, mas nada assegura que a cópia de segurança tenha sido enviada para a *cloud*.

Caso o resultado da verificação inicial, detete que o conteúdo não é do tipo comunidade nem do tipo coleção, mas sim do tipo item, é verificado se existe algum registo na

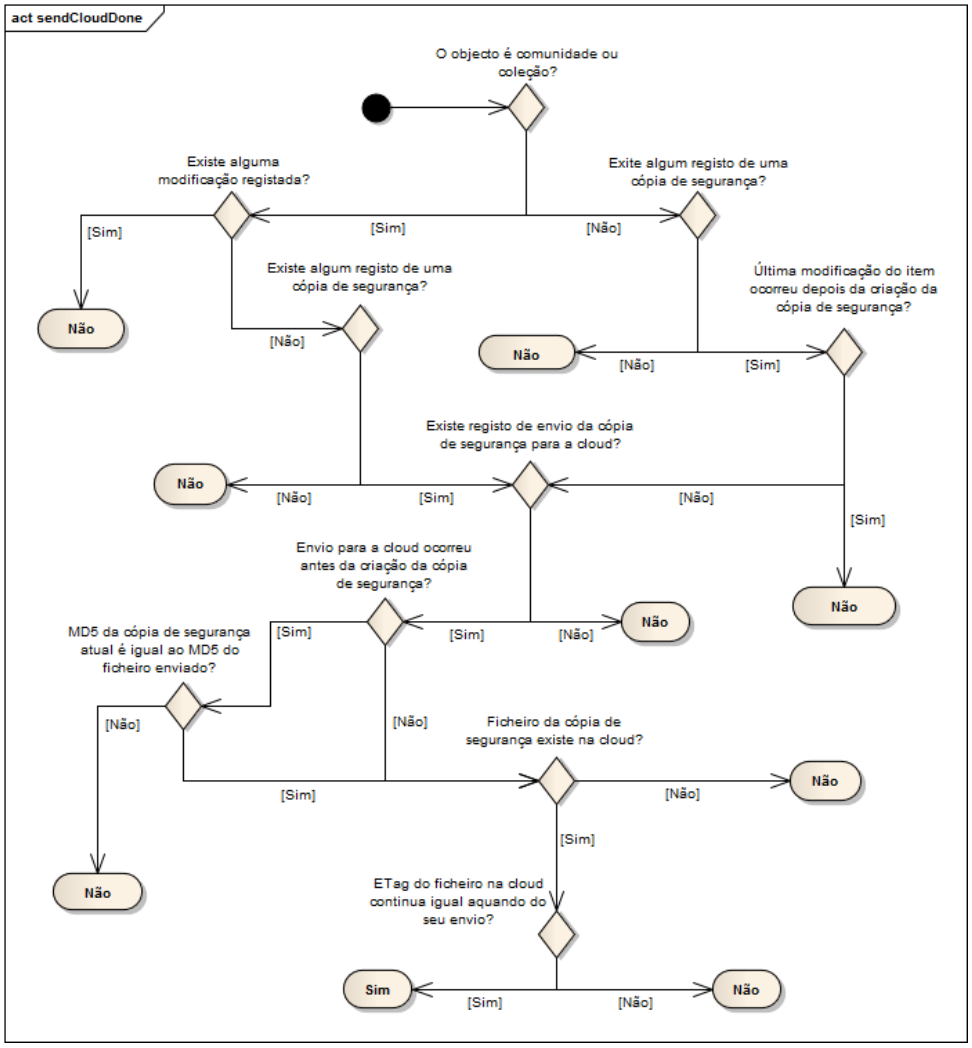

Figura 4.2: Algoritmo de confirmação se ficheiro AIP de um dado conteúdo DSpace encontra-se na *cloud*.

tabela `logBackup` associado com o dado item. Caso a tabela não tenha nenhum registo associado, significa que nenhuma cópia de segurança foi gerada daquele conteúdo, como tal deverá ocorrer primeiro a geração da respetiva cópia de segurança, antes de se equacionar o envio dela para a *cloud*. Por outro lado, caso exista um registo associado na tabela `logBackup`, tem que se verificar se a data da última modificação do item ocorreu antes ou depois da geração da última cópia de segurança do dado item. Se ocorreu depois, significa que uma mudança ocorreu no item, e que a cópia de segurança supostamente existente encontra-se desatualizada, como tal, deve-se efetuar um novo *backup* antes do envio da cópia de segurança para a *cloud*. Por outro lado, se a data da modificação é anterior à data de geração da última cópia de segurança, significa que a última cópia de segurança gerada se encontra atualizada, mas nada assegura que a mesma tenha sido enviada para a *cloud*.

Neste momento, caso ainda não se tenha detetado a impossibilidade de envio da cópia

de segurança para a *cloud*, independentemente do tipo do conteúdo, já se assegurou que a última cópia de segurança gerada encontra-se atualizada, mas nada assegura que ela já tenha sido enviada para a *cloud*.

Sendo assim, é verificado se na tabela `logBackup`, o registo correspondente ao conteúdo, tem algum valor no campo `last_sendcloud`. Caso não contenha significa que nenhum ficheiro AIP deste conteúdo foi alguma vez enviado para a *cloud*. Por outro lado, se existir uma data válida no campo, a mesma é comparada com a data da geração da última cópia de segurança do conteúdo.

No caso da data do último envio para a *cloud* ser posterior à data de geração da última cópia de segurança, deve-se certificar da existência do ficheiro na *cloud* e da sua consistência.

Por outro lado, no caso da data do último envio para a *cloud* ser anterior à data de geração da última cópia de segurança, isto significa que houve a criação de uma cópia de segurança após o último envio realizado para a *cloud*. De seguida, averigua-se se o ficheiro gerado no *backup* é igual ao último ficheiro AIP enviado para a *cloud*, comparando-se o MD5 da última cópia de segurança gerada com o MD5 do ficheiro enviado para a *cloud*. Caso o resultado da comparação dite que os MD5 não são iguais, então uma cópia de segurança pode e deverá ser enviada para a cloud com o intuito de preservação da mesma. Contudo, se a comparação confirmar que os dois MD5 são iguais, neste caso, confirma-se se a cópia de segurança ainda existe na cloud, e se o valor `ETag` da mesma ainda é igual ao valor retornado aquando do envio do ficheiro.

No caso de inexistência da cópia de segurança na cloud ou o valor `ETag` da mesma ter-se alterado desde o último envio, deverá ser reenviada a cópia de segurança do conteúdo DSpace de modo a se assegurar a sua preservação.

4.2.4.2 Algoritmo II

Elaborou-se este algoritmo de modo a verificar a possibilidade de envio para a *cloud* de ficheiros AIP, correspondentes a conteúdos já removidos da instância DSpace. Verifica-se a existência de algum registo da criação dessa cópia de segurança e se o ficheiro AIP gerado foi enviada para a *cloud*. Além disso, se o ficheiro AIP ainda existe na *cloud*, e se ele ainda contínua íntegro. Na figura 4.3 pode-se visualizar a representação do algoritmo.

Inicialmente verifica-se se existe algum registo na tabela `logBackup`, associado com o dado ficheiro AIP. Caso não exista nenhum registo associado, o ficheiro não deve ser enviado para a *cloud*, visto que isto significa que o mesmo não foi gerado pela instância DSpace.

Caso exista o registo associado na tabela `logBackup`, verifica-se se o dado registo contém alguma data válida no campo `last_sendcloud`. Caso não contenha, significa que o ficheiro AIP nunca foi enviado para a *cloud*, e neste caso, verifica-se para o dado

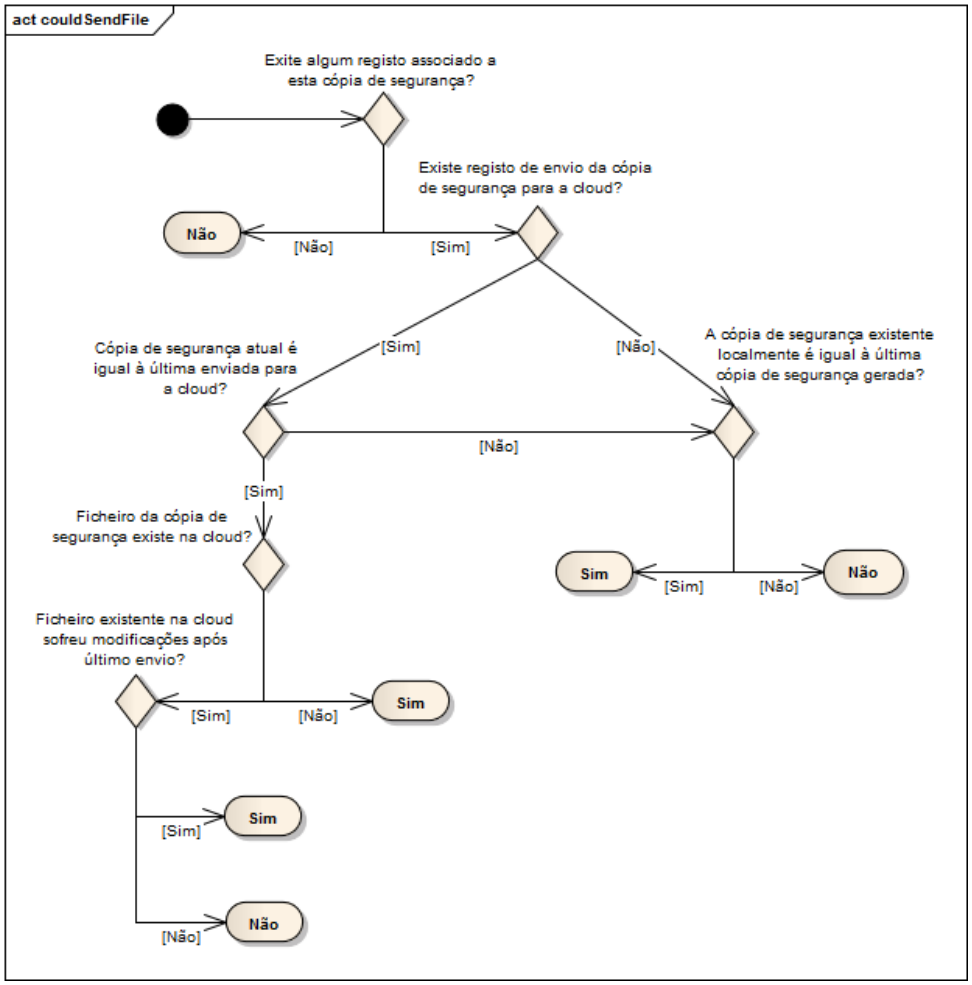

Figura 4.3: Algoritmo de verificação da possibilidade de envio de ficheiro AIP para a *cloud*

ficheiro AIP, se o seu MD5 continua igual ao MD5 preservado aquando da sua geração. Caso os MD5 sejam iguais, o ficheiro AIP pode ser enviado para a *cloud*, visto que continua íntegro, caso contrário, o ficheiro não deve ser enviado pois encontra-se corrompido ou corresponde a uma versão antiga.

No caso de existência de uma data válida no campo `last_sendcloud`, significa que o dado ficheiro AIP, este ou outra versão anterior, já foi enviado para a *cloud*. Averiguasse então para o dado ficheiro AIP, se o seu MD5 é igual ao MD5 do último ficheiro enviado para a *cloud*. Se os MD5 forem iguais, significa que o dado ficheiro já foi enviado para a *cloud*, e como tal verifica-se se o mesmo ainda se encontra na *cloud* e se não sofreu nenhuma modificação. Na confirmação da sua presença e da sua conservação intacta, não é necessário o envio do ficheiro, caso contrário o ficheiro AIP deverá ser reenviado para a *cloud*.

Se a comparação dos MD5, do ficheiro AIP dado e do último ficheiro enviado para a *cloud*, não for idêntico, significa que o dado ficheiro AIP não foi enviado para a *cloud*.

Nesta situação, verifica-se para o ficheiro AIP dado, se o seu MD5 continua igual ao MD5 preservado aquando da sua geração. Caso os MD5 sejam iguais, o ficheiro AIP pode ser enviado para a *cloud*, caso contrário, o ficheiro não pode ser enviado pois encontra-se corrompido ou corresponde a uma versão antiga.

Em suma, este algoritmo é direcionado ao envio de ficheiros AIP que correspondam a conteúdos que já foram removidos da instância DSpace. Através deste algoritmo certifica-se que não é reenviado nenhum ficheiro AIP já presente na *cloud*, e que no caso de envio, o ficheiro AIP corresponde a um ficheiro enviado anteriormente ou à última cópia de segurança gerada do conteúdo por ele representado, antes de o mesmo ter sido eliminado.

4.2.5 Sistema de Envio de AIP's para a *Cloud*

Este sistema tem por objetivo o envio de cópias de segurança para a *cloud*, os ditos ficheiros AIP, de modo a assegurar a preservação e acessibilidade das mesmas a longo prazo.

A implementação deste sistema foi precedida de várias fases. A primeira delas envolveu a análise da API BlobStore da biblioteca jclouds, no contexto de envio de ficheiros para um serviço de armazenamento *cloud*. Foi notado que a API permite dois tipos de ligação, síncrona ou assíncrona, tendo-se optado pela ligação síncrona para o envio de ficheiros de tamanho inferior a 35MB, e para os restantes ficheiros optou-se pela utilização de uma ligação assíncrona. Para mais detalhes, consultar a secção 4.2.1.

De modo a centralizar os pedidos efetuados com os serviços de armazenamento *cloud*, procedeu-se à criação da classe `cloudConnection`. Consultar a secção 4.2.3 para mais informações.

Como já foi referido anteriormente, uma cópia de segurança só deve ser enviada para a *cloud* caso se verifique uma das seguintes situações: inexistência da mesma na *cloud* ou a cópia de segurança presente na *cloud* encontra-se desatualizada ou corrompida. Mais informações sobre os algoritmos implementados estão presentes na secção 4.2.4.

Por fim, procedeu-se à criação do sistema de envio de cópias de segurança para a *cloud*, tendo o sistema sido implementado numa classe denominada `contentCloudManagement`, composta pelas seguintes funções principais:

- `sendCommunity()`: para uma dada comunidade existente na instância DSpace, é verificado se a cópia de segurança atualizada dessa comunidade já se encontra na *cloud*, e caso tal não se verifique, faz-se o envio do respectivo ficheiro AIP para a *cloud*;

- `sendCollection()`: para uma dada coleção existente na instância DSpace, verifica-se a existência da cópia de segurança atualizada na *cloud*, e caso tal não se verifique, é efetuado o envio do respectivo ficheiro AIP para a *cloud*;

- `sendItem()`: para uma dado item existente na instância DSpace, é verificada a existência na *cloud* da cópia de segurança atualizada desse item, e caso tal não se verifique, o respectivo ficheiro AIP é enviado para a *cloud*;

- `sendCommunityAndChilds()`: função responsável pelo envio para a *cloud* da cópia de segurança de uma dada comunidade existente na instância DSpace e também das cópias de segurança de todos os conteúdos do hierarquicamente relacionados. Resumidamente, esta função recebe uma certa comunidade e faz a chamada da função `sendCommunity()` para que ocorra o envio para a *cloud* do ficheiro AIP da comunidade dada. De seguida, verifica se a comunidade dada contêm subcomunidades, e caso tal se verifique, para cada uma das subcomunidades, faz a chamada recursiva desta função, para que seja enviado para a *cloud* a cópia de segurança das subcomunidades e dos seus respetivos filhos. Por último, é verificado se a comunidade também contêm coleções, e caso as contenha, faz a chamada da função `sendCollectionAndChilds()`, de modo a que cópias de segurança da coleção e dos seus respetivos filhos também sejam enviados para a *cloud*;

- `sendCollectionAndChilds()`: função que além de fazer a envio para a *cloud* da cópia de segurança de uma dada coleção existente na instância DSpace, também manda para a *cloud* as cópias de segurança dos conteúdos hierarquicamente relacionados. Em suma, esta função recebe uma coleção e faz a chamada da função `sendCollection()`, para que ocorra o envio da respetiva cópia de segurança para a *cloud*. De seguida, confirma se a coleção contém itens, e em caso afirmativo, para cada um dos itens, faz a chamada da função `sendItem()`, de modo a enviar para a *cloud* a cópia de segurança de cada um dos itens;

- `checkCommunitiesInCloud()`: esta função recebe uma lista de comunidades, existentes na instância DSpace, e devolve as comunidades que possuem o respetivo ficheiro AIP atualizado e preservado na *cloud*;

- `checkCollectionsInCloud()`: esta função após receber uma lista de coleções existentes na instância DSpace devolve as coleções que têm o respetivo ficheiro AIP atualizado e preservado na *cloud*;

- `checkItemsInCloud()`: recebe uma lista de itens, existentes na instância DSpace, e devolve os itens que já têm o respetivo ficheiro AIP atualizado e preservado na *cloud*;

- `sendAIPFile()`: esta função é responsável pelo envio de ficheiros AIP para a *cloud*, ficheiros esses que correspondem a cópias de segurança de conteúdos que já foram removidos da instância DSpace. Para que os ficheiros sejam enviados terão

de passar pelas verificações impostas por um algoritmo elaborado, já anteriormente explicado em detalhe na secção 4.2.4.2;

- `checkAIPFilesInCloud()`: esta função recebe uma lista de ficheiros AIP, verifica se eles encontram-se preservados na *cloud*, e devolve a lista dos que realmente estão preservados. Os ficheiros AIP recebidos pela função correspondem a cópias de segurança de conteúdos que foram removidos da instância DSpace.

Através desta API, é assegurado o envio dos ficheiros AIP atualizados para a *cloud*, independentemente de os mesmos corresponderem a conteúdos atualmente presentes ou não, na instância DSpace. De realçar que esta API também certifica-se que não são enviados ficheiros desnecessários, utilizando para isso, os dois algoritmos elaborados.

4.3 Componente de Receção da *Cloud*

Como já foi referido anteriormente, na secção 4.2.1, com o intuito de preservação a longo prazo das cópias de segurança dos conteúdos de uma instância DSpace, optou-se pela utilização de um serviço de armazenamento *cloud*. Deste modo, assegura-se a preservação e acessibilidade futura das cópias de segurança geradas, os ditos ficheiros AIP.

A acessibilidade futura das cópias de segurança é essencial de modo a assegurar-se a recuperação de conteúdos de uma instância DSpace, que por variados motivos, se perderam ou ficaram danificados. Neste contexto, surge a implementação deste componente.

Sendo assim, o componente de receção da *cloud*, é o componente do sistema de preservação responsável pela obtenção das cópias de segurança preservadas na *cloud*, ou seja, a receção dos ficheiros AIP que foram gerados pelo sistema de *backup* e enviados posteriormente para a *cloud*, de modo a garantir a sua correta preservação.

À semelhança do componente de envio para a cloud, descrito na secção 4.2, optou-se neste componente pela utilização da biblioteca jclouds, com o intuito de uniformizar a ligação com diferentes serviços de armazenamento *cloud*, para que deste modo o processo de obtenção dos ficheiros existentes na *cloud* seja uniforme.

Com o intuito de obter os ficheiros preservados na *cloud*, procedeu-se à implementação de uma nova função na classe `cloudConnection`, para que deste modo fosse possível estabelecer pedidos a um dado serviço de armazenamento *cloud*, para a receçao de ficheiros AIP preservados na *cloud*.

À semelhança dos outros componentes desenvolvidos, neste componente também foi necessário a implementação de um algoritmo, para que deste modo fosse evitado a obtenção desnecessária de ficheiros AIP preservados na *cloud*. Este algoritmo previne assim a obtenção de ficheiros AIP que já existam localmente, reduzindo assim o tráfego de *download*.

Por último, procedeu-se à implementação do sistema de receção de ficheiros AIP provenientes *cloud*, tendo em conta todas as análises efetuadas e a novas funcionalidades implementadas.

4.3.1 Testes na Receção de Ficheiros da *Cloud*

Como já foi referido anteriormente, na secção 4.2.1, aquando da criação da componente de envio para a *cloud*, a preservação das cópias de segurança dos conteúdos de uma instância DSpace, foi assegurada através da utilização de um serviço de armazenamento *cloud*. A principal razão para a preservação das cópias de segurança dos conteúdos de uma dada instância DSpace prende-se com a possibilidade de utilização futura das mesmas para a recuperação de conteúdos, que se possam ter perdido ou tenham ficado danificados.

Tal como no componente de envio para a *cloud*, descrito na secção 4.2, optou-se pela utilização da biblioteca jclouds, de modo a uniformizar a ligação com os serviços de armazenamento *cloud*, e assim obter-se as cópias de segurança aí preservadas, os ditos ficheiros AIP. Aquando do desenvolvimento do componente de envio para a *cloud*, os ficheiros AIP foram enviados para o serviço de armazenamento *cloud* disponibilizado pela Amazon S3, e como tal, é lá que se irá aceder de modo a recupera-los.

Recorrendo à API BlobStore optou-se pelo estabelecimento de uma ligação assíncrona, visto que a mesma permitia a receção paralela, e deste modo assegurava-se velocidades de *download* mais elevadas. Foi também notado, que independentemente do tamanho do ficheiro a ser recebido, a ligação assíncrona consegue garantir a receção do mesmo sem problemas.

Em conclusão, pode-se afirmar que a API BlobStore da biblioteca jclouds, além do envio também garante a receção de ficheiros independentemente do tamanho deles, sem que se perca as potencialidades e flexibilidades do serviço, concluindo-se assim, que a mesma cumpre novamente as necessidades impostas pelo sistema, neste caso específico, para a receção de ficheiros da *cloud*.

4.3.2 Novos Pedidos a um Serviço de Armazenamento *Cloud*

Na secção 4.2.3, foi referida a criação de uma classe denominada `cloudConnection`, que continha todos os pedidos estabelecidos a um dado serviço de armazenamento *cloud*. Esta classe permite, entre outras funções, o estabelecimento da ligação, término da mesma, envio de ficheiros, e receção de informação sobre os ficheiros existentes na *cloud*.

Para o correto funcionamento desta componente, falta na classe `cloudConnection`, uma função que permita a receção de ficheiros existentes num dado serviço de armazenamento *cloud*. Como tal, procedeu-se à adição de uma nova função na classe `cloudConnection`:

- `getFile()`: função responsável pela receção de um dado ficheiro AIP proveniente de um serviço de armazenamento *cloud*.

Deste modo, a classe `cloudConnection` vai conter todos os pedidos necessários sobre um dado serviço de armazenamento *cloud*: estabelecimento da ligação; término da mesma; envio de ficheiros; receção de ficheiros; e receção de informação sobre os ficheiros existentes na *cloud*.

4.3.3 Algoritmos para a Receção de Ficheiros da *Cloud*

A receção de ficheiros desnecessários da *cloud* provoca um aumento do tráfego, e tal situação pode ser facilmente evitada, questionando-se a necessidade de receção de uma dada cópia de segurança proveniente da *cloud*.

Com este intuito, desenvolveu-se um algoritmo para a receção de ficheiros provenientes da *cloud*, de modo a controlar-se os ficheiros que podem ser obtidos, evitando-se assim a receção de ficheiros desnecessários. Sendo assim, para que se possa obter uma dada cópia de segurança da *cloud*, deve-se verificar uma das seguintes condições:

- ficheiro AIP inexistente localmente;
- ficheiro AIP existente localmente é diferente do presente na *cloud*.

O algoritmo desenvolvido, de modo a responder às necessidades impostas, pode ser visualizado na figura 4.4.

Inicialmente verifica-se a existência na *cloud* do ficheiro AIP pretendido. Em caso de inexistência do mesmo, torna-se impossível a sua obtenção. Caso contrário, verifica-se se o ficheiro AIP existente na *cloud* não sofreu modificações desde que foi enviado, através da comparação do valor atual `ETag` com o valor `ETag` retornado aquando do seu envio.

Caso os dois valores `ETag` não sejam iguais, o ficheiro não pode obtido da *cloud*, visto que não corresponde ao ficheiro enviado anteriormente para lá.

Pelo contrário, caso se verifique que os dois valores `ETag` são iguais, confirma-se que o ficheiro AIP não sofreu modificações desde o seu envio para a *cloud*. Neste caso verifica-se a existência do ficheiro AIP localmente.

Caso se confirme a presença do ficheiro AIP localmente, e o mesmo seja semelhante ao ficheiro existente na *cloud*, não será possível a obtenção do ficheiro, visto que ele já existe localmente. A semelhança dos ficheiros é calculada através da comparação do

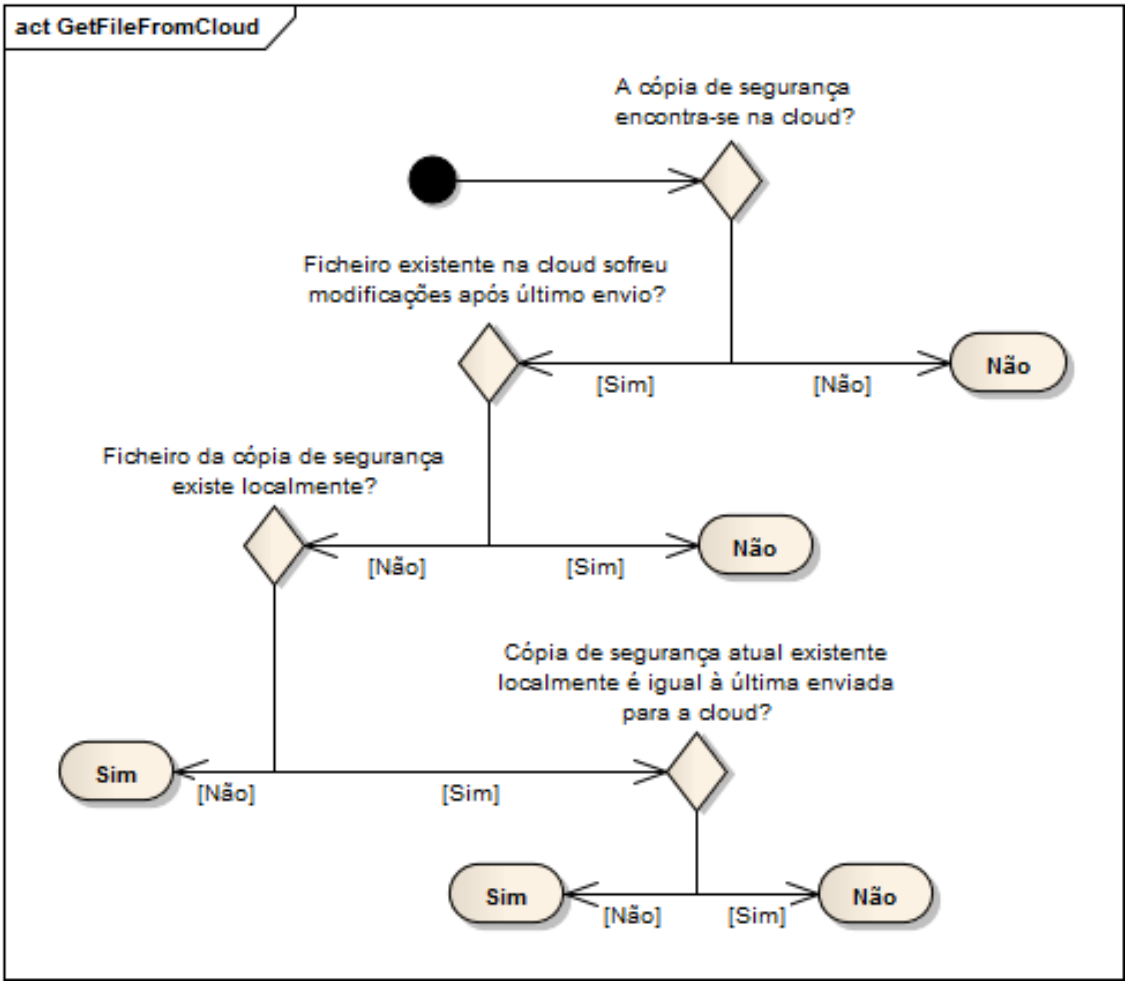

Figura 4.4: Algoritmo de confirmação da possibilidade de obtenção de ficheiro AIP da *cloud*.

MD5 do ficheiro existente localmente, com o MD5 do ficheiro enviado, gravado aquando do envio do mesmo para a *cloud*.

Por outro lado, caso não se confirme a presença local do ficheiro, ou no caso de o ficheiro existente localmente ser diferente do existente na *cloud*, será possível a obtenção do ficheiro AIP preservado na *cloud*. A existência de um ficheiro local diferente do existente na *cloud* pode-se dever a dois factos, o ficheiro local estar corrompido, ou o ficheiro local ser uma cópia de segurança mais recente, que foi gerada após a ocorrência de modificações no conteúdo existente na instância DSpace, a que corresponde o ficheiro AIP local.

Resumidamente, este algoritmo evita a receção de ficheiros AIP que já existam localmente, reduzindo-se assim o tráfego no *download* de ficheiros provenientes da *cloud*.

4.3.4 Sistema de Receção de AIP's da *Cloud*

Este sistema tem como principal objetivo a obtenção de cópias de segurança preservadas na *cloud*, os denominados ficheiros AIP. A obtenção das cópias de segurança é de extrema importância, para a recuperação de cópias de segurança armazenadas localmente, que por variados motivos, se possam ter perdido ou estejam danificadas.

De relembrar, que um ficheiro AIP só será obtido da *cloud* caso o mesmo não exista localmente, ou em caso de existência, a réplica existente seja diferente da existente no serviço de armazenamento *cloud*. A diferença entre as réplicas pode ter sido causada por dois motivos, uma modificação ocorreu no conteúdo existente na instância DSpace e uma nova cópia de segurança foi gerada, ou por outro lado, o ficheiro AIP existente localmente sofreu algum dano. Para a verificação da possibilidade de obtenção de um dado ficheiro AIP da *cloud* é utilizado um algoritmo analisado ao pormenor na secção 4.3.3.

A implementação deste algoritmo também foi precedida de outra fase, a avaliação da API BlobStore para a receção de ficheiros provenientes de um dado serviço de armazenamento *cloud*. Mais informações podem ser consultadas na secção 4.3.1.

Como já foi referido anteriormente, na secção 4.2.5, foi criada uma classe de nome `contentCloudManagement`, com o intuito de assegurar o envio dos ficheiros AIP atualizados para um dado serviço de armazenamento *cloud*. As novas funcionalidades, necessárias à receção de ficheiros AIP da *cloud*, foram implementadas nesta mesma classe, tornando-a assim detentora de todas as funcionalidades necessárias para o envio e receção de ficheiros AIP, entre a localização local e a *cloud*.

Como tal, de modo a suprimir as necessidades existentes aquando da receção de ficheiros AIP provenientes de um dado serviço de armazenamento *cloud*, foram adicionadas as seguintes funções principais à classe `contentCloudManagement`:

- `getCommunity()`: para uma dada comunidade existente na instância DSpace, é verificada a existência e consistência da cópia de segurança preservada na *cloud*, e caso tudo esteja correto, verifica-se a existência da mesma localmente. Caso não exista localmente, é obtida da *cloud*, e armazenada localmente;

- `getCollection()`: perante uma determinada coleção existente na instância DSpace, é verificado se a última cópia de segurança enviada ainda existe e encontra-se correta na *cloud*, e em resultado afirmativo, verifica-se a existência da mesma localmente. Caso se comprove que a cópia de segurança da dada coleção não existe localmente, obtêm-se o respetivo ficheiro AIP da *cloud*, e o mesmo é guardado localmente;

- `getItem()`: para um dado item existente na instância DSpace, é verificada a existência da última cópia de segurança enviada para a *cloud*, e caso a mesma ainda

exista e esteja corretamente preservada na *cloud*, é verificado a sua existência localmente. Caso não se confirme a sua existência local, é obtido e armazenado o respetivo ficheiro AIP preservado na *cloud*;

- `getCommunityAndChilds()`: função responsável pela receção da *cloud* da cópia de segurança de uma dada comunidade existente na instância DSpace e das cópias de segurança de todos os conteúdos hierarquicamente relacionados. Basicamente, esta função recebe uma certa comunidade e faz a chamada da função `getCommunity()` para que ocorra a obtenção da *cloud* do respetivo ficheiro AIP. De seguida, é verificado se a comunidade contém subcomunidades, e em resultado positivo, para cada uma das subcomunidades, faz a chamada recursiva desta função, de modo a que seja obtido da *cloud* a cópia de segurança das subcomunidades e dos seus respetivos filhos. Por último, verifica se a dada comunidade também contêm coleções, e caso as contenha, faz a chamada da função `getCollectionAndChilds()`, para que as cópias de segurança da coleção e dos seus respetivos filhos também sejam obtidas da *cloud*;

- `getCollectionAndChilds()`: função que além de obter da *cloud* a cópia de segurança de uma dada coleção existente na instância DSpace, também obtém da *cloud* as cópias de segurança dos conteúdos hierarquicamente relacionados. Em suma, esta função recebe uma coleção e faz a chamada da função `getCollection()`, para que se obtenha o respectivo ficheiro AIP da *cloud*. De seguida, vê se a coleção contém itens, e em caso positivo, para cada um dos itens, faz a chamada da função `getItem()`, de modo a seja obtido da *cloud* o respetivo ficheiro AIP de cada um dos itens;

- `checkPossibleCommunitiesGet()`: esta função recebe uma lista de comunidades existentes na instância DSpace, e devolve as comunidades sobre as quais é possível a obtenção do respetivo ficheiro AIP da *cloud*;

- `checkPossibleCollectionsGet()`: função que recebe uma lista de coleções, ainda existentes na instância DSpace, e devolve as coleções sobre as quais é possível a obtenção das respetivas cópias de segurança, preservadas na *cloud*;

- `checkPossibleItemsGet()`: após receber uma lista de itens existentes na instância DSpace, esta função devolve os itens sobre os quais é possível obter as respetivas cópias de segurança, os ditos ficheiros AIP, que se encontram preservados na *cloud*;

- `getAIPFile()`: função responsável pela obtenção de um dado ficheiro AIP da *cloud*. Esta função é principalmente destinada à receção de ficheiros AIP provenientes de conteúdos que já foram removidos da instância DSpace. Para que um dado

ficheiro possa ser obtido, o mesmo após ter sido enviado para a *cloud* não poderá ter sofrido nenhuma modificação, e não deverá haver nenhuma réplica local igual;

- `checkPossibleAIPFilesGet()`: esta função recebe uma lista de ficheiros AIP, e verifica se é possível a sua obtenção da *cloud*, tendo em conta as restrições imposta pelo algoritmo implementado e exemplificado na secção 4.3.3. São retornados os ficheiros AIP dos quais é possível a sua obtenção.

Deste modo é possível a obtenção dos ficheiros AIP preservados na *cloud*, independentemente deles corresponderem a conteúdos presentes ou não, na instância DSpace, tendo sempre a consideração de não obter ficheiros AIP já existentes localmente.

4.4 Componente de Importação

A geração de cópias de segurança dos conteúdos existentes numa instância DSpace, e sua consequente preservação, é essencial de modo a garantir futuramente a restituição dessa mesma informação na instância DSpace, em casos de ocorrência de perdas ou danos. Além disso, estas cópias de segurança podem ser utilizadas para importação de conteúdos para outras plataformas de conteúdos digitais existentes.

Até este momento, foi abordada a componente de geração das cópias de segurança dos conteúdos existentes numa instância DSpace, como se pode consultar pela secção 4.1. Na secção 4.2, pode-se observar os detalhes relativos à componente responsável pelo envio das cópias de segurança, os denominados ficheiros AIP, para um serviço de armazenamento *cloud*, com o intuito de assegurar a sua correta preservação. De seguida, na secção 4.3, abordou-se a componente de receção dos ficheiros preservados num dado serviço de armazenamento *cloud*, os denominados ficheiros API.

Por último, como última componente do sistema de preservação, surge a componente de importação, responsável pela utilização das cópias de segurança geradas, os ditos ficheiros AIP, para a importação dos conteúdos para uma dada instância DSpace. A importação de um dado conteúdo poderá ter que ocorrer pelos seguintes motivos:

- conteúdo presente na instância DSpace foi modificado erradamente;
- conteúdo foi removido da instância DSpace;
- inserção de um novo conteúdo na instância DSpace proveniente de outra instância existente.

Inicialmente procedeu-se à análise da API *AIP Backup & Restore*, no contexto de importação pra uma dada instância DSpace dos conteúdos presentes nas cópias de segurança geradas, os denominados ficheiros AIP.

De seguida, procedeu-se à implementação do sistema de importação. De realçar que o sistema de importação será repartido em três funcionalidades diferentes: substituição, restauro e inserção.

4.4.1 Importação através da *AIP Backup & Restore*

Como já foi referido na secção 3.1.4, esta API encontra-se incorporada no DSpace, desde o lançamento da versão 1.7, disponibilizando assim ferramentas que permitem a exportação e importação de conteúdos existentes numa instância DSpace: comunidades, coleções, itens ou *site*.

Neste contexto serão abordadas as ferramentas de importação de conteúdos para uma instância DSpace. A importação de um dado conteúdo pode ocorrer por diversos motivos: conteúdo presente na instância DSpace foi modificado erradamente ou encontra-se danificado; conteúdo foi removido da instância DSpace; ou inserção de um novo conteúdo proveniente de outra instância existente.

À semelhança das ferramentas de exportação, também foi notado na vertente de importação a existência de duas variantes: individual e hierárquica. A variante individual faz a importação de um único ficheiro AIP, para uma instância DSpace, enquanto a variante hierárquica, além de importar um dado ficheiro AIP, também faz a importação de todos os outros ficheiros AIP que estejam hierarquicamente abaixo, ou seja, ficheiros AIP que correspondam a conteúdos, que aquando da geração da cópia de segurança, eram considerados filhos do conteúdo representado no dado ficheiro AIP.

As funções existentes na API para a importação de ficheiros AIP para uma dada instância DSpace são:

- `replace()`: função que recebe um dado ficheiro AIP de um conteúdo existente na instância DSpace, e faz uma alteração completa do conteúdo existente na instância, através da sobreposição da informação proveniente no dado ficheiro AIP; Além disso, através desta função, também é possível o restauro de conteúdos que existiam previamente na instância DSpace.

- `replaceAll()`: função recebe um dado ficheiro AIP, sobre o qual é executado o que se encontra especificado na função `replace()`. Além disso, identifica os outros ficheiros AIP hierarquicamente relacionados, efetuando também sobre eles o que se encontra especificado na função `replace()`;

- `ingest()`: importação de um dado ficheiro AIP para uma instância DSpace, no qual a submissão ocorre como se tratasse de um novo conteúdo, procedendo-se à criação de um novo registo;

- `ingestAll()`: esta função recebe um dado ficheiro AIP e faz a sua submissão, criando um novo registo para o conteúdo submetido. Além disso, identifica os outros ficheiros AIP hierarquicamente relacionados, e efetua também as suas submissões.

Como de pode observar as funções `replace()` e `ingest()` fazem uma importação individual, enquanto as funções `replaceAll()` e `ingestAll()` fazem uma importação hierárquica.

Ambas as funções hierárquicas funcionam como especificado, mas é preciso ter uma especial atenção relativamente à função `replaceAll()`, pois o uso da mesma pode levar à importação desnecessária de ficheiros AIP, pois os conteúdos existentes na instância DSpace podem ainda não ter sofrido nenhum tipo de alteração desde a geração dos AIP que se encontram em uso.

4.4.2 Sistema de Importação

A importação de ficheiros AIP para uma instância DSpace é de extrema importância de modo a garantir-se a recuperação de conteúdos que se possam ter perdido ou se encontrem danificados. Além disso, é possível a ocorrência da importação de ficheiros AIP provenientes de outras instâncias DSpace, com o intuito de migração de conteúdos.

Resumindo, pode-se afirmar que uma importação pode ocorrer por três motivos diversos:

- conteúdo presente na instância DSpace foi modificado erradamente;
- conteúdo foi removido da instância DSpace;
- inserção de um novo conteúdo na instância DSpace proveniente de outra instância DSpace.

Como tal, aquando da importação de um dado ficheiro AIP, deve-se verificar se o conteúdo representado pelo mesmo faz parte da instância DSpace, se já existiu alguma vez na instância, ou se por outro lado nunca fez parte dela. Se o ficheiro AIP é identificado como sendo de um conteúdo existente na instância DSpace, deve-se analisar a possibilidade de importação por substituição. Contudo, se o ficheiro AIP for relacionado com um conteúdo existente em tempos no DSpace, deve-se fazer a importação do mesmo através de uma recuperação. Como último caso, temos ficheiros AIP, que não correspondem a nenhum conteúdo existente atualmente ou no passado na instância DSpace, e como tal, devem ser submetidos na instância DSpace, pois trata-se de conteúdos totalmente novos.

Como já foi descrito anteriormente, na secção 4.1.6, o nome dos ficheiros AIP é gerado segundo um padrão, para uma melhor identificação futura do conteúdo do qual provêm a cópia de segurança, sendo o padrão o seguinte: `<tipo-conteúdo>@<handler-conteúdo>.zip`. Assim é fácil identificar o tipo de

conteúdo do DSpace do qual provém a cópia de segurança, e o seu identificador único, o `handler`. Deste modo, é fácil de verificar se o conteúdo de um dado ficheiro AIP, ainda existe na instância DSpace, é um ex-conteúdo, ou por outro lado, é um novo conteúdo.

De modo a se cumprir as necessidades de cada um dos modos de importação, procedeu-se ao desenvolvimento individual de cada uma das funcionalidades: substituição; restauro; e inserção.

4.4.2.1 Funcionalidade de Substituição

Esta funcionalidade destina-se à importação de ficheiros AIP que estejam associados a conteúdos existentes na instância DSpace. Através da informação existente no nome do ficheiro AIP é fácil certificar-se que o dado ficheiro corresponde a um conteúdo existente na instância DSpace, bastando para isso, verificar que é possível a obtenção de um objeto DSpace através do `handler` existente.

O processo de substituição é bastante crítico, pois pode fazer com que se perca toda a informação existente, caso o ficheiro AIP transmitido no processo não seja o correto. Como tal, é importante que se certifique primeiramente, se a substituição é realmente necessária, e se o ficheiro que se pretende utilizar no processo corresponde a uma cópia de segurança gerada pelo sistema de *backup* para o conteúdo. Como tal, elaborou-se um algoritmo, que pode ser visualizado na figura 4.5.

Inicialmente, o algoritmo para um dado conteúdo existente na instância DSpace, verifica se a respetiva cópia de segurança existe localmente. Caso tal não se confirme, não é possível que ocorra a importação, contudo caso se confirme a existência da mesma, é verificado se a cópia de segurança existente localmente é igual à última enviada para a *cloud*. Esta verificação faz-se comparando o MD5 do ficheiro local com o MD5 da cópia de segurança enviada para a *cloud*, preservado aquando do seu envio.

Caso se confirme que os MD5 são iguais, significa que o ficheiro AIP existente localmente já foi enviado anteriormente para a *cloud*. Neste caso vai-se verificar se o ficheiro AIP local, corresponde à última cópia de segurança gerada do conteúdo, utilizando-se para isso a comparação do MD5 do ficheiro local com o MD5 do última cópia de segurança gerado, guardado aquando da geração da mesma.

Se confirmar-se que o ficheiro AIP local não corresponde à última cópia de segurança, pode-se proceder a uma substituição. Tal deve-se ao facto de que o ficheiro AIP local, corresponde a uma antiga cópia de segurança que foi enviada para a *cloud*, e após isso uma ou várias cópias de segurança foram geradas, e a um dado momento, alguém recuperou a cópia de segurança armazenada na *cloud* para que possa ocorrer uma recuperação, e assim colocar o conteúdo existente no DSpace como se encontrava aquando da geração do dado ficheiro AIP.

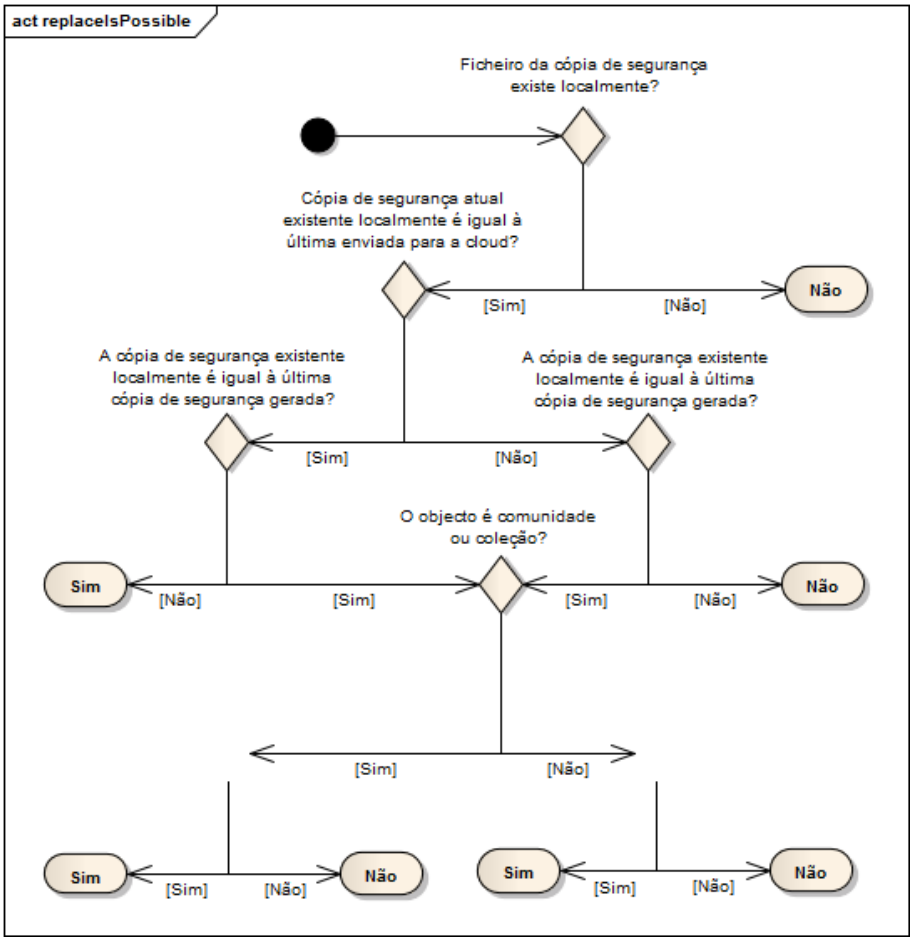

Figura 4.5: Algoritmo de verificação da possibilidade de ocorrência de substituição.

Por outro lado, caso se confirme que o ficheiro AIP local, corresponde à última cópia de segurança, é verificado se ocorreu alguma modificação no conteúdo existente na instância DSpace após a geração da última cópia de segurança, e caso tal se confirme, será possível proceder-se a uma substituição, visto que após a geração da última cópia de segurança, alguma mudança foi efetuada ao conteúdo, e como tal, agora é possível reaver o seu estado anterior. Caso tal não se confirme, não é possível fazer-se uma substituição, visto que o conteúdo do DSpace não sofreu modificações.

No caso da comparação anterior dos MD5, do ficheiro AIP local e da cópia de segurança enviada para a *cloud*, concluir-se que os valores são diferentes, procede-se à verificação se a cópia de segurança existente localmente corresponde à última cópia de segurança gerada. Caso se confirme que não se trata da mesma cópia de segurança, não será possível realizar o processo de substituição, visto que não há nenhum registo que comprove que o dado ficheiro AIP foi gerado legitimamente a partir do conteúdo existente naquela instância DSpace. Caso contrário, se comprove que o ficheiro AIP local corresponde à última cópia de segurança gerada, certificar-se-á se ocorreu alguma mo-

dificação no conteúdo existente na instância DSpace após a geração da última cópia de segurança, e caso tal se confirme, será possível proceder-se a uma substituição, visto que após a geração da última cópia de segurança, alguma mudança foi efetuada no conteúdo da instância, e como tal, agora é possível reaver o seu estado anterior. Caso tal não se confirme, não é possível fazer uma substituição, visto que o conteúdo existente na instância do DSpace não sofreu modificações.

Para que se verifique se um dado conteúdo sofreu modificações, primeiro tem que se saber o tipo do mesmo. Caso o conteúdo seja do tipo comunidade ou coleção, verifica-se a existência de algum registo na tabela `logModifications`, e caso tal se verifique é porque ocorreu uma modificação. Caso o conteúdo DSpace seja do tipo item, verifica-se se a data de geração da cópia de segurança ocorreu depois da última modificação do item, e em caso positivo, confirma-se que ocorreu uma modificação. Mais informações sobre o porque da utilização destes dois tipos de comparação, consultar a secção 4.1.3.

Através deste algoritmo certifica-se que só acorre uma substituição de um dado conteúdo existente na instância DSpace, em caso de necessidade, e no caso de ocorrência certifica-se que existe um registo que identifique o ficheiro AIP utilizado.

Esta funcionalidade, para efetuar a substituição de um dado conteúdo existente na instância DSpace, vai utilizar a função `replace()` da API *AIP Backup & Restore*.

Tendo-se desenvolvido o algoritmo, e escolhido a função de importação a utilizar, procedeu-se à criação do sistema de substituição dos conteúdos existentes numa instância DSpace, tendo o sistema sido implementado numa classe denominada `replace`, composta pelas seguintes funções principais:

- `replaceCommunity()`: para uma dada comunidade existente na instância DSpace, verifica se existe localmente a respetiva cópia de segurança válida, e se a mesma pode ser utilizada, para substituição do conteúdo existente no DSpace;

- `replaceCollection()`: esta função para uma dada coleção existente na instância DSpace verifica se é possível fazer a substituição da mesma, pela informação presente localmente numa cópia de segurança válida;

- `replaceItem()`: perante um item existente numa instância DSpace, é verificado se a respetiva cópia de segurança existe localmente, e se a mesma pode ser utilizada para substituição do conteúdo do item presente na instância DSpace;

- `replaceCommunityAndChilds()`: função responsável pela substituição de uma dada comunidade existente na instância DSpace e também de todos os conteúdos hierarquicamente relacionados, através dos respetivos ficheiros AIP. Basicamente, esta função recebe uma certa comunidade e faz a chamada da função `replaceCommunity()` para que ocorra a substituição da comunidade através do respetivo ficheiro AIP. De seguida, vê se a comunidade contêm subcomunidades, e

em resultado positivo, para cada uma das subcomunidades, faz a chamada recursiva desta função, para que ocorra a substituição das subcomunidades e dos seus respetivos filhos. Por último, verifica se a dada comunidade também contêm coleções, e caso as contenha, faz a chamada da função `replaceCollectionAndChilds()`, de modo a que ocorra a substituição da coleção e dos seus respetivos filhos;

- `replaceCollectionAndChilds()`: função que além de substituir uma dada coleção presente na instância, também substitui os conteúdos DSpace hierarquicamente relacionados, através dos seus respetivos ficheiros AIP. Em suma, esta função recebe uma coleção e faz a chamada da função `replaceCollection()`, para que se faça a substituição da dada coleção. De seguida, vê se a coleção contém itens, e em caso positivo, para cada um dos itens, faz a chamada da função `replaceItem()`, de modo que seja substituído cada um dos itens;

- `doReplace()`: verifica se para um dado conteúdo existente na instância DSpace é possível fazer a sua substituição. Esta validação ocorre através do uso do algoritmo da figura 4.5;

- `checkCommunititesReplace()`: esta função recebe uma lista de comunidades, existentes na instância DSpace, e devolve as comunidades sobre as quais é possível fazer uma substituição, através do respetivo ficheiro AIP;

- `checkCollectionsReplace()`: esta função recebe uma lista de coleções, existentes na instância DSpace, e devolve as coleções sobre as quais é possível fazer substituição, através do respetivo ficheiro AIP;

- `checkItemsReplace()`: recebe uma lista de itens existentes na instância DSpace, e devolve os itens sobre as quais é possível fazer substituição, utilizando o respetivo ficheiro AIP;

Através deste sistema, assegura-se a correta substituição dos conteúdos existentes numa dada instância DSpace, através dos respetivos ficheiros AIP.

4.4.2.2 Funcionalidade de Restauro

Esta funcionalidade destina-se à importação de ficheiros AIP que estejam associados a conteúdos que já existiram na instância DSpace.

Através da informação existente no nome do ficheiro AIP é fácil certificar-se que o dado ficheiro corresponde a um conteúdo não existente na instância DSpace, bastando para isso certificar-se que não é possível obter-se um objeto DSpace com o `handler` especificado. De seguida, utilizando os registos da tabela `logBackup`, é possível certificar-se da existência de um registo com o `handler` especificado, concluindo-se assim que ocorreu a geração de uma cópia de segurança de um conteúdo com aquele `handler`.

Tendo-se certificado que não existe atualmente um objeto na instância DSpace com o `handler` especificado, mas na tabela `logBackup` existe um registo com o `handler` especificado, pode-se concluir que no passado havia um conteúdo na instância DSpace identificado por aquele `handler`, sobre qual foi feito pelo menos uma cópia de segurança, mas após isso, por um motivo não especificado, o conteúdo acabou por ser removido da instância DSpace.

Tendo-se o respetivo ficheiro AIP do conteúdo removido, é possível fazer-se a sua restauração. Como tal, deve-se verificar se o ficheiro AIP dado corresponde a alguma cópia de segurança enviada para a *cloud* ou corresponde à última cópia de segurança gerada. Com esse intuito, foi desenvolvido um algoritmo que pode ser visualizado na figura 4.6.

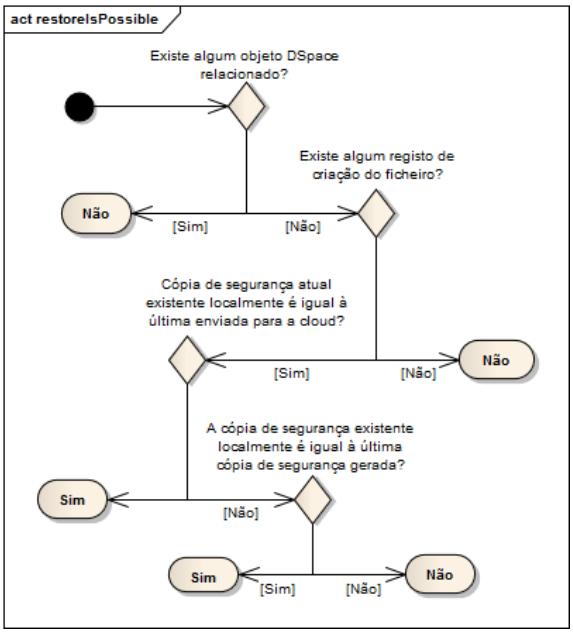

Figura 4.6: Algoritmo de verificação da possibilidade de ocorrência de restauro.

Este algoritmo verifica inicialmente se o conteúdo representado no ficheiro AIP não existe na instância DSpace, e se existe algum registo associado na tabela `logBackup`. Caso não se confirme alguma das condições anteriores, então não é possível fazer o restauro do conteúdo presente no dado ficheiro AIP.

De seguida verifica-se se o dado ficheiro AIP corresponde à última cópia de segurança enviada para a *cloud*, através da comparação do MD5 do ficheiro atual com o MD5 da cópia de segurança enviada para a *cloud*, preservado aquando do seu envio.

Caso os MD5 sejam iguais, isto significa que o ficheiro AIP existente localmente corresponde a uma cópia de segurança válida e como tal pode ser utilizada para restauração do conteúdo na instância DSpace.

Por outro lado, se os MD5 não forem iguais, vai-se certificar se o ficheiro AIP local corresponde à última cópia de segurança gerada, comparando-se o MD5 do ficheiro local com o MD5 da última cópia de segurança gerada, guardado aquando da geração da mesma. Caso esta verificação confirme que os MD5 são iguais, isto significa que o ficheiro AIP corresponde à última cópia de segurança gerada e como tal pode ser utilizada para se efetuar a restauração. Pelo contrário, caso não se confirme que os MD5 são iguais, a restauração não poderá ocorrer, visto que não se consegue assegurar que o ficheiro AIP foi gerado legitimamente a partir do conteúdo anteriormente existente na instância DSpace.

Através deste algoritmo certifica-se que um dado ficheiro AIP só é restaurado, caso não exista atualmente nenhum conteúdo associado ao mesmo na instância DSpace, apesar de haver um registo de uma anterior existência do mesmo, e certificando-se que o ficheiro AIP é legítimo.

Esta funcionalidade, de modo a efetuar a recuperação de um dado conteúdo para a instância DSpace, vai utilizar a função `replace()` da API *AIP Backup & Restore*.

Tendo sido finalizado o algoritmo e a escolha da função de importação, procedeu-se à criação do sistema de restauro de conteúdos na instância DSpace, tendo o sistema sido implementado numa classe denominada `restore`, composta pelas seguintes funções principais:

- `restoreAIP()`: esta função recebe um ficheiro AIP, verifica se é possível o restauro do mesmo, e caso seja possível faz o restauro do conteúdo existente na instância DSpace;

- `restoreAIPAll()`: esta função recebe um ficheiro AIP, verifica se é possível o seu restauro, e caso seja possível faz o restauro do conteúdo na instância DSpace. Além disso, também faz o restauro de todos os ficheiros AIP que correspondam a conteúdos hierarquicamente relacionados;

- `doRestore()`: função responsável por verificar se é possível a restauração a partir de um dado ficheiro AIP;

- `checkAIPFilesRestore()`: esta função recebe uma lista de ficheiros existentes localmente, e devolve os ficheiros com os quais é possível fazer o restauro dos conteúdos representados neles.

Através deste sistema, consegue-se a correta restauração de conteúdos no DSpace, através dos respetivos ficheiros AIP.

4.4.2.3 Funcionalidade de Inserção

Esta funcionalidade destina-se à importação de ficheiros AIP que nunca estiveram associados com a instância DSpace, ou seja a realização da importação de novos conteúdos para a instância DSpace.

Usando o nome do ficheiro AIP é fácil verificar-se que o dado ficheiro não corresponde a nenhum conteúdo existente na instância DSpace, bastando só certificar-se que não é possível obter um objeto DSpace com o `handler` especificado. Após isto, também é necessário verificar se existe algum registo na tabela `logBackup` com o `handler` especificado.

Caso se verifique para um dado ficheiro AIP, que não existe nenhum conteúdo associado na instância DSpace e nenhum registo associado na tabela `logBackup`, pode-se afirmar que o dado ficheiro AIP nunca existiu na instância DSpace e como tal o conteúdo do ficheiro API pode ser inserido. Uma representação deste algoritmo pode ser visualizada na figura 4.7.

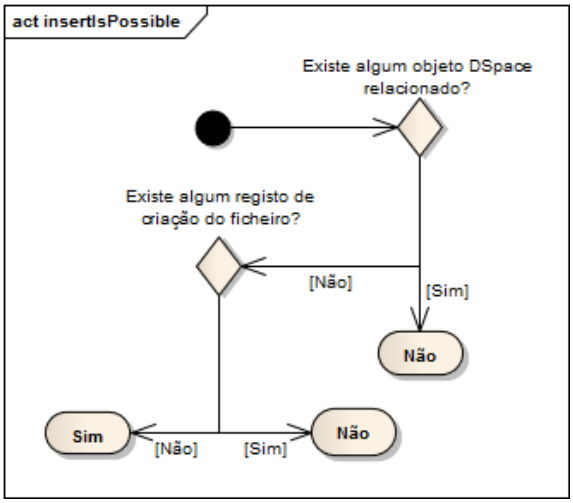

Figura 4.7: Algoritmo de verificação da possibilidade de ocorrência de inserção.

Para efetuar-se a inserção de um dado conteúdo na instância DSpace, vão-se utilizar as funções `ingest()` e `ingestAll()` da API *AIP Backup & Restore*.

Por fim, iniciou-se a criação do sistema de inserção de conteúdos numa instância DSpace, tendo-se criado a classe `insert` para a implementação do respetivo sistema, composto pelas seguintes funções:

- `insertAIP()`: esta função recebe um ficheiro AIP, verifica se é possível a inserção do conteúdo existente no ficheiro AIP, e caso seja possível faz a inserção dele na instância DSpace;

- `insertAIPAll()`: esta função recebe um ficheiro AIP, verifica se é possível a inserção do mesmo, e caso seja possível faz a sua inserção na instância DSpace. Além disso, também faz a inserção de todos os ficheiros AIP que correspondam a conteúdos hierarquicamente relacionados;
- `doInsert()`: função que verifica se é possível a inserção na instância DSpace, de um dado ficheiro AIP, correspondente à cópia de segurança de um conteúdo proveniente de outra instância DSpace;
- `checkAIPFilesInsert()`: esta função recebe uma lista de ficheiros existentes localmente, e devolve os ficheiros com os quais é possível fazer a inserção na instância DSpace.

Capítulo 5

Conclusões

Após a implementação final do sistema de preservação, mencionado no Capítulo 4, pode-se afirmar que é possível a preservação dos conteúdos presentes numa dada instância DSpace, com recurso à criação das respetivas cópias de segurança, resolvendo-se assim o problema detetado na secção 2.4. Para tal, foram implementadas dois componentes do sistema: o componente de *backup*, responsável pela criação das cópias de segurança dos conteúdos existentes numa dada instância DSpace; o componente de importação, responsável pela importação de conteúdos para uma dada instância DSpace, através das cópias de segurança geradas anteriormente. Estes componentes são descritos nas secções 4.1 e 4.4, respetivamente.

Relativamente à necessidade adicional que surgiu com a criação das cópias de segurança, a de garantir a integridade e acessibilidade das mesmas, optou-se pela utilização de um serviço de armazenamento *cloud*, que assegura ambos os requisitos. Para a inserção desta nova funcionalidade no sistema de preservação, foram implementados dois componentes, o componente de envio das cópias de segurança para um dado serviço de armazenamento *cloud*, e o componente para a obtenção das cópias de segurança já preservadas no próprio serviço. Para mais informações sobre os componentes, devem ser consultadas as secções 4.2 e 4.3, respetivamente.

Como referido no Capítulo 3, foram utilizadas duas tecnologias no desenvolvimento deste projeto de dissertação: o DSpace e o jclouds.

Relativamente ao DSpace, do que foi analisado, pode-se afirmar que o mesmo é uma boa solução para quem pretende a criação de um repositório para o armazenamento organizado de conteúdos digitais. O DSpace oferece uma excelente API para a exportação e importação dos conteúdos existentes numa das suas instâncias. Apesar disso, em caso de utilização da função de exportação hierárquica, deve ter-se em atenção que a mesma ignora a existência de cópias de segurança anteriormente geradas. Mais ainda, a utilização da função de importação das cópias de segurança (os chamados ficheiros AIP) para uma instância DSpace, deve ser utilizada cuidadosamente, pois uma importação errada

poderá levar à importação de informação errada ou até mesmo danificar o conteúdo original presente na instância DSpace. Relativamente à estrutura da base de dados do DSpace, o único ponto negativo a apontar é a inexistência da data da última modificação ocorrida nos conteúdos existentes na instância DSpace, caso eles sejam do tipo comunidade ou coleção.

Relativamente à biblioteca jclouds, a mesma mostrou-se bastante eficiente, permitindo a uniformização das ligações com diversos serviços *cloud* provenientes de diferentes fornecedores. O único problema encontrado prende-se com a impossibilidade de enviar ficheiros com menos de 35MB através de uma ligação assíncrona.

O sistema de preservação implementado permite a preservação, e consequente restauro, de todos os tipos de conteúdo do DSpace, exceto do tipo *site*. Um ficheiro AIP gerado a partir deste tipo contém os metadados gerais da instância DSpace, como por exemplo o seu nome. Também contém uma referência às comunidades de topo, ou seja, as comunidades que não são subcomunidade de nenhuma das comunidades existentes. Além disso, contém os metadados sobre os utilizadores e os grupos existentes na instância DSpace. Numa próxima versão deste sistema, o objeto *site* deve estar incorporado nas funcionalidades do sistema.

No futuro, a tabela `logBackup` criada, deixará de existir na base de dados do DSpace, sendo que os registos deverão ser efetuados e preservados num ficheiro XML. Uma cópia atualizada deste ficheiro deverá estar sempre preservado na *cloud*, assegurando-se deste modo que nunca se irão perder os registos associados com a gestão das cópias de segurança dos conteúdos de uma dada instância DSpace.

Por último, como trabalho futuro, a função `exportItem()` presente na classe `Backup` deverá ser melhorada em alguns aspetos. Aquando da geração de uma cópia de segurança de um conteúdo do tipo item, caso a respetiva cópia de segurança já exista, deve ser verificado a que níveis ocorreram as modificações no item. Assim sendo, caso as modificações tenham ocorrido ao nível dos metadados, só deverá ser gerado o ficheiro XML contendo os respetivos METS, e esse ficheiro deverá ser colocado na cópia de segurança existente. Caso as mudanças tenham ocorrido ao nível dos ficheiros existentes, os novos ficheiros deverão substituir as versões antigas presentes na cópia de segurança. Deste modo, assegura-se uma redução a nível de processamento e tempo necessário para a criação de uma cópia de segurança de um conteúdo do tipo item.

Referências

[Car10] Mark Carlson. Cloud Data Management Interface (CMDI) - The Cloud Storage Standard, 2010. Presentation from Storage Networking Industry Association.

[Cha12] Rachel Chalmers. Could jclouds be the abstraction layer that completes Cloudsoft?, January 2012. This report was originally published within 451 Research's Market Insight Service.

[Clo] CloudBees. jClouds Boosts Productivity and Focuses on Delivering Customer Value with DEV@cloud. http://www.cloudbees.com/case-study/jclouds.cb.

[dSRL] João Rocha da Silva, Cristina Ribeiro e João Correia Lopes. UPData Scientific Data Curation at U.Porto. http://sciencedata.up.pt/doc/.

[dSRL11] João Rocha da Silva, Cristina Ribeiro e João Correia Lopes. UPData - A Data Curation Experiment at U.Porto using DSpace. In *8th International Conference on Preservation of Digital Objects*. iPRES, 2011.

[Fer06] Miguel Ferreira. *Introdução à preservação digital : conceitos, estratégias e actuais consensos*. Universidade do Minho, Escola de Engenharia, 2006.

[Haz10] Vikas Hazrati. Getting Started With JClouds, 2010. Presented at IndicThreads.com Conference On Upcoming Tech.

[HBH09] Mark Hedges, Tobias Blanke e Adil Hasan. Rule-based curation and preservation of data: A data grid approach using iRODS. *Future Generation Comp. Syst.*, 25(4):446–452, 2009.

[HTT09] Tony Hey, Stewart Tansley e Kristin Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington, 2009.

[JCLa] INC JCLOUDS. jClouds. http://www.jclouds.org/.

[JCLb] INC JCLOUDS. jClouds - BlobStore API. http://www.jclouds.org/documentation/userguide/blobstore-guide/.

[Orga] DuraSpace Organization. DSpace - List of Members. http://www.dspace.org/whos-using-dspace.

[Orgb] DuraSpace Organization. DSpace - Official Web Site. http://www.dspace.org/.

[PN03] Electronic Resource Preservation e Access Network, editors. *Digital Preservation Policy Tool*. Information Society Technologies, 2003.

[Pru05] Marion Prudlo. E-Archiving: An Overview of Some Repository Management Software Tools. *Ariadne*, (43), April 2005.

[RSRF10] Eloy Rodrigues, Ricardo Saraiva, Cristina Ribeiro e Eugénia Matos Fernandes. Os Repositórios de Dados Científicos: Estado da Arte. Technical report, Julho 2010. Publicado no âmbito do projeto Repositório Científico de Acesso Aberto de Portugal (RCAAP).

[Ser] Amazon Web Services. Amazon Simple Storage Service (Amazon S3). http://aws.amazon.com/s3/.

[Sin07] Neha Singh. Solution for E-Journal Archiving And Framework For Evaluation Of Archiving Software. February 2007. 5th International CALIBER -2007, Panjab University, Chandigarh, 08-10 February, 2007.

[SMM09] Flávio R. C. Sousa, Leonardo O. Moreira e Javam C. Machado. Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios. *Universidade Federal do Ceará (UFC)*, 2009. http://www.es.ufc.br/~flavio/files/Computacao_Nuvem.pdf.

[SS10] Katherine Skinner e Matt Schultz. *A Guide to Distributed Digital Preservation*. Educopia Institute, Atlanta, 2010.

[Tea11] The DSpace Developer Team. *DSpace 1.8 Documentation*. DSpace, November 2011.

[TR10] Heiko Tjalsma e Jeroen Rombouts. Selection of Research Data. Guidelines for appraising and selecting research data. A report by DANS and 3TU.Datacentrum, July 2010.

[TW11] Aaron Trehub e Andrew Waller. Private LOCKSS networks: overview and working examples. Presented at the LYRASIS Staying On TRAC Post-Workshop Webinar, March 2011.

Anexo A

Anexos do DSpace

A.1 Estrutura da Base de Dados

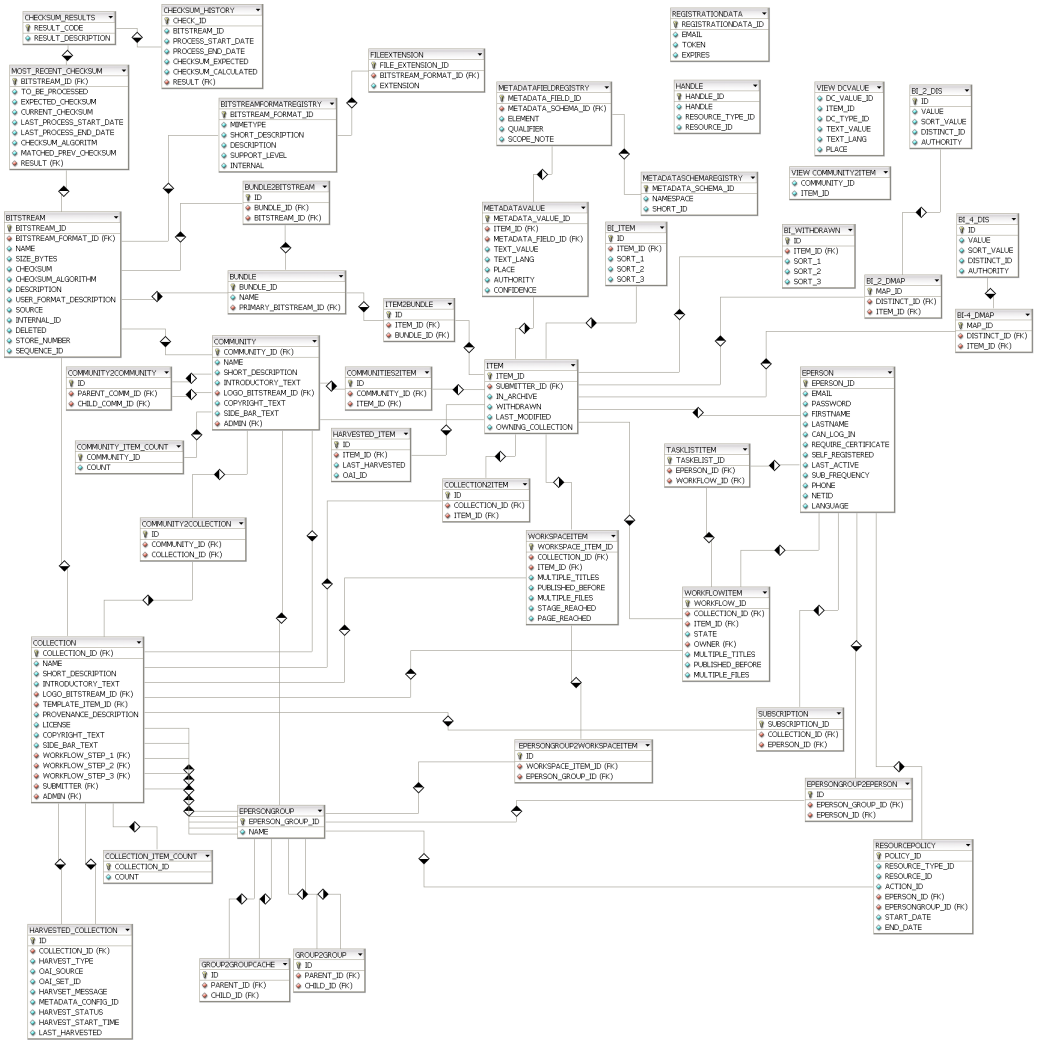

Figura A.1: Estrutura organizacional da base de dados do DSpace.
Imagem retirada de: https://wiki.duraspace.org/display/DSDOC18/Storage+Layer