Universidade de Aveiro
2012
Departamento de Eletrónica, Telecomunicações e Informática

Tito Augusto de Carvalho Azevedo

Repositório de dados pessoais das Redes Sociais

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Joaquim Arnaldo Carvalho Martins, Professor Catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais, restante família e amigos.

o júri

Presidente

Prof. Doutor Joaquim Manuel Henriques de Sousa Pinto
Professor Auxiliar da Universidade de Aveiro

Vogais

Prof. Doutor Fernando Joaquim Lopes Moreira
Professor Associado do Departamento de Inovação, Ciência e Tecnologia da Universidade Portucalense

Prof. Doutor Joaquim Arnaldo Carvalho Martins
Professor Catedrático da Universidade de Aveiro (orientador)

agradecimentos

Aos meus pais por todo o apoio e ajuda ao longo de todos estes anos.

À minha família por me ter apoiado sempre que precisei.

Aos meus amigos por terem transformado os momentos difíceis em momentos menos difíceis.

Ao meu orientador, Doutor Joaquim Arnaldo Carvalho Martins e ao Marco Pereira, o meu agradecimento pela disponibilidade e ajuda durante este trabalho.

palavras-chave Repositório de dados, redes sociais, Resource Description Framework (RDF)

Resumo

Uma rede social, resumidamente, é uma estrutura que agrega pessoas e organizações permitindo interações e partilha de informação entre elas.

As redes sociais são, cada vez mais, parte integrante do quotidiano de grande parte da população mundial.

Através destas podemos não só contactar com amigos, colegas ou familiares, usando redes sociais como o Facebook, o Twitter ou o Google+, mas também criar e/ou fortalecer ligações profissionais, usando, por exemplo, o LinkedIn.

Neste momento, assiste-se também a uma presença cada vez maior de todo o tipo de empresas nas redes sociais, seja para divulgar os seus produtos, para divulgar notícias, como por exemplo os meios de Comunicação Social, ou mesmo para simbolizar um simples ato de publicidade gratuita.

Além de todas estas possibilidades, as redes sociais atuais têm ainda uma forte componente ligada às imagens e aos vídeos.

Estes serviços são exaustivamente requisitados pelos utilizadores das redes sociais, havendo inclusive aplicações feitas especificamente para suportar esta vertente, como é o caso do Instagram.

Na verdade, a facilidade que temos em utilizar os serviços disponibilizados pelas redes sociais leva-nos a partilhar os conteúdos supracitados sem pensar na eventualidade de os mesmos poderem ser perdidos, o que já aconteceu em casos pontuais.

O objetivo deste trabalho é precisamente possibilitar uma garantia de que é guardado um histórico destes conteúdos, possibilitando não só a consulta da informação guardada, como também a sua eventual recuperação.

Este facto permite ainda a utilização dos dados de uma rede social noutra, havendo, desse modo, a possibilidade da migração entre redes sociais, nos casos em que tal faça sentido, não sendo o utilizador condicionado por já ter conteúdo numa rede social diferente.

Além da possibilidade dada ao utilizador de guardar os dados, nomeadamente as imagens e vídeos, no disco do seu computador, são ainda disponibilizados serviços que permitem o alojamento *online* desses ficheiros, garantindo assim um risco menor relativamente à possibilidade de perda do conteúdo.

keywords Data repository, social networks, Resource Description Framework (RDF)

abstract

A social network, in short, is a structure that aggregates people and organizations allowing interactions and information sharing between them.

Social networks are increasingly an important part of the daily life in most of the world population.

Through them we can not only contact our friends, colleagues or family, using social networks such as Facebook, Twitter or Google+, but also create and/or strengthen professional connections using, for instance, LinkedIn.

Currently we also witness an increasing presence of all types of companies in social networks, whether to announce their products, to unveil news, like the media does, or simply for a free act of publicity.

Beyond all these possibilities, actual social networks also have a strong part related to images and videos.

These services are exhaustively required by social network users, what caused the appearance of applications made specifically to support this aspect, like Instagram.

In truth, the ease we have while using the services made available by the social networks makes us share the aforementioned contents without thinking of the possibility of losing them, something that has happened in certain occasions.

The objective of this work is precisely the possibility of giving an assurance that a history of these contents will be kept, allowing not only the lookup of the information kept, but also its eventual recovery.

This fact allows, as well, the usage of a certain social network data in other social networks, whenever that possibility makes sense, allowing migration between social networks, giving the user a certain independence regarding the content he has on the original social network.

Beyond the possibility given to the user of storing the data, in particular images and videos, on the computer's hard drive, online file hosting services are also available so the user can have less risk when it comes to the possibility of losing the content.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AIISO	Academic Institution Internal Structure Ontology
AGPL	GNU Affero General Public License
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASP	Active Server Pages
BBS	Bulletin Board System
BD	Base de Dados
BSD	Berkeley Software Distribution
BSON	Binary JSON
CAP	Consistency, Availability, Partition Tolerance
COCOMO	Constructive Cost Model
CSS	Cascading Style Sheets
DCMI	Dublin Core Metadata Initiative
DETI	Departamento de Eletrónica, Telecomunicações e Informática
DOM	Document Object Model
FOAF	Friend Of A Friend
Gzip	Gnu Zip
HDFS	Hadoop Distributed File System
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IIS	Internet Information Services
IRC	Internet Relay Chat
JAR	Java Archive
JAX-WS	Java API for XML Web Services
JDBC	Java Database Connectivity
JEE	Java Enterprise Edition
JSON	JavaScript Object Notation
JSP	JavaServer Pages
KIPI	KDE Image Plugin Interface
LAMP	Linux/Apache/MySQL/PHP
MA	Media Annotations
MDX	MultiDimensional Expressions

OG	Open Graph
PAC	Photo Access Control
PHP	Hypertext Preprocessor
PIN	Personal Identification Number
RDF	Resource Description Framework
REST	Representational State Transfer
RPC	Remote Procedure Call
RSS	RDF Site Summary
S3	Amazon Simple Storage Service
SCF	Class Scheme for Science Fields
SMS	Short Message Service
SIOC	Semantically-Interlinked Online Communities
SO	Sistema Operativo
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
UA	Universidade de Aveiro
UDDI	Universal Description Discovery and Integration
URI	Uniform Resource Identifier
URIS	URI References
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WISA	Windows/IIS/SQL Server/ASP.NET
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible Markup Language

1. Introdução

Com este capítulo pretende-se introduzir os objetivos alcançados com esta dissertação, bem como mostrar as motivações que levaram à escolha deste tema.

1.1. Contexto

Uma rede social pode ser definida como uma plataforma que permite a criação de relações sociais entre utilizadores e/ou organizações.

Um utilizador é, normalmente, identificado por um nome, uma fotografia que pode ou não estar presente, um conjunto de informações pessoais que pode apenas ser disponibilizado a um número restrito de utilizadores e pela multiplicidade de serviços que usa, como a troca de informação entre contas, a partilha de conteúdo multimédia, a inscrição em eventos, entre outros.

As redes sociais atuais promovem não só uma interação assíncrona, como vem sendo habitual, com a criação de tópicos de conversação, com a possibilidade do envio de comentários a outros tópicos, a imagens, a vídeos, etc. mas também, cada vez mais, uma interação síncrona com a possibilidade da troca de mensagens pessoais em tempo real.

Inerente a esta definição está um conjunto de problemáticas que surgem, com a crescente utilização das redes sociais nos dias que correm. Questões como a privacidade, dado que muitas vezes os conteúdos partilhados podem ser acedidos por pessoas que não têm uma relação direta com a pessoa que as partilha. Outra questão relevante é a possibilidade do aparecimento de utilizadores mal-intencionados que consigam explorar algumas vulnerabilidades deixadas por utilizadores mais jovens.

Relevante é, também, o facto de muitas das redes sociais utilizadas manterem uma Base de Dados com informação confidencial do utilizador, tal como o acesso às mensagens privadas ou aos dados pessoais não partilhados com terceiros e a possibilidade de acesso aos interesses de um utilizador dada a certas empresas para que possam, desse modo, criar ações publicitárias mais apelativas e eficazes.

As redes sociais são, inegavelmente, uma das ferramentas mais poderosas da Internet atual, mas os seus efeitos psicológicos em crianças e jovens adultos podem trazer graves problemas e devem ser alvo de análises cuidadas [1].

Em termos históricos, antes da proposta para a criação da *World Wide Web* (WWW) por Tim Berners-Lee em 1989 [2], surgiram os primeiros sistemas que permitiam a

criação de pequenas comunidades onde se trocavam ideias e opiniões. Estas comunidades, por si só, acabavam por ser uma pequena demonstração de uma rede social embora, obviamente, sem a dimensão que estas atingem atualmente. Muitas destas páginas permitiam a participação em *chats* onde era possível interagir em tempo real com outros utilizadores.

Em 1978 foi criado o BBS (Bulletin Board System) que foi inventado em Chicago por dois informáticos para informar os amigos de reuniões, marcar anúncios e partilhar informações. Algumas destas ações acabam por resumir as possibilidades dadas pelas redes sociais nos dias que correm, pelo que esta acabou por ser, por si só, um bom exemplo de uma rede social [3].

Em 1980 surgiu a Usenet que consistia num serviço de troca de artigos, através da utilização de *newsgroups*, que consistiam em fóruns de discussão agrupados por assuntos comuns [4].

Em 1985 foi criado o The WELL, uma comunidade *online* que permitia conversação e discussão de variadas temáticas. Foi definido como a primeira "comunidade virtual". [5]

Em 1993 surgiu o IRC (Internet Relay Chat) que não era mais do que uma rede social em tempo real. As pessoas podiam trocar mensagens diretamente entre elas ou em canais criados para agrupar pessoas com interesses comuns. Era ainda possível enviar e receber ficheiros.

Alguns exemplos de páginas que tinham pequenas comunidades que podiam trocar ideias e debater assuntos são o Geocities, criado em 1994 e que permitia a criação de páginas pelos seus utilizadores em diferentes temas e comunidades, como o desporto, o entretenimento, etc. [6] e o Tripod criado em 1995, que também permitia a criação de páginas pessoais ou representativas de organizações, mas com especial ênfase em estudantes e jovens [7].

Em 1999 foi lançado o MSN Messenger que permitia a interação entre utilizadores possuidores de uma conta de *e-mail* pertencente à Microsoft. Era permitida a troca de mensagens diretamente entre utilizadores, bem como a adição de um ou mais amigos à conversação. Era ainda possível conversar por voz e/ou vídeo.

No final do Século XX começou a surgir a definição de amigo entre os utilizadores da WWW. Este amigo representava nada mais do que um utilizador que era conhecido por quem acedia ao serviço e que, ao ser adicionado, permitia um mais fácil acesso e interação.

Desde então foram surgindo cada vez mais plataformas que permitiam a interação entre amigos, como o Friendster, lançado em 2002, que não era mais do que uma página

de jogos que permitia o contacto entre utilizadores, mas o verdadeiro crescimento das redes sociais surgiu em 2003 com o aparecimento do MySpace que rapidamente se tornou um local de visita diária. Em 2005 foi revelado que o MySpace tinha mais visualizações do que o Google [8]. Nesta altura surgiu também o LinkedIn que se tornou na principal rede social para profissionais, permitindo a criação de ligações entre colegas, utilizadores com interesses profissionais semelhantes, ou mesmo o recrutamento de trabalhadores.

Em 2004 surgiu a grande explosão nas redes sociais, com o aparecimento do Facebook. No Facebook inicial eram reunidos todos os tipos de interação disponíveis nas redes sociais predecessoras, o que trouxe um acrescido interesse. Em 2010 foi anunciado que o Facebook era a página mais consultada de toda a WWW pelo segundo ano consecutivo [9].

Em 2006 surgiu o Twitter, uma rede social que também se tornou extremamente popular e que se caracteriza por possibilitar o envio de mensagens com apenas 140 caracteres, ao estilo das mensagens SMS, extremamente populares sobretudo no segmento mais jovem da população.

Em 2011 surgiu o Google+, a tentativa do Google de rivalizar com o Facebook.

Estima-se que o Google+ tenha cerca de 170 milhões de utilizadores registados, sendo cerca de 100 milhões ativos, contra os cerca de 900 milhões de utilizadores registados, sendo 530 milhões ativos do Facebook [10].

1.2. Objetivos

As redes sociais atuais, de uma maneira ou de outra, permitem a interação através da partilha de conteúdo multimédia. A facilidade com que se partilham imagens, vídeos, informações pessoais e outros tipos de conteúdos levam-nos, por vezes, a confiar em demasia nos serviços que usamos.

Pretende-se, com este trabalho, garantir que é mantido um repositório digital pessoal de cada utilizador, que é composto por todos os conteúdos partilhados pelo utilizador, incluindo imagens e vídeos, possibilitando, desse modo, a salvaguarda de um histórico das ações desenvolvidas pelo utilizador ao longo do tempo. Os conteúdos multimédia serão guardados, dependendo da opção selecionada pelo utilizador, no disco rígido do computador pessoal do utilizador ou em serviços de alojamento *online*, podendo os mesmos estar em mais do que um local, garantindo assim uma menor probabilidade da perda desses conteúdos. O conteúdo textual, nomeadamente as publicações dos

utilizadores, as mensagens trocadas, as listas de amigos e os restantes conteúdos partilhados serão guardados em formato Resource Description Framework (RDF) e enviados para uma Base de Dados NoSQL que tratará de salvaguardar essa informação associada a um *login* garantindo segurança no acesso a esses dados.

Pretende-se também possibilitar ao utilizador a interação inversa, isto é, poder recuperar os dados multimédia e textuais e colocá-los na rede social, dando a opção de migrar entre diferentes contas, onde tal fizer sentido já que, por exemplo, não fará sentido querer migrar uma conta de Facebook para uma conta de Twitter tendo em conta a limitação do número de caracteres por mensagem imposta pelo Twitter. Deste modo o utilizador não precisa de estar condicionado pelo conteúdo presente numa rede social quando quer mudar para outra já que pode facilmente migrá-lo sem grandes limitações. Este objetivo também é relevante no caso de uma eventual perda ou roubo de conta, já que permitirá a recuperação de toda a informação para uma nova conta, não ficando o utilizador sem os dados originais.

1.3. Motivação

Como foi referido anteriormente, a grande motivação prende-se com a necessidade encontrada de manter um repositório digital com todos os conteúdos partilhados pelo utilizador ao longo do tempo.

Existem, no entanto, outras motivações que surgem com o desenvolvimento desse repositório. Com o crescimento da utilização das redes sociais, cresce também a necessidade de manter alguma segurança nos dados utilizados nelas e de garantir, não só que os conteúdos partilhados, como referido anteriormente, tenham cópias, mas também que é mantido um histórico das interações desenvolvidas pelo utilizador, tentando assim evitar uma situação complicada em que os conteúdos pessoais do utilizador possam ficar em risco.

Um exemplo dessa situação aconteceu quando o Facebook decidiu eliminar todas as imagens cujo *upload* tinha sido feito através de uma biblioteca partilhada por várias aplicações, o KDE Image Plugin Interface (KIPI), sem qualquer aviso prévio [11].

Nem mesmo o facto de, mais tarde, esta rede social ter devolvido o conteúdo aos utilizadores afetados escondeu a necessidade de garantir que os conteúdos partilhados estejam em mais do que um local para evitar que situações como esta possam eliminar permanentemente todo o histórico de um utilizador, incluindo os conteúdos que partilhou.

Outra motivação prende-se com o facto de haver sempre a possibilidade de as contas de utilizador das redes sociais serem alvo de ataques maliciosos podendo haver um roubo de *password* ou mesmo a eliminação dos dados e conteúdos presentes na mesma, sendo que, com este trabalho, pretende-se dar a possibilidade ao utilizador de recuperar toda essa informação facilmente e sem pôr em risco toda a informação contida na rede social originalmente criada.

1.4. Introdução ao RDF

Para guardar o conteúdo relativo ao utilizador presente na rede social optou-se por usar o formato Resource Description Framework (RDF).

O RDF surgiu em 1999 e é um *standard* do World Wide Web Consortium (W3C) para a representação de informação na *web.* Permite, entre outras coisas, descrever recursos da *web*, como o autor, o título ou a data de uma página. Uma descrição RDF é referida como metadados ou "dados sobre dados".

O objetivo do RDF é normalizar a definição e o uso de metadados, mas pode perfeitamente servir para representar dados.

Na sua base, um objeto RDF é constituído por atributos que têm determinados valores. Outra maneira de pensar no RDF é imaginar a existência de dois nós, o objeto e o valor, e uma ligação existente entre eles, o atributo. Eventualmente, no lugar do valor, pode surgir uma ligação para outro objeto, permitindo aumentar o alcance da representação dos dados.

Em teoria, o RDF permitirá maior facilidade na partilha de dados e as suas descrições, já que, no caso específico do trabalho desenvolvido, permite a representação, de forma estruturada, da informação proveniente das redes sociais, tornando mais fácil não só a manipulação dessa informação, mas também a obtenção dos valores relevantes presentes na mesma por qualquer aplicação capaz de lidar com o formato.

Qualquer recurso é definido com um Uniform Resource Identifier (URI), ou seja, tanto o Uniform Resource Locator (URL) que define uma página de Internet, como qualquer endereço de uma página dentro da página original [12].

Os principais benefícios do RDF são [13]:

- A possibilidade de integrar cada vez mais e melhor os metadados nos recursos disponíveis na Internet;

- A utilização da sintaxe própria do RDF permite produzir e obter os dados de forma mais fácil e rápida, bem como facilitar a interação entre aplicações;
- As pesquisas apresentarão resultados mais precisos se forem baseadas nos metadados em vez dos índices das páginas como é feito atualmente, como é exemplo o Swoogle Semantic Web Search Engine [14];
- Agentes de *software* inteligente terão dados mais precisos.

Um exemplo de uma descrição RDF pode ser vista na Figura 1. Neste exemplo descreve-se uma página (www.tito.pt) com o título "Exemplo RDF" e com o autor "Tito Azevedo".

```
<?xml version="1.0"?>

<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:si="http://www.w3schools.com/rdf/">

    <rdf:Description rdf:about="http://www.tito.pt">
      <si:title>Exemplo RDF</si:title>
      <si:author>Tito Azevedo</si:author>
    </rdf:Description>

</rdf:RDF>
```

Figura 1 - Exemplo de descrição RDF

1.5. Requisitos

Numa fase inicial do trabalho, e antes de ser iniciado o desenvolvimento do mesmo, foram analisadas as várias possibilidades, as problemáticas e as melhores formas de abordar este projeto, sendo identificados os seguintes requisitos:

- Criação de uma página *web* para possibilitar a interação com o utilizador;
- Possibilidade de escolha entre diferentes redes sociais;
- Possibilitar a autenticação nos serviços utilizados através da página web criada, de modo a evitar que o utilizador tenha que abrir e fechar páginas ou a introduzir códigos textuais para garantir que a autenticação é feita;
- Possibilitar a salvaguarda da informação presente na rede social no disco rígido do computador utilizado para aceder à aplicação, mas também possibilitar o envio dessa informação para serviços de alojamento *online*;
- Gravar os dados em formato RDF;

- Conseguir ler os ficheiros RDF guardados pelo utilizador e, através deles, extrair os atributos e os valores respetivos;
- Criação de um *Web Service* que permita, através da aplicação *web* criada, executar os métodos que acedem à informação das redes sociais e efetuam as operações selecionadas pelo utilizador.

1.6. Estrutura da dissertação

Este documento contém cinco capítulos, bem como as respetivas referências.
Os capítulos presentes são:

- **Introdução** – pretende-se com este capítulo mostrar um pouco da história das redes sociais e o modo como foram evoluindo ao longo do tempo. Da mesma forma pretende-se analisar as motivações que levaram ao desenvolvimento deste trabalho, nomeadamente no que diz respeito à necessidade de ter cópias da informação partilhada de modo a garantir a segurança da mesma. São também analisados os objetivos apresentados para este trabalho;
- **Estado da arte** – neste capítulo analisam-se as opções adotadas durante o desenvolvimento do trabalho, nomeadamente no que diz respeito aos sistemas e *software* utilizados. É feito também um estudo das soluções existentes que possam ter aspetos em comum com o trabalho desenvolvido. É, ainda, feita uma descrição mais detalhada do RDF e do modo como este é usado neste trabalho e, por fim, uma breve análise à filosofia NoSQL usada para guardar os ficheiros RDF provenientes das redes sociais;
- **Arquitetura** – este capítulo pretende dar uma noção mais exata do que está por dentro deste trabalho. Inicialmente é feita uma análise das diferentes camadas que compõem a arquitetura da aplicação desenvolvida, bem como dos elementos integrantes de cada uma delas, incluindo os processos de autenticação e autorização necessários para interagir com as redes sociais e serviços utilizados, bem como as bibliotecas externas utilizadas;
- **Repositório de dados** – neste capítulo são mostrados exemplos da interação do utilizador com a aplicação *web* desenvolvida, bem como os resultados provenientes dessa interação;
- **Conclusão** – neste capítulo é feita uma análise crítica aos resultados obtidos, é feita uma comparação dos resultados com os objetivos iniciais e são analisadas as diferentes possibilidades de evolução do trabalho num futuro próximo.

2. Estado da arte

Neste capítulo pretende-se descrever as análises feitas às soluções existentes para permitir o desenvolvimento da aplicação. Ao mesmo tempo serão justificadas algumas escolhas feitas.

Numa fase inicial será feita uma pequena apresentação das diferentes possibilidades existentes atualmente no mercado para o desenvolvimento de uma aplicação *web*, bem como as vantagens de umas em relação a outras.

Será feita, posteriormente, uma análise às soluções existentes no mercado que possam apresentar semelhanças com alguns dos objetivos apresentados neste trabalho, bem como à maneira como estas podem, ou não, solucionar alguns dos problemas que foram surgindo no desenvolvimento do projeto.

De seguida será descrito de forma mais pormenorizada o RDF e a maneira como este pode ajudar a melhorar a maneira como é transmitido o conteúdo na WWW atual.

Por último é descrita a filosofia NoSQL com maior detalhe.

2.1. Desenvolvimento de páginas *web* dinâmicas

Páginas *web* dinâmicas são cada vez mais usadas nos dias que correm. Muitas empresas procuram pessoas capazes de desenvolver tais páginas.

Uma página *web* dinâmica, atualmente, tem quatro componentes muito importantes para o seu correto funcionamento:

- O Sistema Operativo que é essencialmente a plataforma onde todos os outros componentes vão correr;
- O servidor *web* que é a parte de *software* responsável por receber os pedidos de um *browser* de Internet, gerar a informação relevante com o mesmo e dar ao utilizador a resposta adequada. Para páginas Hypertext Markup Language (HTML) simples limita-se a retornar o HTML pedido, para páginas dinâmicas tem a necessidade de gerar o conteúdo baseado nos diferentes componentes presentes nas páginas, devolvendo, também um HTML;
- A Base de Dados (BD) que tem a função de guardar informação que pode ser usada, guardada e manipulada pela *framework*, descrita de seguida. Para uma página *web* pode ser usada para guardar conteúdo a mostrar ao utilizador e/ou para guardar informação relativa ao utilizador;

- Por último, a *framework* de aplicação *web*. Quando um *browser* faz um pedido de *script*, o servidor *web* transfere o controlo para uma zona dedicada à análise da linguagem, que vai ler um programa escrito na linguagem específica escolhida [15].

Existem fundamentalmente três plataformas para a construção de páginas *web* dinâmicas que variam nas diferentes combinações entre os componentes integrados:

- WISA, baseado em tecnologias Microsoft, usa como Sistema Operativo o Windows, como servidor *web* o Internet Information Services (IIS), como Base de Dados o Microsoft Structured Query Language (SQL) Server e como *framework* de aplicação *web* o Active Server Pages (ASP);
- LAMP, baseado em tecnologias *open source* que usa como Sistema Operativo o Linux, como servidor *web* o Apache, como Base de Dados o MySQL e como *framework* de aplicação *web* o Hypertext Preprocessor (PHP);
- Java Platform, Enterprise Edition (JEE), permite múltiplas opções na escolha dos componentes usados para a produção das páginas *web* dinâmicas. Para o Sistema Operativo pode usar Windows, Linux, entre outros. Para o servidor web existem algumas alternativas diferentes, com as suas vantagens e desvantagens. Alguns mais usados são o Glassfish, o Tomcat e o JBoss. Para a Base de Dados pode usar-se o Java Database Connectivity (JDBC), que permite a utilização de qualquer Base de Dados que suporte esta interface. Por último, existem diferentes *frameworks* de aplicação *web* que podem ser usadas, como JavaServer Pages (JSP) e JavaServer Faces (JSF).

Na Tabela 1 é mostrado um resumo destas 3 plataformas. No caso do Java EE, que foi a plataforma escolhida, apenas são mostrados os componentes usados e não todos os que estão disponíveis, excetuando a Base de Dados já que, como é referido de seguida, não existe a necessidade de utilizar uma BD para a salvaguarda de informação na aplicação desenvolvida.

Plataforma	Sistema Operativo	Servidor *web*	Base de Dados	*Framework*
WISA	Windows	IIS	SQL Server	ASP
LAMP	Linux	Apache	MySQL	PHP
Java EE	Windows	Glassfish	-	JSP

Tabela 1 - Resumo das três principais plataformas para páginas web dinâmicas

São ainda possíveis várias alternativas, podendo combinar diferentes componentes dos que foram referidos anteriormente. Podem ser usados, nomeadamente, como Sistema operativo o Unix, o Solaris ou o MacOs. Como servidor *web*, além das opções já referidas, podem ser usadas opções como o WebStar ou o Xitami, como Base de Dados podem ainda ser usados o Postgre SQL ou o IBM DB, entre outros. Por último, como *framework* de aplicação *web*, existem várias opções, entre as quais o Catalyst, o Django e o Ruby on Rails.

Na Tabela 2 é feita uma comparação mais detalhada entre diferentes aspetos a ter em conta quando se faz uma escolha entre as plataformas disponíveis para a criação de páginas *web*.

Área	WISA	LAMP	JEE
Licenciamento	Caro	Sem custos	Sem custos
Suporte e custo	Grátis via comunidade mas existem opções pagas		
Plataformas	Apenas Windows	Múltiplas	Múltiplas
Custos *hardware*	Servidores mais caros	Servidores baratos	Servidores caros
Alojamento externo	Muito disponível, mas mais caro	Muito disponível e barato	Pouco disponível
Segurança	Historicamente má, mas melhorada	Muito boa	Boa
Performance	*Hardware* mais caro	Muito boa	*Hardware* caro e configurações
Escalabilidade	Difícil escalar	Fácil escalar	Fácil escalar se bem configurado
Administração	Fácil	Difícil	Dificuldade moderada
Configuração	Fácil	Difícil	Dificuldade moderada
Flexibilidade de configuração	Pouco flexível	Extremamente flexível	Moderadamente flexível
Frameworks	Apenas uma	Muitas	Uma
Componentes	Largamente disponíveis		

Tabela 2 - Comparação mais detalhada sobre Plataformas *Web* [16]

Olhando para a tabela anterior pode ver-se que, apesar de tanto o WISA como o LAMP poderem, eventualmente, dar melhores garantias em algumas das áreas analisadas, o JEE acaba por ter um bom compromisso, sendo eficiente na maioria das áreas, o que pode possibilitar um desenvolvimento de aplicações sem grandes problemas e sem eventuais dificuldades acrescidas com a adição de novos componentes a utilizar.

O WISA acaba por ser a opção mais usada. Não só por usar muitas tecnologias da Microsoft, que são sobejamente conhecidas pela maioria das pessoas, mas também pelo suporte que poderão ter no caso do surgimento de questões relacionadas com as ferramentas usadas.

Por outro lado surge o LAMP, que tem um custo muito mais baixo do que o WISA, já que está assente em tecnologias *open source*, mas tem a desvantagem de não ter tanto suporte. Apesar de tudo, não havendo um grande suporte por parte das empresas responsáveis pelo desenvolvimento da ferramenta, há grandes comunidades disponíveis a ajudar e responder a eventuais questões que apareçam. Têm surgido grandes empresas, ultimamente, a usar esta tecnologia, como por exemplo o Facebook.

Por último surge o Java Platform, Enterprise Edition (Java EE) que tem como principal arma o facto de utilizar linguagem Java, que é, hoje em dia, muito usada e uma das mais populares. Por maior familiaridade com esta linguagem, optou-se por usar esta plataforma.

2.2. Análise às alternativas existentes no mercado

Nesta subsecção pretende-se analisar as opções existentes que possibilitam ao utilizador salvaguardar o conteúdo de uma ou mais redes sociais. Não foi encontrada nenhuma solução que desse ao utilizador todas as possibilidades que este projeto pretende dar, como tal não houve o risco de estar a criar um trabalho duplicado. No entanto existem algumas soluções que permitem completar alguns dos objetivos identificados para este trabalho e que, por isso, devem ser alvo de um estudo adequado.

2.2.1. Facebook

Durante o desenvolvimento do trabalho verificou-se a existência no Facebook de uma opção, embora esta seja de difícil visibilidade, para transferir para o disco rígido uma cópia dos dados da conta do utilizador.

Numa primeira instância o seu objetivo parece chocar com o objetivo deste trabalho, mas a verdade é que se limita a guardar a informação do utilizador em formato HTML e em termos de conteúdo multimédia apenas guarda as imagens do utilizador.

Relativamente a este trabalho perde por não guardar a informação num formato que possibilite a obtenção dos atributos e dos valores dos conteúdos do utilizador, bem como por não possibilitar ao utilizador uma cópia local dos vídeos que estejam alojados no próprio sistema de vídeo do Facebook. Também não providencia informação acerca dos contactos presentes na conta já que, segundo o próprio Facebook "Focámo-nos em providenciar uma maneira para as pessoas normais poderem fazer o *download* de uma cópia do seu perfil. Informação acerca dos contactos não é algo que faça parte do perfil atualmente" [17].

Terá apenas o objetivo de possibilitar ao utilizador salvaguardar os conteúdos para consulta própria, já que não parece ter o objetivo de permitir a exportação desses dados para a rede social. Fazendo a experiência com uma conta pessoal, outro fator negativo desta solução foi o facto de demorar algum tempo a gerar o arquivo para *download*. Nos dois casos testados, utilizando para isso uma conta normal de Facebook e uma conta criada para testar a aplicação desenvolvida, com pouco conteúdo, foi necessário esperar cerca de duas horas até ser possível obter o ficheiro gerado, levando a pensar que o tempo necessário para a disponibilização da informação depende da quantidade de pedidos pendentes no servidor.

2.2.2. SocialSafe

Após alguma pesquisa surgiu outra solução com um objetivo semelhante ao do trabalho, o SocialSafe.

Segundo a página *web* dedicada a esta aplicação, esta possibilita fazer o *download* e unificar as contas de Facebook, Facebook Pages, Twitter, Instagram, LinkedIn, Google+ e Viadeo. Tem as seguintes características [18]:

- É um modo agradável de salvaguardar a vida social *online*;
- Permite exportar fotos, amigos, *updates* e mais;
- Permite pesquisar no tempo, entre todas a redes;
- Pesquisar no diário digital;
- *Download* diretamente para o PC ou Mac.

Segundo aquilo que é observável, o que este serviço faz é criar uma aplicação local que reúne todas as redes sociais em que o utilizador esteja registado e mostra a informação misturada de todas elas num só sítio, bem como todos os eventos que o utilizador tenha registado *online* ordenados por data. Permite ainda pesquisar por palavras-chave dentro da informação disponibilizada.

Apesar disso, olhando para as características apresentadas inicialmente, vê-se que é possível exportar a informação, mas apenas para o disco. Não é possível exportar a informação para uma rede social, ou mesmo migrar entre redes sociais, como a informação apresentada na página pode dar a entender. Além disso só permite guardar os dados das redes sociais no disco rígido, não permitindo, por exemplo como é um dos objetivos deste trabalho, colocar conteúdos em serviços de alojamento *online* de forma automática.

É sem dúvida uma aplicação interessante, especialmente pelo aspeto que tem, já que possui uma interface visual muito apelativa, mas, mais uma vez, não tem uma relação muito direta com este trabalho, já que tem objetivos algo diferentes. É de destacar ainda o facto de possibilitar a interação com diversas redes sociais, o que não é muito frequente nas aplicações de *backup* que se encontram.

Tem ainda um ponto muito negativo, que é o facto de ser uma aplicação paga. Este facto afastará seguramente muita gente desta aplicação, não permitindo que possa haver mais cópias da informação [18].

2.2.3. ArchiveFacebook

Existe um *add-on* para o Mozilla Firefox que permite fazer um *backup* dos dados do Facebook. A ideia por trás desta extensão é ter uma maneira de guardar no disco rígido a informação proveniente do Facebook e navegar nela como se o utilizador estivesse atualmente no seu Facebook real. Tenta reproduzir uma cópia local da conta.

O ArchiveFacebook guarda a seguinte informação do perfil do Facebook:

- Fotografias;
- Mensagens;
- Atividade do utilizador;
- Lista de amigos;
- Notas;
- Eventos;
- Grupos;
- Informação.

Mais uma vez, acaba por ter objetivos relativamente diferentes dos objetivos propostos inicialmente para este trabalho, nomeadamente por só interagir com uma das redes sociais, neste caso o Facebook, também por guardar apenas localmente a informação, neste caso também não guarda o Mural do utilizador, isto é, o conjunto de conteúdos e informações publicados pelo utilizador no seu espaço público de acesso e por não permitir qualquer tipo de exportação desta informação.

Tem ainda a desvantagem de ser uma extensão apenas do Mozilla Firefox, impedindo utilizadores de outro *browser* de usar esta aplicação [19].

2.2.4. Backupify

Backupify é um serviço bastante interessante que permite efetuar *backups* regulares de diversas páginas *web* e guardá-las num serviço de *cloud* onde estes podem estar acessíveis ao utilizador. Possibilita, entre outras opções, efetuar o *backup* do Google Apps, Facebook, Twitter e LinkedIn.

Tem a vantagem de permitir *backups* automáticos da informação. Para guardar os dados usa os *Web Services* disponibilizados pela Amazon, nomeadamente no Amazon Simple Storage Service (S3) que guarda os objetos em múltiplos dispositivos com redundância, de modo a garantir que os dados se mantêm mesmo que existam falhas nos dispositivos já que conseguem detetar e reparar quaisquer redundâncias perdidas. Além de tudo, necessita apenas da criação de uma conta, podendo esta ter associadas diversas contas de outros serviços.

Segundo a sua página [20], tem as seguintes características:

- Proteção adicional para as fotografias e vídeos online que são insubstituíveis;
- Opções de *download* e *export* para os *updates* do Twitter e Facebook;
- *Backup* centralizado para os dados do Gmail, Google Drive e Google Calendar.

Os passos necessários para fazer um *backup*, segundo a página [20], são:

- Criação de uma conta, escolhendo a opção que melhor se adeque à utilização que vai ser dada;
- Configurar os *backups*, escolhendo as redes sociais de onde se pretende recolher a informação. De seguida, na própria rede social, terá de ser dada autorização para a aplicação aceder à informação pessoal, tal como acontece também neste trabalho;
- De seguida é necessário esperar que a aplicação recolha toda a informação das redes sociais selecionadas;
- Quando o passo anterior estiver completo é possível navegar pelas redes sociais e verificar os dados guardados;
- Por último é possível efetuar o *download* dos dados para a máquina local e também é possível restaurar a informação do Gmail ou do Google Drive com um simples clique.

Tem a grande vantagem de usar um serviço de *cloud* como é o Amazon S3 e, para um utilizador que já tenha acesso a um servidor Amazon S3, permite a utilização do servidor próprio. Por outro lado permite também interagir com diversas redes sociais, o que também é extremamente positivo. Permite, além de guardar os dados na *cloud*, efetuar o *download* dos mesmos para o disco. Extremamente útil é o facto de efetuar *backups* regulares de forma automatizada. É a primeira aplicação encontrada que possibilita esta funcionalidade que é, sem dúvida, muito interessante.

Tem mais uma vez o problema de não permitir exportar os dados para as redes sociais, não servindo os dados para nada além de consulta. Por outro lado, o número de redes sociais disponibilizadas é muito limitado, parecendo ser um serviço mais dedicado a outro tipo de interações, nomeadamente com Google Apps, um serviço disponibilizado pelo Google que fornece o acesso a vários produtos desenvolvidos pela empresa e com o Salesforce, uma empresa dedicada ao *cloud computing* [21].

Além disso, é um serviço que tem funcionalidades limitadas para um utilizador que não pague. Na Tabela 3 são mostradas as diferenças entre os tipos de conta no Backupify.

Nome	Personal	MyCloud 100	MyCloud 500
Serviços	3 contas	5 contas	25 contas
Limite de espaço	1 GB	10 GB	50 GB
Backups	Semanais	Diários	Diários
Opções de Suporte	E-mail	E-mail, Telefone e *web*	E-mail, Telefone e *web*
Preço	Grátis	$4.99/mês	$19.99/mês

Tabela 3 - Diferenças entre os tipos de conta no Backupify [20]

Analisando a tabela verifica-se que, para um utilizador casual que pretenda fazer ocasionalmente o *backup* dos seus dados pessoais, parece ser perfeitamente acessível o perfil gratuito. O utililzador estará limitado a três contas, mas como o Backupify não tem suporte para muitas redes sociais, não será muito relevante esse facto [20].

2.2.5. Pick&Zip

Pick&Zip é um *software* gratuito que permite fazer o *backup* e guardar fotografias e vídeos do Facebook.

Segundo a página *web*, as principais caraterísticas desta aplicação são:

- Possibilidade de fazer o *download* de fotografias onde o utilizador foi identificado, mesmo que não tenha sido o próprio a colocar essa fotografia *online*;
- Opção de fazer o *backup* das fotografias do Facebook, podendo facilmente efetuar o *download* de todos os álbuns e fotos onde a pessoa foi identificada;
- Seleção das imagens preferidas do utilizador entre as suas fotografias, as da sua página de fãs e dos grupos a que pertence;
- Possibilidade de efetuar o *download* de vídeos do perfil do utilizador, do perfil de amigos ou de vídeos em que o utilizador tenha sido identificado.

Analisando as possibilidades dadas por esta aplicação é notório que tem algumas diferenças relativamente ao trabalho desenvolvido.

Primeiramente apenas lida com uma rede social, o que não acontece com este trabalho. Seguidamente apenas trata de guardar as imagens e vídeos do Facebook, não guardando o restante conteúdo como lista de amigos, mensagens privadas, informações pessoais, conteúdos partilhados pelo utilizador no seu mural, etc.

Não dá também quaisquer possibilidades de guardar a informação em locais diferentes do disco rígido do utilizador.

Por último, não permite exportar a informação para uma rede social, servindo apenas para guardar localmente os ficheiros multimédia.

O facto de ser gratuita é um ponto a seu favor, mas torna-se claro que as funcionalidades desta aplicação são bastante limitadas, tendo em conta o que se pretende dela [22].

2.2.6. BackupMyTweets

BackupMyTweets apresenta-se dando a conhecer ao utilizador que o Twitter apenas disponibiliza os 3200 *tweets*, mensagens publicadas nesta rede social, mais recentes de cada utilizador. Com esta aplicação torna-se possível guardar todos os *tweets* publicados pelo utilizador, não estando dependente de limitações impostas pelo Twitter.

As principais características deste serviço são:

- Transformar os *tweets* do utilizador em algo permanente, já que podem ser guardados evitando que o Twitter possa torná-los indisponíveis;
- *Backups* diários, de modo a procurar todos os dias por nova informação na conta de Twitter associada;
- Possibilidade de pesquisar através de todos os *tweets*;
- Segurança para escrever tudo o que a pessoa quiser, já que essa informação ficará para sempre guardada.

Este serviço é gratuito, mas tem uma versão paga que, além das caraterísticas referidas anteriormente, adiciona ainda as seguintes:

- *Backup* da lista de amigos, bem como da lista de pessoas que seguem o utilizador associado com a rede social;
- *Backup* de toda a *timeline*, isto é, dos *tweets* mais recentes não só do utilizador, mas também das pessoas que segue;
- 1 GB de espaço para guardar a informação, o que equivale a cerca de um milhão de *tweets*;
- *Backup* por tópico, permitindo guardar apenas a informação associada a um tema específico;
- Interface de pesquisa para permitir uma melhor procura dos *tweets* guardados.

Apesar de este serviço ter uma versão gratuita, pelo *feedback* obtido no espaço para troca de mensagens entre utilizadores e *staff*, esta não tem um funcionamento correto, não permitindo, por exemplo, obter os dados guardados na *cloud* depois de efetuado o *backup*, acabando por ser obrigatório obter a versão paga para conseguir recuperar a informação do Twitter.

Mais uma vez, como se dedica apenas a uma rede social, acaba por ter proporções distintas do projeto desenvolvido. Também não permite exportar a informação para a rede social usada e apenas guarda a informação textual, não havendo qualquer indicação relativamente a conteúdo multimédia.

Embora isso não seja referido diretamente pelo serviço, pelo que é possível verificar, deverá guardar as informações num serviço de *cloud*, o que é um ponto positivo, mas o facto de não ter uma versão gratuita a funcionar corretamente torna este serviço inacessível ou pouco apelativo a um grande número de utilizadores [23].

2.2.7. TweetBackup

Outra aplicação interessante é o TweetBackup que tem um funcionamento bastante parecido ao BackupMyTweets apresentado anteriormente mas que, como funciona através do Backupify que também foi apresentado, torna desnecessária a repetição da informação anterior.

2.3. Resource Description Framework (RDF)

Já foi feita uma pequena introdução ao Resource Description Framework. Neste capítulo pretende-se definir melhor esta tecnologia, bem como a utilidade que ela tem para o trabalho desenvolvido.

Antes de falar sobre o RDF deve-se definir o conceito, intrínseco a este, de *Web* Semântica. Segundo o documento “The Semantic Web: An Introduction” [24] a *Web* Semântica é uma extensão ao World Wide Web (WWW) que permite que as pessoas partilhem conteúdo para além dos limites do que é possível com uma aplicação ou uma página. Não é mais do que uma rede de informação ligada de uma maneira que torna fácil o processamento por máquinas. Pode olhar-se para a *Web* Semântica como uma forma eficiente de representar dados na WWW ou como uma Base de Dados globalmente ligada.

A *Web* Semântica foi definida originalmente por Tim Berners-Lee, o inventor do WWW, do Uniform Resource Identifier (URI), do HTTP e do HTML e existe uma equipa dedicada no World Wide Web Consortium (W3C) responsável por melhorar, estender e normalizar o sistema. Apesar de já terem sido desenvolvidas várias linguagens, publicações e ferramentas, as tecnologias ligadas à *Web* Semântica ainda estão numa fase embrionária e, apesar de haver um futuro promissor no projeto, a verdade é que há pouco consenso acerca do melhor caminho a tomar e das melhores opções.

A lógica deste sistema está no facto de os dados usados em alguns contextos estarem geralmente escondidos em ficheiros HTML, o que dificulta a sua publicação, bem como a obtenção dos mesmos. Pensando num caso prático, se fosse necessário representar os dados estatísticos sobre a Liga Portuguesa de Futebol Profissional, recorrendo unicamente ao HTML, seria tremendamente difícil usar estes dados de todas as maneiras necessárias.

A verdade é que, a um nível lógico, quanto mais fácil for representar dados, mais pessoas o vão querer fazer, e este efeito é uma das premissas por trás do desenvolvimento da *Web* Semântica.

A *Web* Semântica lida, geralmente, com Uniform Resource Identifiers (URIs), simples identificadores na *Web*, geralmente iniciados por “http” ou “ftp”, para representar os dados, geralmente com estruturas baseadas em triplos. É aqui que entra o RDF.

O RDF é particularmente adequado para representar metadados, dados sobre outros dados, acerca de recursos *web* como, por exemplo, o título, o autor e a data de modificação de uma página *web*. Pode ainda ser usado para representar informação acerca de elementos que podem ser identificados na *web*, mas não podem ser diretamente obtidos nela, como por exemplo informação acerca de itens disponíveis em lojas *online*, como o preço, disponibilidade, etc.

O RDF é útil quando esta informação necessita de ser processada por aplicações em vez de ser apenas mostrada às pessoas. Providencia uma *framework* comum para expressar a informação de modo a poder ser trocada entre aplicações sem perda de significado. Esta possibilidade faz com que a informação possa chegar a aplicações diferentes daquela onde foi originalmente criada.

O RDF identifica os elementos através de URIs e descreve os recursos com propriedades simples e valores, o que permite ao RDF representar declarações simples sobre recursos como grafos de nós e arcos que representam os recursos e as suas propriedades e valores.

Na Figura 2 mostra-se o exemplo de uma representação em grafo do seguinte conjunto de declarações: "existe uma Pessoa identificada por http://www.w3.org/People/EM/contact#me, cujo nome é Eric Miller, cujo e-mail é em@w3.org e que tem o grau de doutor".

Figura 2 - Grafo RDF que descreve Eric Miller [25]

Esta figura ilustra que o RDF usa URIs para identificar:

- Indivíduos, como por exemplo Eric Miller, identificado por http://www.w3.org/People/EM/contact#me;
- Tipos de elementos, como por exemplo Person identificado por http://www.w3.org/2000/10/swap/pim/contact#Person;
- Propriedades desses elementos, como por exemplo mailbox, identificado por http://www.w3.org/2000/10/swap/pim/contact#mailbox;
- Valores dessas propriedades, como por exemplo mailto:em@w3.org como valor da propriedade mailbox. O RDF também usa conjuntos de caracteres como "Eric Miller" e valores de outros tipos de dados como números inteiros e datas para os valores das propriedades.

O W3C tratou de desenvolver uma serialização XML do RDF, denominada de RDF XML, que é considerado o formato *standard* para o intercâmbio de informação do RDF na *Web* Semântica, embora não seja o único formato. Existem outros exemplos, como o Notation3 que é uma excelente alternativa para uma serialização de texto simples.

A partir do momento em que a informação está no formato RDF, torna-se mais fácil processá-la, dado que o XML é um formato genérico que já tem bastantes *parsers*. O RDF XML requer algum estudo e alguns conhecimentos prévios de XML e *namespaces*, mas será analisado um pequeno exemplo.

Na Figura 3 temos a representação do grafo da Figura 2 no formato RDF/XML.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>

</rdf:RDF>
```

Figura 3 - Representação em RDF/XML do grafo anterior

De notar que, também neste formato, o RDF contém os URIs bem identificados, bem como propriedades como mailbox e fullName (de forma abreviada) e os seus valores, respetivamente em@w3.org e Eric Miller.

Como o HTML, este RDF/XML é processável por máquinas e, usando URIs, pode ligar partes de informação pela *Web* mas, ao contrário do HTML convencional, os URIs do RDF podem referenciar qualquer elemento identificável, incluindo elementos que não possam ser diretamente obtidos na *Web* (como a pessoa Eric Miller). O resultado é que, além de ser possível descrever elementos como páginas *web*, o RDF também consegue descrever carros, negócios, pessoas, eventos, etc. Adicionalmente, as propriedades RDF têm URIs que identificam com precisão as relações que existem entre os objetos ligados.

Por exemplo, uma forma simples de dizer que alguém com o nome Tito Azevedo criou uma página *web* seria: "**http://www.example.org/index.html** tem um **criador** cujo valor é **Tito Azevedo**"

Algumas partes desta frase devem ser enfatizadas de modo a ilustrar que, para descrever as propriedades de algo, há necessidade de nomear ou identificar um número de elementos:

- O elemento que a frase descreve, neste caso a página *web*;
- Uma propriedade específica do elemento que a frase descreve, neste caso o criador;
- O elemento que a frase diz ser o valor da propriedade do elemento que a frase descreve, neste caso quem o criador é.

Nesta frase o Uniform Resource Locator (URL) da página é usado para a identificar. A palavra criador é usada para identificar a propriedade e as duas palavras “Tito Azevedo” são usadas para identificar a pessoa que é o valor desta propriedade.

Outras propriedades desta página podiam ser descritas usando afirmações da mesma forma. Por exemplo:

- **“http://www.example.com/index.html** tem uma **data de criação** com o valor **14 de Fevereiro de 2012”**;
- “**http://www.example.com/index.html** tem uma **língua** cujo valor é **Português**”.

O RDF é baseado na ideia de que os elementos descritos têm propriedades que têm valores e que os recursos podem ser descritos criando frases semelhantes às anteriores que especificam as propriedades e os seus valores. O RDF usa uma terminologia própria para falar acerca das várias partes das afirmações. Especificamente, a parte que identifica o elemento que é o tema da frase, neste caso a página *web*, é chamado de sujeito. A parte que identifica a propriedade ou característica do sujeito que a declaração especifica, criador, data de criação e língua, nos exemplos, é chamado de predicado e a parte que identifica o valor dessa propriedade é chamado de objeto.

Analisando a frase original “**http://www.example.org/index.html** tem um **criador** cujo valor é **Tito Azevedo**”, temos:

- Um sujeito cujo URL é http://www.example.org/index.html;
- Um predicado que é a palavra “criador”;
- Um objeto que é a frase “Tito Azevedo”.

Para tornar este tipo de frases adequadas para o processamento por máquinas, são necessárias duas condições:

- Um sistema de identificadores processáveis por máquinas para identificar sujeitos, predicados ou objetos numa frase sem haver qualquer confusão com identificadores de aspeto parecido usados noutro local da *web*;
- Uma linguagem processável por máquinas para representar essas frases e trocá-las entre máquinas.

Felizmente a *web* atual providencia ambas as características.

Como dito anteriormente, a *web* já permite usar um tipo de identificador, o URL, que foi usado no exemplo para identificar a página *web* criada. Não é mais do que um conjunto de caracteres que identifica um recurso *web* ao representar o seu mecanismo de acesso primário. Contudo existe a necessidade de guardar informação sobre outros elementos que não podem ser descritos através de URLs.

A *web* contém ainda uma forma mais geral de identificação, o URI. Os URLs são um caso particular de URI. Os URIs têm a vantagem de não estar limitados a identificar elementos que tenham localizações na rede associadas. Devido a esta grande vantagem o RDF usa URIs como a base da identificação dos sujeitos, predicados e objetos. Para ser mais preciso, o RDF usa URI References (URIS) que não são mais do que URIs com um fragmento opcional de identificação. Por exemplo o URIS http://www.example.org/index.html#section2 consiste no URI http://www.example.org/index.html e, separado pelo caracter #, o fragmento identificador section2. O RDF define um recurso como sendo tudo o que é possível identificar através de URIS, pelo que se torna possível identificar praticamente tudo e, ao mesmo tempo, definir as relações entre os elementos identificados.

Em RDF, a frase "**http://www.example.org/index.html** tem um **criador** cujo valor é **Tito Azevedo"** pode ser representada por uma declaração RDF com os seguintes elementos:

- Como sujeito terá http://www.example.org/index.html;
- Como predicado terá http://purl.org/dc/elements/1.1/creator;
- Como objeto poderá ter, como exemplo, http://www.example.org/staffid/85740.

De destacar que foram usados URIS não só para definir o sujeito da frase original, mas também para o predicado e para o objeto, em vez dos conjuntos de caracteres anteriormente usados.

Como já foi referido anteriormente, o RDF pode ser representado por nós e arcos. Neste modelo uma frase será representada por:

- Um nó para o sujeito;
- Um nó para o objeto;
- Um arco para o predicado, direcionado do nó do sujeito para o nó do objeto.

A frase RDF descrita anteriormente poderia ser descrita pelo grafo da Figura 4.

Figura 4 - Uma frase simples em RDF [25]

Grupos de frases são representados pelos conjuntos de nós e arcos correspondentes. Desse modo, a Figura 5 representa as frases seguintes:

- **"http://www.example.org/index.html** tem um **criador** cujo valor é **Tito Azevedo";**
- **"http://www.example.com/index.html** tem uma **data de criação** com o valor **14 de Fevereiro de 2012"**;
- "**http://www.example.com/index.html** tem uma **língua** cujo valor é **Português**".

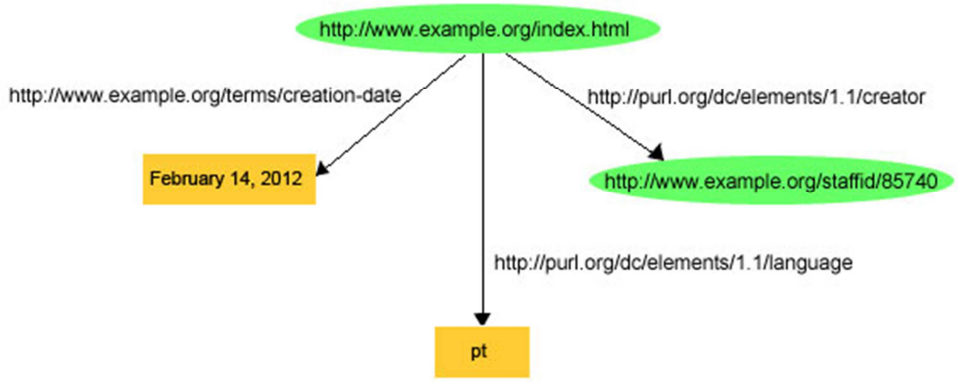

Figura 5 - Múltiplas frases sobre o mesmo recurso

Neste exemplo demonstra-se que os objetos em RDF podem ser URIS ou valores constantes, ou literais, representados por conjuntos de caracteres. Os literais não podem ser usados como sujeitos ou predicados em frases RDF. Na representação de grafos em RDF, os literais são mostrados dentro de um retângulo e os URIS dentro de uma elipse.

Uma maneira alternativa de representar esta informação é através do uso de triplos. Nesta notação cada frase de um grafo é representada como um simples triplo composta por sujeito, predicado e objeto, respetivamente. As três frases presentes na Figura 5 seriam representadas, segundo esta notação, pelos triplos representados na Figura 6.

```
<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/creator> <http://www.example.org/staffid/85740> .
<http://www.example.org/index.html> <http://www.example.org/terms/creation-date> "February 14, 2012" .
<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/language> "pt" .
```

Figura 6 - Representação em triplos das frases anteriores

A cada triplo corresponde um único arco no grafo, completo com o nó inicial e o nó final, respetivamente o sujeito e o objeto. A informação presente nos triplos, apesar de ter um aspeto diferente, representa exatamente a mesma informação presente nos grafos, já que o que é fundamental para o RDF é o modelo de grafo das frases, a notação usada para representar essa informação é irrelevante.

Para evitar a escrita completa dos URIS, o que poderia ser, nalguns casos, pouco apelativo e ocupar um tamanho excessivo em cada linha, por conveniência usam-se prefixos. Por exemplo para representar o *namespace* http://www.w3.org/1999/02/22-rdf-syntax-ns# usa-se habitualmente o prefixo rdf:, para identificar o *namespace* http://www.example.org/ usa-se o prefixo ex:, etc.

Usando a notação com prefixos, esta torna-se bastante mais legível. A informação presente na Figura 6 poderia ficar como está ilustrado na Figura 7.

```
ex:index.html dc:creator            exstaff:85740 .

ex:index.html exterms:creation-date "February 14, 2012" .

ex:index.html dc:language           "pt" .
```

Figura 7 - Representação de triplos com prefixos

Para demonstrar apenas a facilidade da criação de ficheiros RDF, na Figura 8 está um possível documento RDF criado.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
    xmlns:cv="http://purl.org/captsolo/resume-rdf/0.2/cv#"
    xmlns:teach="http://linkedscience.org/teach/ns#"
    xmlns:dcterms="http://purl.org/dc/terms/">
  <rdf:Description rdf:about="">
    <dc:creator rdf:parseType="Resource">
        <foaf:name> Tito Azevedo </foaf:name>
        <vcard:EMAIL>titoazevedo@ua.pt</vcard:EMAIL>
        <cv:studiedIn>Universidade de Aveiro</cv:studiedIn>
    </dc:creator>
    <dc:title> Repositório de dados pessoais das Redes Sociais </dc:title>
    <teach:teacher> Joaquim Arnaldo Carvalho Martins </teach:teacher>
    <dcterms:date> 01/01/2012 </dcterms:date>
  </rdf:Description>
</rdf:RDF>
```

Figura 8 - Documento RDF que descreve esta dissertação

Este documento RDF define um artigo com o título “Repositório de dados pessoais das Redes Sociais” que foi escrito por alguém com o nome “Tito Azevedo”, cujo e-mail correspondente é titoazevedo@ua.pt e que estudou na Universidade de Aveiro. Descreve também o professor responsável com o valor “Joaquim Arnaldo Carvalho Martins”. Relativamente à data, como apenas é importante o ano, tanto o dia como o mês foram colocados com o valor 1. Este documento podia facilmente ser usado para descrever esta dissertação. Este ficheiro produz os triplos identificados na Figura 9.

```
<> <http://purl.org/dc/terms/creator> _:x0 .
this <http://purl.org/dc/terms/title> "Repositório de dados pessoais das Redes Sociais" .
this <http://linkedscience.org/teach/ns#teacher> "Joaquim Arnaldo Carvalho Martins" .
this <http://purl.org/dc/terms/date> "01/01/2012" .
_:x0 <http://xmlns.com/foaf/0.1/name> "Tito Azevedo" .
_:x0 <http://www.w3.org/2001/vcard-rdf/3.0#EMAIL> "titoazevedo@ua.pt" .
_:x0 <http://purl.org/captsolo/resume-rdf/0.2/cv#studiedIn> "Universidade de Aveiro" .
```

Figura 9 - Triplos gerados pelo ficheiro RDF

Este formato não é mais do que a serialização em texto simples Notation3 referida anteriormente e permite identificar facilmente os elementos presentes no RDF. Dependendo da preferência do utilizador, poderão ser usados quaisquer formatos, desde que os mesmos sejam válidos.

A verdade é que, com o RDF, a informação é mapeada diretamente e sem ambiguidade a um modelo, modelo esse que é descentralizado e que possui, como já foi referido, diversos *parsers* capazes de extrair a informação relevante. Isto significa que, quando se tem uma aplicação que lida com o RDF, sabe-se que dados são relevantes para a semântica da aplicação e que dados não o são. Mais do que isso, não é só a pessoa que o sabe, é toda a gente que lide com a aplicação e muitas vezes sem sequer ler a especificação, por ser tão conhecido. Por outro lado, existe o objetivo de que o RDF se torne parte da *Web* Semântica, o que só pode trazer benefícios à sua utilização. Pode comparar-se o uso atual do RDF com o uso do HTML nos tempos primordiais da *Web* [24,25].

2.4. NoSQL

Uma filosofia cuja utilização tem crescido de modo bastante rápido é o NoSQL. Organizações como o Facebook e o Twitter são exemplos da utilização do NoSQL para obter uma maior escalabilidade e performance, com um menor custo relativamente a um sistema relacional.

No trabalho desenvolvido, como já foi referido, existe um repositório digital que vai guardar a informação gerada em RDF numa Base de Dados NoSQL. Como é dada uma abstração a este tipo de linguagem, necessitando apenas de comunicar com o repositório via REST, não existe a necessidade de conhecer ao pormenor o funcionamento das diferentes tecnologias que utilizam NoSQL, no entanto torna-se necessário fazer uma introdução a esta filosofia, de modo a demonstrar a sua eficácia na situação em que é usada.

O termo NoSQL não pretende rejeitar o SQL, sendo que o “No” representa “Not Only”, ou seja, não se limita a utilizar SQL, tentando compensar algumas das falhas existentes no mesmo.

O uso do NoSQL em empresas como Amazon, Google, LinkedIn, Facebook, Twitter, entre outras, deveu-se, sobretudo, à necessidade de melhorar as tecnologias existentes na altura, de modo a que estas pudessem satisfazer os requisitos destas empresas. Não se limitaram a recusar o SQL, mas sim a encontrar uma maneira de melhorar esta tecnologia.

Nos dias de hoje, as necessidades ao nível das Bases de Dados são diferentes daquelas que ditaram o uso recorrente do SQL. Não só há mais dados, mas estes têm uma variabilidade muito maior e as BDs necessitam de suportar esta variabilidade sem ser necessário proceder a uma reconfiguração em cada alteração existente. No âmbito do trabalho, é fundamental o uso do NoSQL devido à existência de campos variáveis, o que tornaria o uso de uma tecnologia relacional impossível ou extremamente difícil.

No ano 2000 o número de organizações disponíveis na Internet, bem como o número de funcionalidades disponibilizadas teve um grande crescimento, levando a um aumento dos pontos de pressão em aplicações SQL. Estes pontos de pressão não eram mais do que tecnologias e/ou filosofias que eram necessárias e que a tecnologia SQL necessitava de garantir para que a sua utilização fosse adequada. Na Figura 10 está uma representação de alguns desses pontos de pressão identificados.

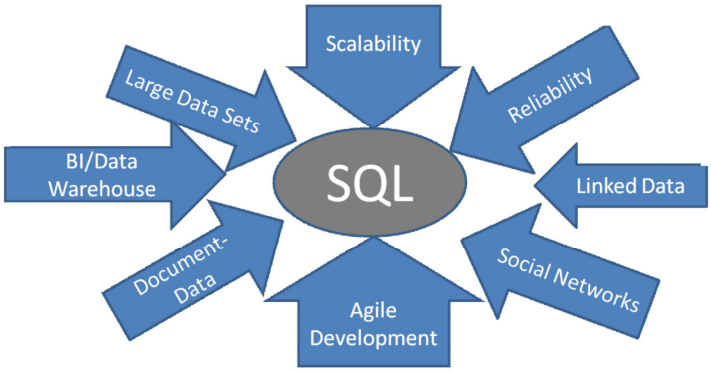

Figura 10 - Pontos de pressão da tecnologia SQL [26]

O facto de, muitas vezes, não ser possível garantir os pontos identificados na Figura 10 na utilização de um sistema baseado em SQL levou a que muitos programadores começassem a explorar diferentes possibilidades, levando ao surgimento de soluções que complementavam o uso regular do SQL.

Ao longo do tempo foram surgindo evoluções e novas tecnologias que permitiram uma evolução no NoSQL. Alguns dos principais marcos desta filosofia estão presentes na Tabela 4.

Evento	Data
Microsoft publica *standard* MDX	1997
Google MapReduce	Novembro de 2004
Google BigTable	Novembro de 2006
Especificação W3C XQuery 1.0	2006
Amazon Dynamo	Outubro de 2007
Yahoo/Amazon Hadoop	2008

Tabela 4 – Principais eventos na evolução do NoSQL [26]

Em 1997, ainda antes do ano 2000 referido anteriormente, surgiu uma nova linguagem, denominada de MultiDimensional Expressions (MDX) que, como o nome indica, foi criada para a representação de expressões multidimensionais. Esta linguagem foi introduzida pela Microsoft.

Em 2004 surgiu o MapReduce, um marco bastante importante na história do NoSQL já que este algoritmo permitia transformar cópias de dados *web* para pesquisa indexada usando CPUs de baixo custo.

Em 2006, continuando o sucesso obtido com o MapReduce, a Google lançou o BigTable, que confirmou que era possível criar sistemas de dados de grande escala sem necessitar de recorrer ao SQL.

Em 2006 surgiu também a especificação do XQuery por parte do W3C. O XQuery era uma das primeiras linguagens que permitia facilmente obter informação de dados guardados em tabelas e em documentos.

O Amazon Dynamo surgiu em 2007 e foi uma tecnologia baseada em NoSQL criada pela Amazon para uso interno, de modo a conseguir lidar com os problemas de escalabilidade [27].

O Yahoo/Amazon Hadoop, criado em 2008, foi uma combinação de dois grandes componentes de *software*, um gestor de ficheiros distribuídos conhecido como HDFS e o MapReduce, já definido anteriormente.

Quando se avalia o NoSQL ou qualquer outro sistema distribuído, é comum referir-se o teorema CAP [28]. Este teorema foi proposto, em 2000, por Eric Brewer, que defendia que um sistema distribuído não consegue garantir consistência, disponibilidade e tolerância a partições em simultâneo. CAP é definido como:

- **Consistência:** Todos os nós veem os mesmos dados ao mesmo tempo;
- **Disponibilidade:** Uma garantia de que todos os pedidos recebem uma resposta acerca do sucesso ou insucesso da operação;
- **Tolerância a partições:** O sistema continua a funcionar, mesmo que sejam perdidas mensagens arbitrariamente.

Tal como referido, o teorema defende que não é possível ter estas três condições em simultâneo, podendo, no máximo, ter duas.

Existem, atualmente, alguns produtos que fazem parte do mercado de Bases de Dados NoSQL. Embora a quantidade de produtos disponíveis tornasse impossível a sua caracterização e descrição, na Tabela 5 encontra-se um resumo de algumas das principais soluções existentes. Algumas das soluções, nomeadamente o MongoDB e o Neo4j foram utilizadas no repositório digital que guarda os documentos RDF.

	MongoDB	Neo4j	Redis	Cassandra	HBase
Linguagem	C++	Java	C++	Java	Java
Licença	AGPL	AGPL	BSD	Apache	Apache
Modelo	Documento	Grafo	Chave/valor	Coluna	Coluna
Protocolo	BSON	HTTP/REST	TCP	TCP/Thrift	HTTP/Rest ou TCP/Thrift
Armazenamento	Árvores binárias em memória / Em Disco	Em disco	Em memória e disco	Memtable / SSTable	HDFS
Pesquisa	Sim	Sim	Não	Sim	Sim
MapReduce	Sim	Não	Não	Sim	Sim

Tabela 5 - Resumo de algumas das principais soluções NoSQL [29,30,31]

Alguns exemplos de utilização de soluções referidas anteriormente são o uso de Cassandra no Facebook, bem como do HBase para o armazenamento e gestão das mensagens e o uso de MongoDB na CraigsList, uma página internacional de classificados.

Segundo o artigo "The CIO's Guide to NoSQL" [26] existe a previsão de que a maioria das maiores empresas mundiais vão ser forçadas a incorporar projetos NoSQL de modo a manter a competitividade, o que poderá criar alguma disparidade entre este tipo de empresas e outras empresas que não estejam dispostas a usar este tipo de tecnologias. O tamanho da *web* e a quantidade de recursos continuará a crescer e as tecnologias de *Web Semântica* tornarão as páginas cada vez mais fáceis de interpretar por máquinas. As entidades e informações presentes nestas serão cada vez mais guardadas e extraídas utilizando RDF e outros formatos.

A disputa entre SQL e NoSQL continuará, o que só poderá beneficiar ambos os lados. A necessidade de utilizar soluções SQL não vai desaparecer, pelo que o ideal será promover uma cooperação entre estes dois tipos de soluções [26,29].

3. Arquitetura

É habitual haver uma representação da arquitetura de uma aplicação com diversas camadas. Essa distribuição permite a manutenção de uma divisão lógica dos componentes e das funcionalidades, menosprezando a localização física. Mesmo que a aplicação não tenha uma representação visual para o utilizador, é possível separar a sua arquitetura em camadas de componentes de *software*. Estas camadas permitem diferenciar os diferentes tipos de tarefas executadas pelos componentes, tornando mais fácil criar um projeto que promova a reutilização dos mesmos. Em cada camada é possível, ainda, criar subcamadas que executem diferentes tipos de tarefas.

Através da identificação dos tipos genéricos dos componentes utilizados na solução, torna-se possível construir uma mapeamento da aplicação que pode ser usado na construção do projeto. Ao dividir uma aplicação em camadas diferentes com tarefas diferentes torna-se possível maximizar a manutenção do código, otimizar o modo como a aplicação funciona num ambiente diferente e delinear a localização exata em que uma decisão de tecnologia ou ao nível do projeto deve estar.

Numa representação geral de uma arquitetura de aplicação, no nível mais alto e abstrato, a visualização lógica da arquitetura pode ser compreendida como um conjunto de componentes que cooperam entre si, agrupados em diferentes camadas [32]. Na Figura 11 está uma representação simples dessas camadas, bem como das ligações existentes entre as mesmas e os utilizadores, outras aplicações, fontes que promovem o acesso a dados como Bases de Dados ou *Web Services* e serviços externos ou remotos que são consumidos pela aplicação.

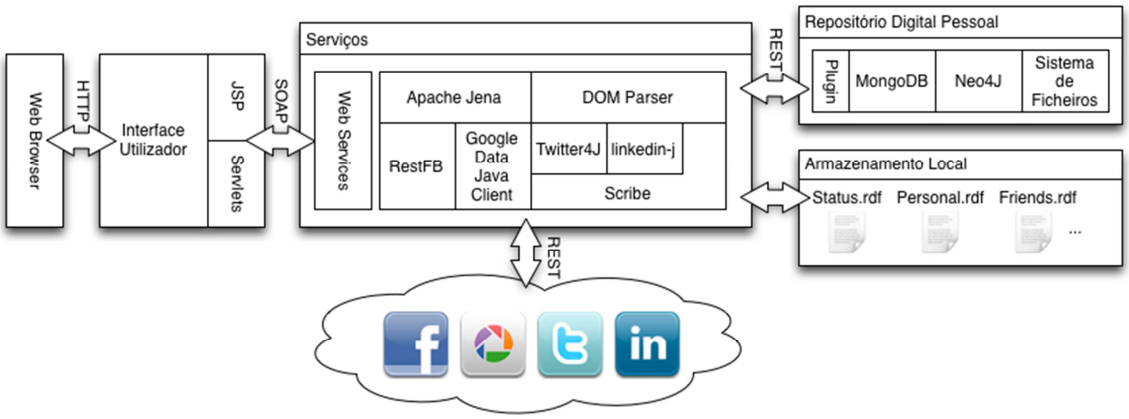

Figura 11 - Arquitetura da aplicação desenvolvida

Neste capítulo será feita uma análise mais detalhada à arquitetura da aplicação *web* desenvolvida, nomeadamente à Camada de Apresentação, de Negócio e de Dados.

Como referido anteriormente, foi usado o Java EE para desenvolver esta aplicação, pelo que será com base nesta plataforma que serão analisadas as diferentes camadas estruturais.

Os casos de uso disponíveis para um utilizador da página *web* desenvolvida estão presentes no Anexo A.

3.1. Camada de Apresentação

Numa fase inicial, para definir as especificações e os conteúdos necessários da página, optou-se por seguir uma proposta para a escrita de especificações de uma página *web* criada por Helen M. Horton [33].

Segundo esta proposta, uma página *web* será mais fácil de desenvolver se houver uma resposta pronta para uma série de perguntas. Essas perguntas são:

- Quais são os objetivos mensuráveis para a página;
- Qual é o público-alvo;
- Quais são as limitações técnicas;
- Qual é o conteúdo desejado;
- Quais são as características e as suas prioridades;
- Qual é a estrutura da página;
- Qual é o calendário proposto para estar completa;
- Quem vai gerir a manutenção da página.

Será feita, de seguida, uma análise por subtópicos a estas questões.

3.1.1. Objetivos

Como já foi analisado anteriormente, o desenvolvimento deste trabalho prevê que um conjunto de objetivos seja completado. Esses objetivos podem ser resumidos na seguinte lista:

- Permitir o acesso à informação partilhada numa determinada rede social utilizada pela pessoa;
- Garantir que o conjunto de redes sociais contém aquelas que são as mais populares e/ou mais utilizadas;

- Permitir guardar não só os conteúdos textuais mas também as imagens e os vídeos, quando tal for possível;
- Garantir a possibilidade de os dados serem guardados não só no disco rígido local mas também em serviços de alojamento *online*;
- Utilizar os procedimentos de autenticação e autorização disponibilizados pelas redes sociais usadas, bem como pelos serviços de alojamento, de modo a evitar a necessidade de utilizar *logins* e/ou *passwords* que podem aumentar o risco no uso da aplicação desenvolvida;
- Garantir que os dados são salvaguardados num formato que possibilite o fácil acesso a toda a informação presente. Por tudo o que já foi descrito anteriormente, foi escolhido o RDF;
- Possibilitar a salvaguarda da informação em RDF numa Base de Dados NoSQL, presente no servidor do grupo sobre repositórios digitais pessoais [34,35];
- Dar a possibilidade ao utilizador de migrar a informação relativa a uma rede social para outra rede social, nos casos em que tal se aplique. Para isso o utilizador terá de escolher a rede social para onde será colocada a informação e, de seguida, a rede social de onde deverão vir os dados originais, sendo que esses dados terão de estar já guardados na Base de Dados NoSQL presente no servidor e/ou no disco rígido;
- Ter as funcionalidades de interação com as redes sociais presentes em *Web Services* de modo a que estes possam ser acedidos pela aplicação e, no caso de ser necessário alterar estes serviços, não ser necessário alterar a aplicação original, bem como possibilitar que estes possam ser usados em aplicações diferentes;
- Dar aos utilizadores a possibilidade de aceder a informação sobre a página *web*, bem como sobre o contexto em que esta se enquadra;
- Dar aos utilizadores a possibilidade de contactar o autor da aplicação para eventuais esclarecimentos ou outras questões pontuais.

3.1.2. Público-alvo

Esta página *web* tem como público-alvo qualquer utilizador de uma rede social, dado que, potencialmente, qualquer pessoa registada numa rede social tem interesse em, não só obter uma garantia de que os seus dados partilhados são assegurados, mas também que consegue ter acesso a um histórico da sua atividade, independentemente do que aconteça aos dados originais.

Serão sobretudo pessoas jovens e adultas. Em termos de faixa etária, os utilizadores desta aplicação estarão maioritariamente num conjunto de idades entre os 17 e os 65 anos [36].

Em termos de conhecimentos e experiência, são necessários alguns conhecimentos mínimos relativamente ao modo de atuar nas páginas *web* atuais dado que é necessário proceder ao *login* nas redes sociais desejadas, é necessário aceitar permissões para que a rede social possa ter acesso à informação e é necessário, no caso de se usar um serviço de alojamento *online*, ter uma conta registada no serviço escolhido.

3.1.3. Limitações Técnicas

Serão usadas bastantes imagens e elementos gráficos para que o *site* possa ser apelativo para o utilizador. Além disso é usada uma biblioteca de funcionalidades de JavaScript com o nome jQuery que permite, entre outras coisas, adicionar efeitos, transições e *dialogs* a objetos como menus, botões e ligações.

Como já foi referido e justificado anteriormente, optou-se por usar o Java EE como plataforma de desenvolvimento. Ao mesmo tempo foi escolhido usar a tecnologia JSP em simultâneo com o desenvolvimento de Servlets para, não só mostrar o conteúdo das páginas ao utilizador, mas também permitir o desenvolvimento do código Java necessário para efetuar as tarefas necessárias.

Por último, para efetuar a interação com as redes sociais usou-se um *Web Service* para possibilitar, tal como já foi referido anteriormente, que eventuais alterações no modo de interagir com as redes sociais não obriguem a alterações na estrutura da aplicação *web,* bem como a utilização dos serviços desenvolvidos por uma aplicação diferente, caso seja necessário.

Em termos de tecnologias, o IDE escolhido para desenvolver a aplicação foi o NetBeans, e usam-se neste trabalho linguagens como o HTML, o JSP, o Cascading Style Sheets (CSS) que permite definir o aspeto e formatação de um documento escrito numa linguagem de *markup* como o HTML, o JavaScript e o Java.

3.1.4. Requisitos de conteúdo

Os requisitos, em termos de conteúdo, estão resumidos na Tabela 6.

Nome Conteúdo	Descrição	Tipo Conteúdo	Formato Conteúdo
Global (Aplicável a todas as páginas)			
Título da página	Assunto da página	Texto	HTML
Menu	Menu superior para navegação	Imagens/Texto	.png/HTML/ JavaScript
Fundo	Imagem de fundo usado nas páginas	Imagem	.gif
Página Inicial			
Introdução	Introdução à página *web* da aplicação	Texto	HTML
Informação			
/info.jsp	Informação acerca deste projeto e do âmbito em que ele surge	Texto	HTML
Página Inicial relativa a uma rede social			
/redesocial.jsp	Explicação breve do processo de autenticação na rede social.	Texto	HTML
Página de Autenticação na rede social			
/RSAuth.jsp	Autenticação propriamente dita da aplicação na rede social escolhida	Texto/Imagens	HTML/ JavaScript
Opções disponíveis dentro da rede social			
/RSOptions.jsp	Botões que mostram as diferentes maneiras de interagir com a rede social selecionada	Texto/Imagens	HTML/ JavaScript/ .jpg

Opção selecionada			
/RSDisco.jsp	Se for escolhido guardar os dados no disco é mostrado um texto informativo acerca do procedimento a seguir	Texto	HTML
/RSPicasa	Este Servlet mostra o procedimento a seguir para guardar no Google Picasa	Texto	HTML
/UploadToRS.jsp	Se a opção for de enviar informação para a rede social é mostrado o procedimento	Texto	HTML
Servlets para chamar os *Web Services*			
/RSDiscoWSCall	Este Servlet vai chamar o *Web Service* responsável por guardar a informação da rede social no disco e mostra o resultado	Texto/Imagem	HTML/.gif
/RSSavePicasa	Este Servlet vai chamar o *Web Service* responsável por guardar a informação da rede social no Google Picasa e mostra o resultado	Texto/Imagem	HTML/.gif
/UploadToRSWS Call	Se a opção for de enviar informação para a rede social é chamado o WS responsável e mostrado o resultado	Texto/Imagem	HTML/.gif
Nota: onde se encontra redesocial ou RS deve substituir-se pelo nome da rede social escolhida, por exemplo Facebook ou Twitter			

Tabela 6 - Resumo dos requisitos de conteúdo da página

Os requisitos apresentados são apenas uma versão resumida dos mesmos. De notar que as páginas respetivas às redes sociais e às opções selecionadas para guardar a informação são repetidas para as diferentes redes sociais e para as diferentes soluções existentes para completar essas opções.

3.1.5. Características e prioridades

Para esta temática será, mais uma vez, usada uma tabela para facilitar a compreensão da informação presente na página.

Esta subsecção pretende analisar as características que foram definidas como fundamentais e aquelas que podem ou não estar presentes.

Na Tabela 7 estão as características identificadas para este trabalho.

Característica	Prioridade
- Página inicial; - Menu superior com todas as opções em todas as páginas; - Informação em cada página; - Páginas de autenticação nas redes sociais; - Página com as opções para guardar a informação; - Página de autenticação no serviço de alojamento, caso este seja escolhido; - *Web Services* perfeitamente funcionais capazes de interagir com a rede social selecionada; - Página de chamada ao *Web Service* correspondente.	Fundamental
- Páginas iniciais das redes sociais; - Página inicial da opção selecionada; - Página atualizada com resultados do *Web Service*; - Página com informação relativa ao trabalho; - Permitir ao utilizador contactar o autor do trabalho.	Deve ter
- Aparecimento de um *dialog* para a seleção da localização para guardar os ficheiros quando se escolhe a opção "Guardar no Disco" com validação da entrada; - Animações no uso do menu superior; - Animações na escolha dos botões disponíveis para as opções de guardar os dados; - Aparecimento de uma imagem animada que indique que a ação está a decorrer quando a aplicação está a interagir com a rede social, de modo a dar ao utilizador *feedback* do que está a acontecer.	Pode ter

Tabela 7 - Lista de características e prioridades das mesmas

Foi esta a lista definida no início do processo de desenvolvimento da aplicação. Mais características podiam ser adicionadas, mas, no geral, esta lista consegue resumir bem os principais aspetos necessários.

3.1.6. Diagrama da página

É mostrado agora um diagrama representativo da página desenvolvida.

Como há diferenças significativas relativamente aos métodos de autenticação e na interação com os serviços disponíveis, mas, no geral, a estrutura das páginas *web* utilizadas nesse processo são similares, é apresentada na Figura 12 uma representação possível de um diagrama geral para este trabalho, com a estrutura das páginas necessárias para completar com sucesso os objetivos referidos anteriormente e que foi obtido antes de ser feito o desenvolvimento em si com base nos pontos definidos nas subsecções anteriores.

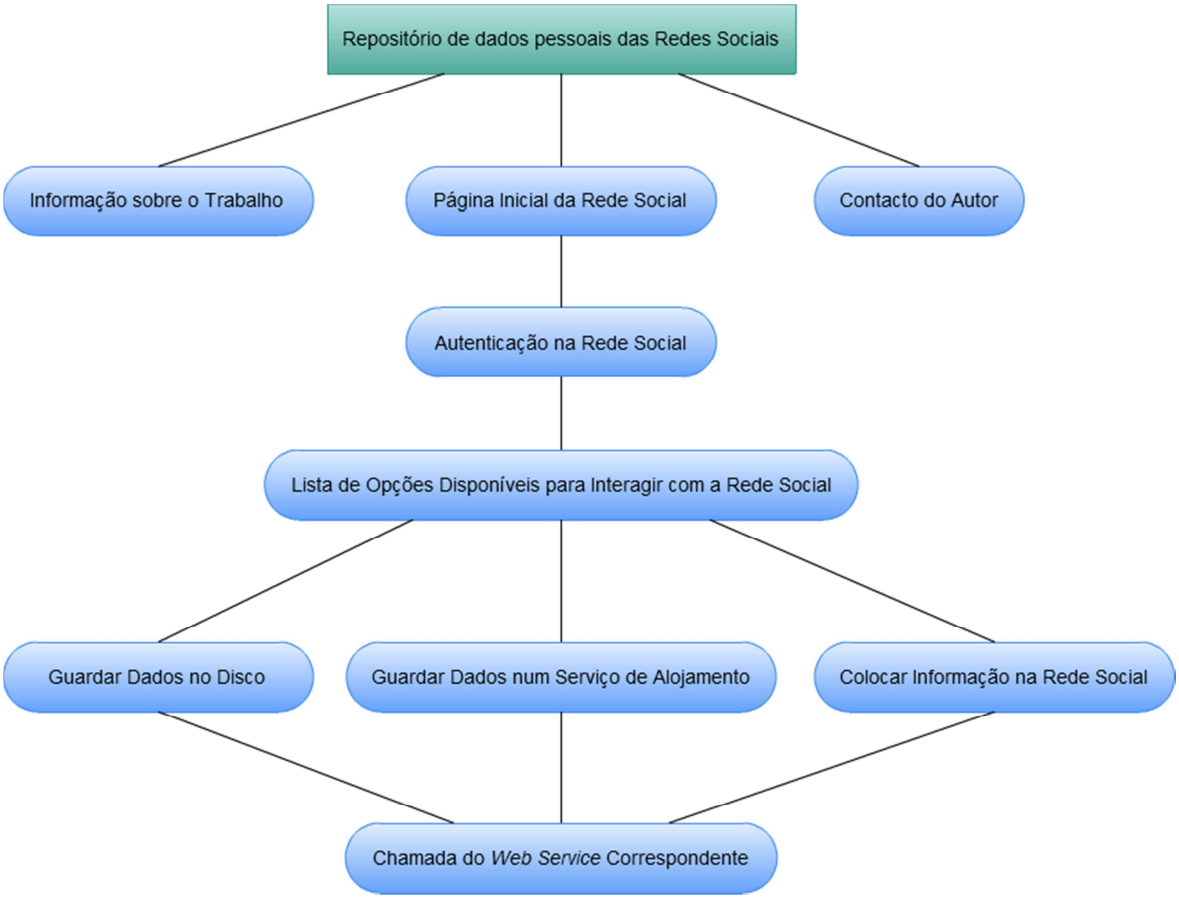

Figura 12 - Diagrama geral da página

É possível perceber, pelo diagrama, que as redes sociais presentes na página têm um comportamento similar. Após escolher qualquer rede social, o utilizador é direcionado para a página que faz uma explicação ao processo que se vai desenrolar e trata de o autenticar na rede social selecionada, num processo que varia consoante a rede social, de seguida é-lhe apresentada a lista de opções disponíveis. Consoante a opção selecionada o utilizador é encaminhado para uma página que resume o processo que se vai desenrolar. No caso de escolher o disco tem de indicar uma localização e, de seguida, é feita a chamada ao *Web Service* correspondente. No caso de selecionar um dos serviços de alojamento disponíveis é necessário efetuar mais uma autenticação, neste caso no serviço selecionado, para, de seguida, a informação ser guardada no serviço através de um *Web Service*. Caso seja selecionada a opção de efetuar o *upload* da informação para uma rede social, é mostrada uma página com a explicação do processo e, de seguida, é invocado o *Web Service* correspondente.

Estão ainda presentes neste diagrama as páginas para a obtenção de informação relativa ao trabalho e ao âmbito do mesmo e de contacto do autor deste trabalho, como tinha sido opcionalmente referido na recolha de requisitos.

3.1.7. Usabilidade

Nesta subsecção vai analisar-se a página relativamente a um termo muito em voga nos tempos atuais, a usabilidade.

Em primeiro lugar, deve definir-se usabilidade.

A usabilidade, resumidamente, é um atributo de qualidade que avalia quão fácil é usar uma interface de utilizador. A palavra refere-se também aos métodos existentes para melhorar a facilidade de utilizar os recursos disponibilizados durante a fase de projeto.

A usabilidade é, geralmente, definida por 5 componentes de qualidade:

- **Aprendizagem**: Quão fácil é, para os utilizadores, efetuar tarefas básicas na primeira vez que encontram o projeto?
- **Eficiência**: Depois de os utilizadores aprenderem a utilizar o projeto, com que rapidez conseguem efetuar as tarefas?
- **Memória**: Quando os utilizadores regressam ao projeto depois de um período em que não o usaram, com que facilidade conseguem recuperar as competências anteriores?

- **Erros**: Quantos erros são cometidos pelos utilizadores, quão graves são esses erros e com que facilidade é possível recuperar dos erros?
- **Satisfação**: Quão apelativo é usar o projeto?

Existem ainda outros atributos de qualidade importantes, nomeadamente a utilidade que se refere à funcionalidade do projeto, garantindo que o mesmo faz o que os utilizadores necessitam.

Na *Web* a usabilidade é uma condição fundamental. Se uma página é difícil de usar, se a página inicial não indica claramente o que é oferecido e o que os utilizadores podem fazer na mesma, se os utilizadores se perdem nas páginas existentes, se a informação de uma página for difícil de ler ou não responde às questões dos utilizadores, os utilizadores simplesmente deixam de usar a página.

A primeira lei do comércio eletrónico é "se os utilizadores não conseguem encontrar o produto, não o podem comprar", o que diz muito acerca da necessidade da usabilidade para resolver estes eventuais problemas.

Atualmente, as melhores práticas sugerem que se use cerca de 10% do orçamento disponível para um projeto em usabilidade. Em média esta percentagem tem um impacto pequeno, embora substancial em produtos de *software*, mas em termos de páginas *web* duplica as métricas de qualidade das mesmas [37].

3.1.8. Heurísticas

Jakob Nielsen, consultor de usabilidade *web*, definiu uma lista de 10 heurísticas de usabilidade que devem ser seguidas por uma aplicação para proporcionar ao utilizador uma agradável interação.

Estas heurísticas não são mais do que princípios gerais para o projeto de interfaces. Será feita uma análise a cada uma das heurísticas, tendo em conta também o que foi feito neste projeto para seguir essas regras:

- **Visibilidade do estado do sistema:** O sistema deve manter sempre o utilizador informado do que se está a passar através de *feedback* apropriado no tempo adequado. No projeto tentou-se seguir esta heurística. Em primeiro lugar, sempre que é selecionada uma opção que necessite de algum conhecimento, é mostrada a informação do que vai acontecer e do que o utilizador necessita de efetuar para conseguir interagir com sucesso. É também dado *feedback* ao utilizador, por exemplo, quando interage com uma

rede social já que, enquanto essa interação é efetuada, é mostrada uma imagem animada demonstrando que a informação está a ser processada;

- **Correspondência entre o sistema e o mundo real:** O sistema deve falar a mesma língua que o utilizador, com palavras, frases e conceitos familiares, em vez de usar termos orientados ao sistema. Devem seguir-se convenções do mundo-real, fazendo com que a informação apareça numa ordem natural e lógica. Na aplicação desenvolvida foi criado um menu superior que está disponível em todas as páginas do projeto e que permite um fácil acesso por parte do utilizador. Neste menu foi usada uma ordem tradicional, em que a imagem relativa à pagina inicial aparece em primeiro lugar e a informação da página e contacto em último e foram usadas imagens que são de fácil compreensão por parte do utilizador, mas também o nome de cada imagem, sempre que o rato passa pela mesma, de modo a evitar potenciais confusões. O mesmo acontece com os botões que permitem a interação com as redes sociais selecionadas no menu superior, que, além da imagem identificativa dos serviços usados, têm também texto que define os mesmos. Da mesma forma foi usada uma linguagem bastante fácil de compreender, sem termos técnicos, para que a página seja acessível a um grande número de utilizadores;
- **Controlo e liberdade do utilizador:** Os utilizadores muitas vezes escolhem funcionalidades do sistema por equívoco e vão necessitar de uma "saída de emergência" claramente marcada para sair do estado indesejado sem ser necessário um *dialog* extenso. As aplicações devem suportar *undo* e *redo*. Na aplicação desenvolvida, como já foi referido anteriormente, existe um menu superior que está presente em todas as páginas. A qualquer momento, mesmo em caso de um engano por parte do utilizador, a navegação pode ser passada para qualquer outra página através da simples escolha da opção do menu;
- **Consistência e *standards*:** Os utilizadores não devem ter de se preocupar se diferentes palavras, situações ou ações significam a mesma coisa. Devem seguir-se convenções da plataforma escolhida. Esta heurística está presente, mais uma vez através do menu superior e dos botões de interação com as redes sociais. No menu superior optou-se por usar uma imagem bem identificativa da página inicial, com a representação tradicional de uma casa, bem como da letra i para a informação e de uma carta para o contacto. Para

as redes sociais, foi usado o logotipo oficial de cada uma delas, evitando assim potenciais confusões por parte do utilizador. Como já foi referido anteriormente, em todas estas imagens está também um texto associado que permite, em caso de dúvida, perceber o que é que cada uma representa. No caso dos botões para interação com a rede social foi usada a mesma abordagem com a representação dos serviços com os logotipos oficiais e texto que define os mesmos;

- **Prevenção de erro:** Ainda melhor do que uma boa mensagem de erro é um projeto cuidadoso que previne o aparecimento de problemas. Deve eliminar-se uma potencial condição de erro ou verificar a presença dela com uma verificação, dando ao utilizador uma opção de confirmação antes de proceder à ação. No caso da aplicação desenvolvida esta heurística é seguida, por exemplo, quando se escolhe guardar a informação no disco, em que é pedido o local onde a informação deve ser guardada e é feita uma verificação, através de expressões regulares, para garantir que a localização introduzida é correta. Da mesma forma, sempre que vai haver um processo de autenticação e autorização, o utilizador é confrontado com uma mensagem que introduz esse processo e só depois de confirmar que quer avançar com o processo é que este é executado. O mesmo acontece com as chamadas aos *Web Services* que vão proceder às interações com as redes sociais propriamente ditas;
- **Reconhecimento ao invés de recordar:** Deve ser minimizada a necessidade de recorrer à memória por parte do utilizador tornando objetos, ações e opções visíveis. O utilizador não deve ter de se lembrar de informação de uma página para outra. As instruções para o uso do sistema devem ser visíveis ou facilmente obtidas sempre que for apropriado. Mais uma vez, na aplicação desenvolvida, esta heurística é seguida com base no menu superior, nos botões para interação com os diferentes serviços e com o aparecimento das instruções antes de ser efetuada qualquer interação. Todas as opções disponíveis ao utilizador estão presentes no menu superior, que está presente em qualquer página, pelo que não existe a necessidade de usar a memória para saber o que fazer. Da mesma forma, sempre que for escolhida uma rede social, as diferentes opções disponíveis aparecem, evitando que o utilizador tenha de recordar quaisquer aspetos. As instruções necessárias para a interação, por parte do utilizador, são mostradas imediatamente antes de

qualquer ação ser executada, dando ao utilizador a possibilidade de avançar ou de eventualmente escolher outra opção;

- **Flexibilidade e eficiência de utilização:** Aceleradores, que não são visíveis por utilizadores menos experientes, podem, muitas vezes, aumentar a velocidade da interação de um utilizador experiente de modo a que o sistema consiga abranger utilizadores com diferentes níveis de experiência. Deve garantir-se que o sistema está construído para estes dois tipos de utilizadores. Nesta aplicação, um utilizador com alguma experiência, que já tenha uma noção das ações necessárias para que a página efetue o que deseja, consegue ter uma sucessão de interações bastante rápida e que não lhe causa o constrangimento de ter de esperar que a página acompanhe o seu ritmo. Isto é claro, por exemplo, no caso da interação com o Facebook em que, caso o utilizador já tenha dado acesso à aplicação, não é necessária qualquer ação para garantir a autenticação e autorização, passando, desse modo, imediatamente para o passo seguinte que mostra as opções disponíveis para a interação. Do mesmo modo, no caso de um utilizador com menos experiência, a sua interação não é, de modo algum, afetada por este facto, garantindo que todos os passos estão disponíveis para explorar;
- **Estética e design minimalista:** Os *dialogs* não devem ter informação irrelevante ou raramente necessária. Cada unidade extra de informação num *dialog* compete com as unidades de informação relevantes e faz com que a sua visibilidade relativa seja diminuída. Na aplicação desenvolvida foi seguida esta heurística. Tentou ter-se um *design* apelativo, mas ao mesmo tempo simples. O menu superior tentou resumir nos botões existentes toda a informação necessária para que o utilizador compreenda as escolhas a fazer, tal como acontece nos botões para interação com os serviços disponibilizados. Da mesma forma, apesar de terem sido adicionados alguns efeitos nos menus disponibilizados, tentou-se que, no essencial, fosse mantida a simplicidade das ações;
- **Ajudar os utilizadores a reconhecer, diagnosticar e recuperar de erros:** As mensagens de erro devem ser expressas em linguagem simples, evitando o uso de códigos, para indicar com precisão o problema e, construtivamente, sugerir uma solução. Neste projeto, cada vez que existe algum problema que impossibilite a obtenção dos resultados esperados, é mostrada uma mensagem de erro com um texto que explica o porquê do surgimento desse

erro. Seria mais fácil optar por apenas mostrar o tipo de erro, ou utilizar um código que o identificasse, mas isso tornaria o erro incompreensível por parte do utilizador;

- **Ajuda e documentação:** Apesar de ser melhor o sistema poder ser utilizado sem qualquer documentação, pode ser necessário providenciar ajuda e documentação. Qualquer informação deve ser fácil de procurar. Tendo em conta a tarefa que o utilizador está a efetuar, devem ser listados os passos concretos que devem ser seguidos, tendo em consideração que a lista não deverá ser demasiado longa. Como já foi referido anteriormente, neste trabalho há uma secção dedicada a informação acerca da aplicação. Além disso, tanto na página inicial como em qualquer página que necessite de interação com o utilizador, é mostrado texto que explica os passos a seguir e os resultados que devem ser verificados [38].

Esta lista de heurísticas, apesar de não poder ser a única fonte de informação para o desenvolvimento do projeto, permite ter a certeza de que o trabalho desenvolvido permitirá uma utilização eficiente por parte do utilizador.

3.1.9. Mockups

Como é visível na Figura 11 e como já foi referido, a Camada de Apresentação, no projeto desenvolvido, foi construída através da utilização de páginas JSP e *Servlets*, bem como de todos os componentes que constituem estas página, como por exemplo a utilização da biblioteca jQuery [39] para integrar animações e/ou melhorar as interações com o utilizador.

Uma estratégia importante no desenvolvimento de projetos é a utilização de *mockups*. Um *mockup* é um modelo do projeto desenvolvido que permite avaliar as ideias integradas no mesmo. Funciona, por exemplo, como uma maquete em projetos de arquitetura. Neste caso, um *mockup* funcionará como um protótipo do que se vai poder usar, providenciando, pelo menos, uma parte das funcionalidades que o sistema terá, permitindo ainda testar se o *design* idealizado é implementável e funcional.

Com base nas informações anteriores, nesta fase do projeto foram elaborados os *mockups* presentes na sequência de imagens que vai desde a Figura 13 à Figura 19.

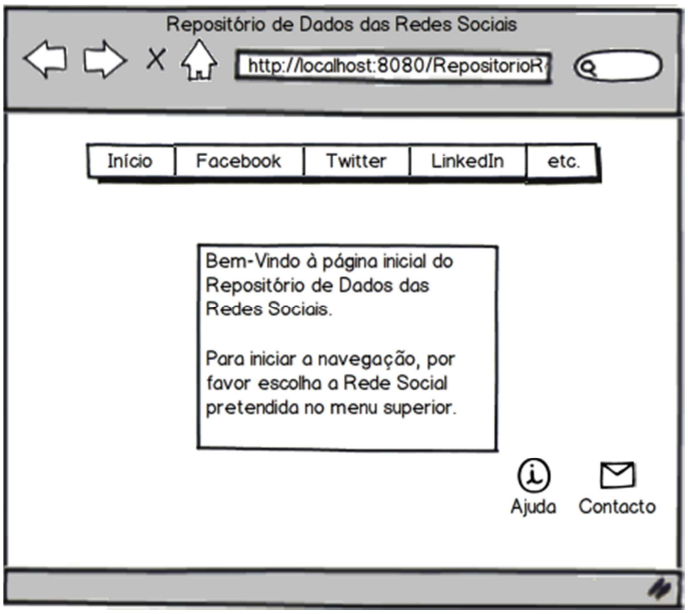

Figura 13 - Mockup da Página Inicial da Aplicação Web

Nesta página inicial, destaca-se a presença do menu superior, que estará presente em todas as páginas da aplicação. No centro estará um campo de texto que dará as boas-vindas ao utilizador, bem como uma breve ajuda para a navegação na aplicação, que poderá ser complementada carregando no botão Ajuda, que terá informação mais detalhada sobre a interação com a aplicação. No botão Contacto será possível enviar um e-mail para o autor da aplicação. Numa fase posterior optou-se por integrar os botões Ajuda e Contacto no menu superior, pelo facto da possibilidade de, assim, estarem sempre disponíveis para o utilizador selecionar, independentemente da localização atual.

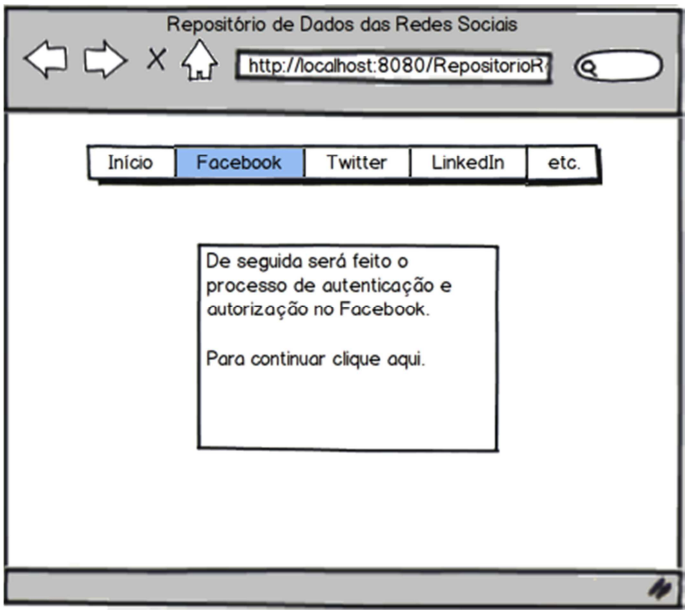

Figura 14 - Mockup da Página Inicial do Facebook

Depois de escolhido, por exemplo, o Facebook, é apresentada uma página que faz uma introdução ao processo de autenticação e autorização, sendo necessário clicar na ligação existente para continuar a navegação. Optou-se por esta abordagem para evitar que a pessoa navegasse diretamente para uma página do Facebook em que necessita de aceitar as permissões, sem qualquer introdução que permitisse perceber os passos necessários.

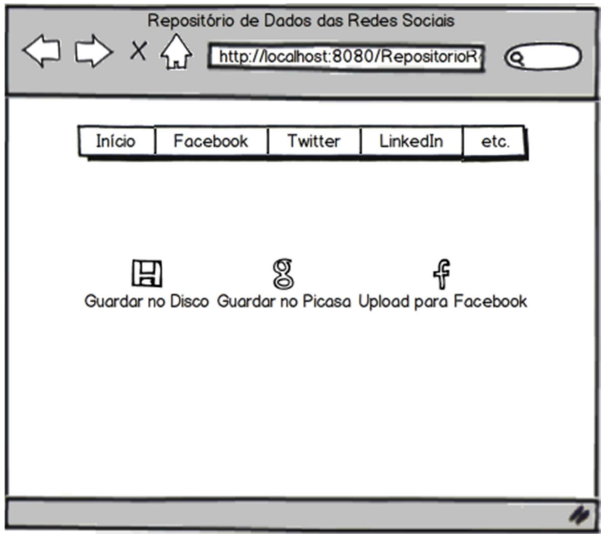

Figura 15 - Mockup das Opções Disponíveis no Facebook

Na fase seguinte haverá, então, a necessidade de aceitar as permissões da aplicação no Facebook. Depois dessa etapa, serão mostradas ao utilizador as diferentes opções disponíveis para guardar a informação, como mostrado na Figura 15.

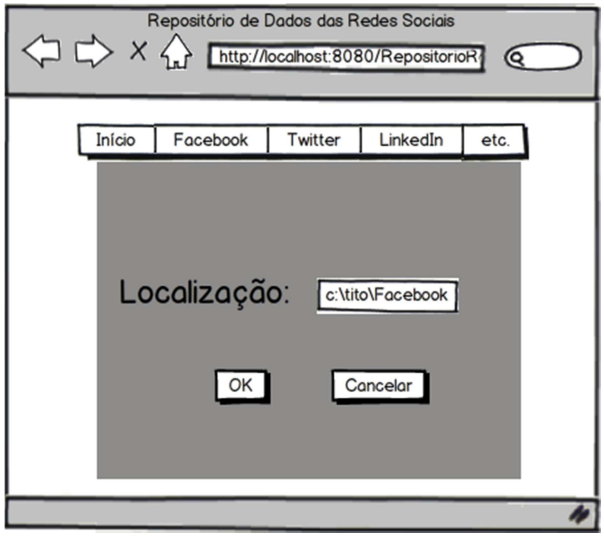

Figura 16 - Mockup da Inserção da Localização para Guardar a Informação

Caso seja escolhida a opção de guardar no disco, deverá ser mostrada a informação presente na Figura 16. Terá de ser feita uma verificação ao texto inserido, para garantir que essa informação constitui uma localização válida no Sistema Operativo.

A ideia inicial seria colocar uma janela no navegador de pastas existente no Sistema Operativo utilizado, mas após alguma pesquisa, verificou-se que as opções existentes permitem apenas a escolha de ficheiros, e não de pastas. Como neste projeto é necessário indicar a localização de uma pasta, essa abordagem não tornava possível o sucesso deste trabalho, optando-se por inserir a localização por via textual, mas com verificações, através de expressões regulares, da informação inserida, de modo a garantir que a expressão inserida constitui um caminho para uma pasta válido.

No caso da seleção de um dos serviços disponíveis para o alojamento *online*, seria necessário passar por mais uma fase de autenticação e autorização antes de poder enviar os ficheiros para o serviço.

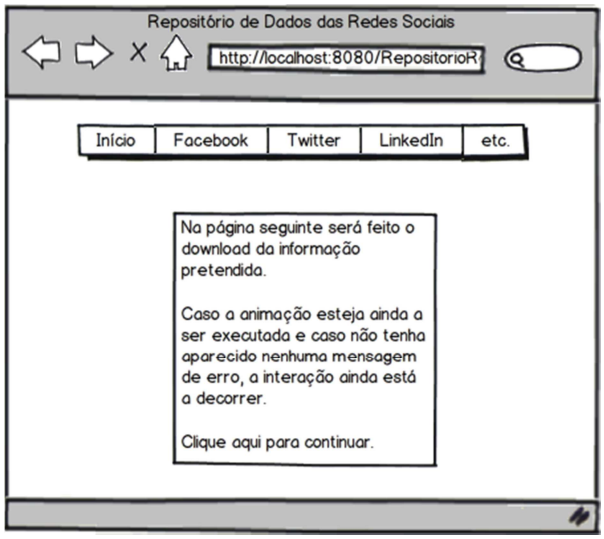

Figura 17 - Mockup da Confirmação da Interação com a Rede Social

Na página seguinte será, mais uma vez, mostrado algum texto que explica o processo que vai decorrer. Optou-se por apresentar esta informação porque o processo de interação seguinte, dependendo da quantidade e tamanho da informação disponibilizada na rede social, pode ser demorado, tentando deste modo evitar que a pessoa cancele o processo por achar que está demorar demasiado tempo.

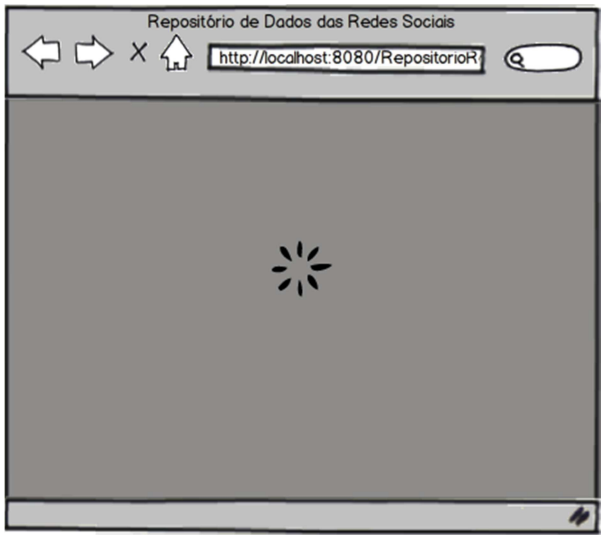

Figura 18 - Mockup da Animação Relativa ao Processamento da Informação

Nesta fase, enquanto decorrer a interação com a rede social escolhida, será mostrada a animação relativa a esse processo, como está representado na Figura 18.

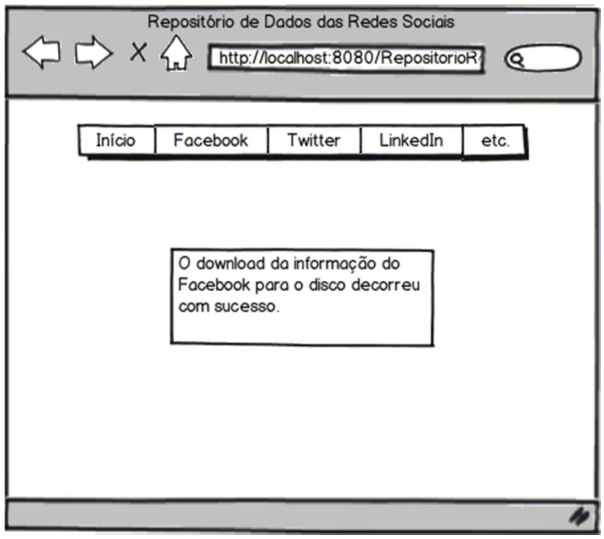

Figura 19 - Mockup do Sucesso da Interação

Caso o processo seja bem-sucedido, será mostrada essa informação na página, e os dados serão guardados, neste caso, na localização selecionada no disco. Na eventualidade de ocorrer algum erro no processo, será mostrada esta mesma janela, mas com a informação do erro ocorrido.

3.2. Camada de Negócio

Esta camada implementa as funcionalidades principais do sistema e encapsula a lógica de negócio relevante. Geralmente é constituída por componentes que podem expor interfaces de serviço que podem ser acedidas. No caso prático da aplicação desenvolvida, estas funcionalidades foram conseguidas através da utilização de *Web Services* que promovem as interações com as redes sociais, bem como com a Base de Dados NoSQL presente no projeto do repositório digital pessoal.

Nesta camada estão presentes todos os mecanismos de interação existentes na aplicação que fazem um trabalho invisível ao utilizador, mas que serão cruciais na obtenção dos resultados pretendidos. Dela fazem parte os mecanismos de autenticação e autorização que serão descritos no capítulo 3.4, mas também os diferentes *Web Services* criados para interagir com as redes sociais. Estes serviços serão descritos na subsecção seguinte.

3.2.1. *Web Services* usados na aplicação

Os *Web Services* tradicionais são, habitualmente, definidos por fazerem parte das aplicações desenvolvidas, usarem protocolos abertos, funcionarem independentemente da aplicação desenvolvida para interagir com os mesmos, poderem ser descobertos através de Universal Description Discovery and Integration (UDDI), um mecanismo de registo e localização de *Web Services* e poderem ser usados em mais do que uma aplicação, promovendo assim uma arquitetura baseada em SOA (Service-Oriented Architecture), possibilitando, assim, que os *Web Services* desenvolvidos possam ser usados em quaisquer aplicações. A base dos *Web Services* é o XML e a plataforma básica é uma conjunção entre o XML e o HTTP.

A escolha destes dois componentes justifica-se com o facto do XML ser uma linguagem que pode ser usada entre diferentes plataformas, mantendo a complexidade necessária para as tarefas desejadas e do HTTP ser o principal protocolo de Internet usado.

Os elementos principais de plataforma dos *Web Services* são:

- SOAP (Simple Object Access Protocol);
- UDDI;
- WSDL (Web Services Description Language).

Como já foi referido, usando *Web Services*, as aplicações podem publicar informação para o resto do mundo, usando XML para codificar e descodificar os dados e SOAP para os transportar.

Segundo o w3schools [40], Os *Web Services* têm dois tipos de uso fundamentais:

- **Permitir a reutilização de componentes:** Se existem funcionalidades de que uma aplicação necessita constantemente, torna-se desnecessário ter de estar sempre a implementá-las. Através dos *Web Services* torna-se possível oferecer este tipo de componentes como serviços, estando acessíveis a qualquer momento para qualquer consumidor;
- **Ligar com o *software* já existente:** Os *Web Services* podem ajudar a resolver os problemas de interoperabilidade dando às diferentes aplicações maneiras de ligar os seus dados. Torna-se possível, então, trocar dados entre diferentes aplicações em diferentes plataformas.

Existem três elementos de plataforma básicos que fazem parte dos *Web Services* e que já foram descritos anteriormente.

Começando pelo SOAP, é um protocolo baseado em XML que permite a troca de informação entre aplicações sobre HTTP ou, de forma mais simples, é um protocolo para aceder a *Web Services*. O SOAP tem as seguintes características [40]:

- É um protocolo de comunicação;
- É um formato para envio de mensagens;
- Foi projetado para comunicar via Internet;
- É independente da plataforma;
- É independente da linguagem de programação;
- É baseado em XML;
- É simples e extensível;
- Permite ultrapassar *firewalls*;
- É um *standard* do W3C.

Outro dos elementos referidos é o WSDL, que é uma linguagem baseada em XML para localizar e descrever *Web Services*. O WSDL caracteriza-se por [40]:

- Ser baseado em XML;
- Ser usado para descrever *Web Services*;
- Ser usado para localizar *Web Services*;
- Ser um *standard* do W3C.

Por último, existe também o UDDI que, como já foi referido anteriormente, é um serviço de diretórios onde as empresas podem registar e procurar por *Web Services*. Tem as seguintes características [40]:

- É um diretório para guardar informação sobre *Web Services*;
- É um diretório de interfaces de *Web Services* descritas por WSDL;
- Comunica via SOAP;
- Está incluída na plataforma Microsoft .NET, que não é a escolha deste trabalho, pelo que esta característica não é importante no âmbito deste projeto.

Optou-se por usar uma abordagem baseada em Java API for XML Web Services (JAX-WS) que é um conjunto de tecnologias Java usadas para desenvolver este tipo de serviços. Na Figura 20 é mostrado um esquema representativo da comunicação existente no JAX-WS.

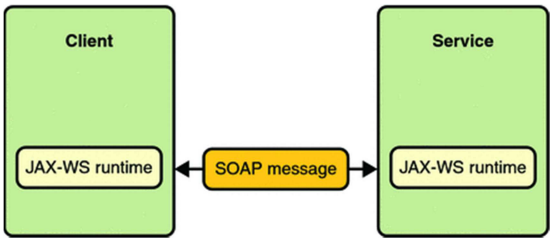

Figura 20 - Comunicação em JAX-WS [41]

Esta tecnologia é fundamental para desenvolver *Web Services* Java baseados em XML, bem como os clientes correspondentes que utilizem esta linguagem para troca de mensagens ou utilizem Remote Procedure Call (RPC), um tipo de comunicação que permite que uma aplicação seja executada num espaço de endereçamento diferente.

O JAX-WS representa as mensagens ou RPCs utilizando protocolos baseados em XML como o SOAP, mas esconde a complexidade inerente a esta tecnologia através de uma API baseada em Java. Os programadores geralmente utilizam esta API para definir os métodos necessários e, de seguida, tratam de criar as diferentes classes que implementam esses métodos, deixando os detalhes de comunicação sob responsabilidade da API do JAX-WS. O JAX-WS converte, em tempo real, chamadas da API e as respetivas respostas para mensagens SOAP [42].

Para proceder às necessárias interações com as redes sociais e, eventualmente, com os serviços de alojamento selecionados, optou-se por integrar estas funcionalidades num conjunto de *Web Services*. Deste modo, facilita-se a experiência do utilizador, já que necessita apenas de esperar pelo processamento da informação uma vez, não havendo um conjunto de interações ao longo das diferentes páginas, bem como o controlo das exceções e eventuais problemas, já que se torna possível efetuar este controlo de forma centralizada.

Dentro dos diferentes *Web Services*, específicos a cada rede social utilizada, é criado, inicialmente, o cliente que vai possibilitar a interação com a rede social, usando para isso o *token* de acesso obtido durante o processo de autenticação e autorização. Caso, por qualquer motivo, o *token* seja inválido, o utilizador é confrontado com uma mensagem de erro e a interação não pode ser efetuada. De seguida são criados os ficheiros que vão guardar os conteúdos partilhados pelo utilizador em formato RDF. Caso o utilizador tenha selecionado a opção de guardar a informação no disco, estes ficheiros serão criados na pasta selecionada. Caso tenha escolhido qualquer outra opção, estes ficheiros serão guardados numa pasta temporária, sendo, depois do envio para a Base de Dados NoSQL ou para a rede social selecionada, eliminados.

Depois de completados os passos supracitados, serão executados os métodos correspondentes ao tipo de interação desejada. Estes métodos serão descritos de seguida, separados por rede social.

3.2.1.1. *Web Service* do Facebook

O *Web Service* do Facebook é composto por dois métodos principais, que estão representados na Figura 21.

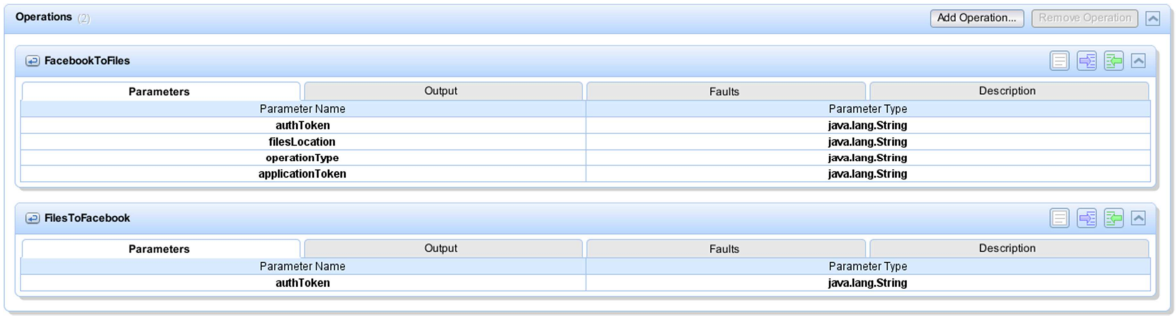

Figura 21 - Métodos que fazem parte do *Web Service* do Facebook

Nas Figuras 22 e 23 encontram-se exemplos do código XML necessário para invocar estes métodos.

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns0:FacebookToFiles xmlns:ns0="http://facebook.tese.ua.pt/">
            <authToken>sample text</authToken>
            <filesLocation>sample text</filesLocation>
            <operationType>sample text</operationType>
            <applicationToken>sample text</applicationToken>
        </ns0:FacebookToFiles>
    </soap:Body>
</soap:Envelope>
```

Figura 22 - Código XML do método FacebookToFiles

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns0:FilesToFacebook xmlns:ns0="http://facebook.tese.ua.pt/">
            <authToken>sample text</authToken>
        </ns0:FilesToFacebook>
    </soap:Body>
</soap:Envelope>
```

Figura 23 - Código XML do método FilesToFacebook

De seguida serão descritos os métodos auxiliares que fazem parte dos métodos referidos anteriormente, analisando-os individualmente.

FacebookToFiles

No caso do primeiro método, é devolvida uma String, ou seja, um conjunto de caracteres, que identifica o sucesso ou insucesso da operação. Este método recebe como parâmetros o *token* de acesso obtido no processo anteriormente descrito, a localização dos ficheiros, relevante no caso de se optar por guardar a informação no disco, o tipo de operação, que identifica se os ficheiros são guardados no disco ou num serviço de alojamento, identificando o mesmo, e o *token* da aplicação, caso a opção seja a de guardar a informação num serviço de alojamento.

No *Web Service* do Facebook, caso tenha sido selecionada uma operação que permita guardar a informação proveniente da rede social, depois de criado o cliente utilizando, para isso, o *token* de autenticação, são criadas as diversas ligações necessárias para proceder às diferentes interações. As ligações criadas são as seguintes:

- ***User*:** Permite obter a informação pessoal associada ao utilizador;
- ***Messages*:** Permite obter as mensagens recebidas e enviadas por parte do utilizador;
- ***Friends:*** Obtenção da lista de amigos do utilizador;
- ***Likes:*** Lista de *likes* feitos pelo utilizador.

Este método é constituído, ainda, pelos seguintes métodos auxiliares, usados neste método principal:

- **writePersonalDataToFile**(OutputStream os, User user): Recebe como parâmetros a *Stream* responsável por possibilitar a escrita da informação no ficheiro e o User descrito anteriormente. Será responsável por escrever a informação pessoal do utilizador;
- **writeStatusToFile**(FacebookClient facebookClient, OutputStream os, String fileLocation, String operationType, String applicationToken): Este método será responsável por escrever a informação partilhada pelo utilizador, além de guardar todas as imagens e vídeos presentes na página pessoal. De notar que, neste caso, foi necessário passar como parâmetro o próprio cliente de Facebook já que será necessário interagir com mais do que uma ligação dentro deste método;

- **writeLikesToFile**(OutputStream os, Connection<Post> likesList): Este método será responsável por escrever a informação relativa a todos os conteúdos em que o utilizador deixou um *like*. Desta vez foi possível passar apenas a ligação necessária, já que a interação é apenas feita com esta;
- **writePrivateMessagesToFile**(OutputStream os, FacebookClient facebookClient, Connection<Post> messagePosts): Este método tem a responsabilidade de escrever as mensagens privadas do utilizador. Durante a programação deste método surgiram alguns problemas, face às limitações impostas pelo Facebook relativamente à manipulação de mensagens privadas, e que serão alvo de uma análise com maior detalhe na parte final deste documento, dentro das conclusões retiradas com este trabalho, que obrigaram à utilização, não só da ligação correspondente às mensagens, mas também a um cliente específico, para obter a informação necessária.

FilesToFacebook

Este método será o responsável por colocar a informação proveniente dos ficheiros RDF no Facebook. Para isso recorre a uma pasta temporária onde guardará os ficheiros RDF depois de os obter da Base de Dados NoSQL, apagando-os no final da interação.

Utiliza o tipo de dados Document, que será descrito na subsecção 3.2.5, para efetuar a interpretação dos ficheiros RDF obtidos.

Este método contém dois outros métodos que efetuam as necessárias funcionalidades para colocar na rede social a informação obtida:

- **statusToFacebook**(Document doc, FacebookClient facebookClient): Este método trata de colocar a informação partilhada pelo utilizador na rede social original no Facebook. Para isso utiliza um cliente de Facebook, como também já foi visto nos casos anteriores, e um método de *publish* associado a este cliente, que permite a publicação de diferentes tipos de informação como fotografias, *posts* ou notas;
- **personalToFacebook**(Document doc, FacebookClient facebookClient): Neste método será colocada no Facebook a informação pessoal associada ao utilizador. Como o Facebook não disponibiliza uma maneira de interagir diretamente com esta informação, optou-se por colocar esta informação disponível através da criação de uma nota. O utilizador poderá, depois, introduzir manualmente esta informação, associando-a ao seu perfil.

3.2.1.2. *Web Service* do Twitter

No caso do Twitter, o *Web Service* é composto pelos dois métodos visíveis na Figura 24 e que serão descritos de seguida:

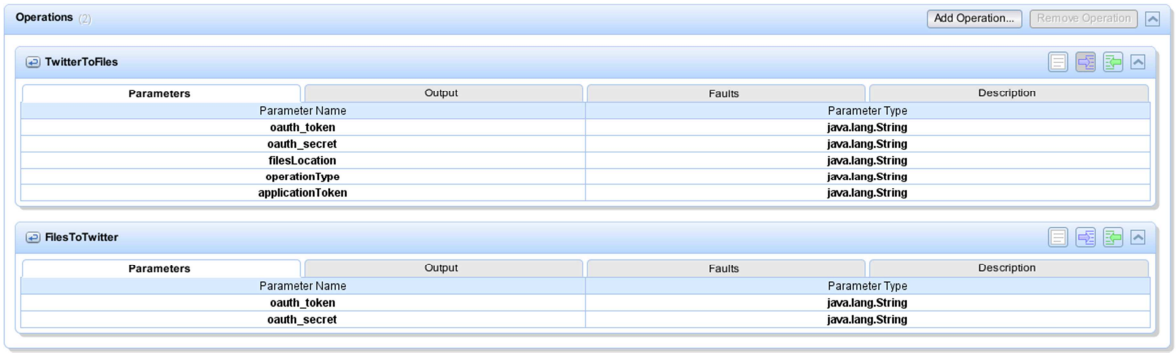

Figura 24 - Métodos do *Web Service* do Twitter

Na Figura 25 está um exemplo do código necessário para chamar o primeiro método deste *Web Service*, estando na Figura 26 um exemplo do segundo método.

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns0:TwitterToFiles xmlns:ns0="http://twitter.tese.ua.pt/">
            <oauth_token>sample text</oauth_token>
            <oauth_secret>sample text</oauth_secret>
            <filesLocation>sample text</filesLocation>
            <operationType>sample text</operationType>
            <applicationToken>sample text</applicationToken>
        </ns0:TwitterToFiles>
    </soap:Body>
</soap:Envelope>
```

Figura 25 - Mensagem XML para invocar o método TwitterToFiles

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns0:FilesToTwitter xmlns:ns0="http://twitter.tese.ua.pt/">
            <oauth_token>sample text</oauth_token>
            <oauth_secret>sample text</oauth_secret>
        </ns0:FilesToTwitter>
    </soap:Body>
</soap:Envelope>
```

Figura 26 - Exemplo de mensagem XML para o método FilesToTwitter

TwitterToFiles

Neste método, tal como acontece com o Facebook, é devolvida uma String com o resultado da interação. Neste caso, não é apenas possível passar como parâmetro um *token* de acesso, já que, como será explicado no processo de autenticação e autorização, especificamente no Anexo B, existe uma segunda chave, denominada de *secret*, que é fundamental para possibilitar a interação com a rede social.

Os restantes parâmetros têm um funcionamento semelhante ao que acontece no caso do Facebook.

Quando este *Web Service* é executado, o primeiro passo é criar um cliente que permita a interação com o Twitter através da utilização do *token* de acesso e do *secret*.

Este método do *Web Service* chama, de seguida, os seguintes métodos auxiliares:

- **writeStatusToFile**(OutputStream os, Twitter twitter, String filesLocation, String operationType, PicasawebService picasa): Este método possibilitará a escrita da informação partilhada pelo utilizador no ficheiro RDF correspondente. Tal como acontece com o Facebook, é passado como parâmetro um OutputStream para efetuar a escrita da informação. Neste caso, é passado um cliente para o Twitter através da biblioteca referida Twitter4J já referida anteriormente, bem como a localização dos ficheiros e o tipo de operação. Neste caso, a passagem do tipo de dados que permite a interação com o Google Picasa facilitou a tarefa de interagir com um serviço externo à aplicação, pelo que é um dos parâmetros passados;
- **writePersonalToFile**(OutputStream os, Twitter twitter, String filesLocation, String operationType, PicasawebService picasa): Neste método os argumentos são exatamente os mesmos do caso anterior, mas aqui haverá a responsabilidade de transportar para um ficheiro RDF a informação pessoal associada com a conta usada;
- **writeFollowersToFile**(OutputStream os, Twitter twitter): Neste caso, serão colocados num ficheiro todos os seguidores da conta de Twitter associada com o utilizador;
- **writeFollowingToFile**(OutputStream os, Twitter twitter): Este método é bastante semelhante ao método anteriormente descrito, mas neste caso envolverá as pessoas que o utilizador da conta de Twitter segue;

- **writeDirectMessagesToFile**(OutputStream os, Twitter twitter): Por último torna-se possível, através deste método, passar para o ficheiro RDF as mensagens privadas recebidas e enviadas pelo utilizador.

FilesToTwitter

Este método, tal como acontece com o Facebook, utiliza o tipo de dados Document para efetuar o *parsing* da informação presente nos ficheiros RDF e produzir as interações necessárias à colocação dessa informação no local correto.

Dele fazem parte os seguintes métodos auxiliares:

- **statusToTwitter**(Document doc, Twitter twitter): Este método tem como parâmetros o tipo de dados descrito anteriormente e o cliente de Twitter criado usando as chaves necessárias. Depois de ser extraída a informação relevante, esta será publicada no Twitter por ordem cronológica, havendo a possibilidade de as mensagens analisadas possuírem imagens, obrigando à colocação das mesmas;
- **personalToTwitter**(Document docPersonal, Twitter twitter): Neste caso, ao contrário do que acontece com o Facebook, existe a possibilidade de interagir diretamente com os dados pessoais do utilizador pelo que, este método, obtém a informação relevante do ficheiro RDF associado à informação pessoal do utilizador e coloca-a diretamente na conta de Twitter desejada. O mesmo acontece com a imagem definida como imagem de perfil que será automaticamente colocada na conta de Twitter;
- **followingToTwitter**(Document docFollowing, Twitter twitter): Este método trata de seguir todas as pessoas presentes na lista de pessoas que o utilizador seguia originalmente. No caso das pessoas que seguiam o utilizador, não é possível efetuar qualquer tipo de interação que as faça voltar a seguir, pelo que esse tipo de função terá de ser efetuada manualmente.

3.2.1.3. *Web Service* do LinkedIn

Para o LinkedIn, a abordagem seguida foi bastante semelhante à do Twitter. Neste caso, o *Web Service* é composto pelos métodos da Figura 27.

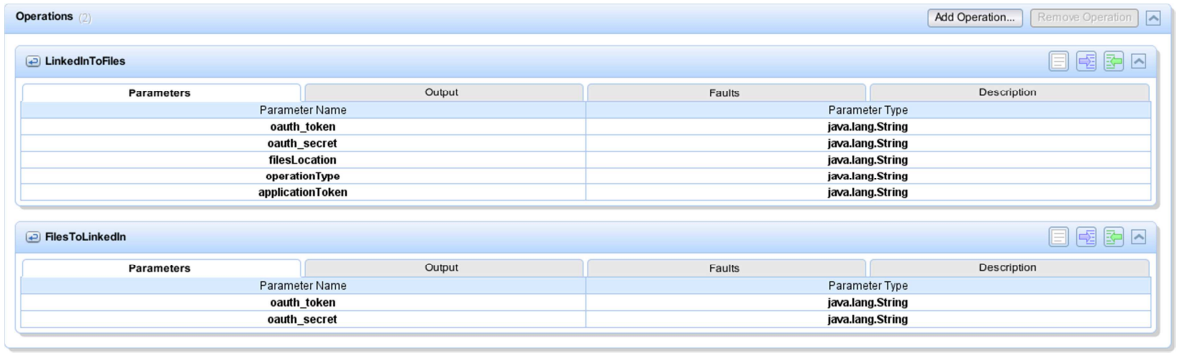

Figura 27 – *Web Service* do LinkedIn

Nas Figuras 28 e 29 é visível o código XML necessário para uma mensagem de *input* para os métodos LinkedInToFiles e FilesToLinkedIn, respetivamente.

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns0:LinkedInToFiles xmlns:ns0="http://linkedin.tese.ua.pt/">
            <oauth_token>sample text</oauth_token>
            <oauth_secret>sample text</oauth_secret>
            <filesLocation>sample text</filesLocation>
            <operationType>sample text</operationType>
            <applicationToken>sample text</applicationToken>
        </ns0:LinkedInToFiles>
    </soap:Body>
</soap:Envelope>
```

Figura 28 – Código XML do método LinkedInToFiles

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns0:FilesToLinkedIn xmlns:ns0="http://linkedin.tese.ua.pt/">
            <oauth_token>sample text</oauth_token>
            <oauth_secret>sample text</oauth_secret>
        </ns0:FilesToLinkedIn>
    </soap:Body>
</soap:Envelope>
```

Figura 29 – Código XML do método FilesToLinkedIn

LinkedInToFiles

Tal como nos métodos anteriores, é devolvida uma String com o resultado da interação. A abordagem para este método é bastante semelhante à seguida no caso do Twitter, havendo também a utilização de *token* e *secret*.

Como no caso anterior, o primeiro passo é criar um cliente para interagir com o LinkedIn através do *token* e do *secret*, havendo, de seguida, chamadas aos seguintes métodos auxiliares:

- **writeStatusToFile**(OutputStream os, LinkedInApiClient client): No caso do LinkedIn, as mensagens partilhadas não são tão importantes como nas redes sociais analisadas anteriormente. Dado que esta rede social se foca mais num ambiente profissional, não é tão relevante haver partilha de conteúdos entre utilizadores. Ainda assim este método guarda a informação de todas as partilhas efetuadas pelo utilizador;
- **writePersonalToFile**(OutputStream os, LinkedInApiClient client, String filesLocation, String operationType, String applicationToken): Este método permitirá escrever para o documento RDF a informação pessoal presente na conta de LinkedIn escolhida. No caso do LinkedIn, esta informação será crucial dado que terá todos os dados relativos à escolaridade da pessoa, a eventuais empregos que a pessoa tenha tido, aos conhecimentos da pessoa, entre outros aspetos que são muito importantes nesta rede social; **writeConnectionsToFile**(OutputStream os, LinkedInApiClient): Este método vai guardar a informação de todas as ligações que o utilizador tem, nomeadamente o nome e o ID que identifica inequivocamente cada utilizador.

FilesToLinkedIn

Tal como nos métodos anteriores, utiliza o tipo de dados Document para efetuar o *parsing* da informação presente nos ficheiros RDF e colocar a informação no LinkedIn.

São usados os seguintes métodos auxiliares:

- **statusToLinkedIn**(Document docStatus, LinkedInApiclient client): Este método vai colocar todas as mensagens partilhadas pelo utilizador na nova conta de LinkedIn. Para a nova mensagem é colocada a informação da mensagem original, nomeadamente do autor da mesma e da data de criação da mensagem;
- **personalToLinkedIn**(Document docPersonal, LinkedInApiclient client): Como o LinkedIn não suporta alterações diretas à informação pessoal do utilizador, mas permite o envio de mensagens privadas, optou-se por colocar esta informação através de uma mensagem privada enviada ao utilizador que acedeu a esta funcionalidade. Apesar de muita desta informação estar

disponível às ligações do utilizador na conta original, pode haver informação cuja partilha não seja desejada pelo utilizador, sendo, assim, mantida a privacidade da mesma, podendo esta informação ser, posteriormente introduzida nos campos corretos por parte do utilizador. Apesar de ser guardada a foto de perfil do utilizador, como o LinkedIn não disponibiliza nenhuma forma programática de esta ser alterada, é apenas colocada nesta parte a informação da localização da mesma;

- **connectionsToLinkedIn**(Document docConnections, LinkedInApiClient client): O LinkedIn também não oferece a possibilidade de seguir diretamente utilizadores pelo que, mais uma vez, optou-se por enviar uma mensagem privada com a informação do nome e identificador de cada utilizador ligado ao dono da conta de LinkedIn original.

3.2.2. Autenticação e autorização

Um dos aspetos mais importantes a ter em conta, quando se tenta interagir com uma rede social, é a autenticação. A autenticação, em suma, é o processo que permite determinar se alguém ou algo é realmente aquilo que diz ser.

Hoje em dia qualquer rede social exige pelo menos um método de autenticação para garantir que a pessoa que está a tentar interagir com a API disponibilizada é realmente quem diz ser, garantindo assim que o acesso está condicionado ao próprio utilizador.

Apesar de todas as redes sociais usarem variações do OAuth, estas aplicam processos de autenticação e/ou autorização que são ligeiramente diferentes entre si. No entanto, no geral, os processos são semelhantes, variando em alguns nomes atribuídos aos campos usados para efetuar as interações. De seguida será feita uma análise às diferentes interações que cada serviço usado necessita de fazer de modo a garantir a autenticação e autorização do utilizador. No Anexo B encontra-se uma descrição mais detalhada destes processos relativamente às diferentes redes sociais utilizadas, mas antes de o fazer convém definir as palavras referidas.

Em redes públicas ou privadas a autenticação é, geralmente feita através da utilização de *passwords*. O conhecimento da *password* costuma ser assumido como suficiente para garantir que o utilizador é autêntico, o que pode trazer problemas pois, algumas *passwords* podem ser facilmente roubadas, acidentalmente reveladas ou esquecidas.

Devido a isto, os negócios na Internet tendem a usar processos de autenticação mais rigorosos e que permitam uma melhor identificação. É de prever que o uso de certificados digitais verificados por uma entidade de certificação, como parte de infraestruturas de chaves públicas, se torne na maneira mais usual de autenticação nos próximos anos.

Logicamente um processo de autenticação garante também autorização, que garante que o utilizador tem permissões para aceder aos recursos pretendidos [43].

Autorização é um mecanismo para garantir o nível de acesso que o utilizador autenticado tem nos recursos disponibilizados no serviço utilizado. Os serviços de autorização tentam garantir que o utilizador tem permissões suficientes para efetuar a interação que pretende [44].

A grande maioria dos serviços que requerem autenticação, nos dias de hoje, usa OAuth, um *standard* para garantir a autorização dos serviços. Este *standard* será analisado no próximo subtópico.

3.2.2.1. OAuth

Como referido anteriormente, o OAuth é um *standard* para a autorização de serviços na *Web*. O seu desenvolvimento teve iníncio em 2006 por Blaine Cook, Chris Messina, David Recordon, Larry Halff, entre outros e tinha como objetivo inicial permitir a autenticação na API do Twitter.

A ideia do OAuth é permitir um acesso limitado aos serviços disponibilizados, desta forma um utilizador corretamente identificado pelo serviço poderá apenas aceder a um número restrito de funcionalidades, dependendo do grau de acesso que lhe seja concedido. Na página oficial do OAuth [45] é feita uma analogia do OAuth com algumas chaves especiais existentes em alguns carros de luxo para serem usadas, por exemplo, por pessoas responsáveis por estacionar os veículos em grandes festas. Essas chaves permitem apenas a condução por períodos limitados e podem impossibilitar o acesso a certas funções às quais o dono do veículo teria acesso, sendo esta a ideia por trás do OAuth. Deste modo, além de permitir um método para autorizar um utilizador corretamente identificado, também evita a utilização do nome do utilizador e *password*, garantindo, assim, que há um risco menor de haver uma utilização indesejada destes dados.

O OAuth não é um conceito novo, sendo apenas a combinação e normalização de vários protocolos existentes, mas é já extremamente usado, particularmente nas redes sociais e em serviços que permitam a partilha de informação ou que disponham de APIs que providenciam algum tipo de interação com os dados existentes [45].

O fluxo normal da utilização do OAuth encontra-se na Figura 30 e será descrito de seguida.

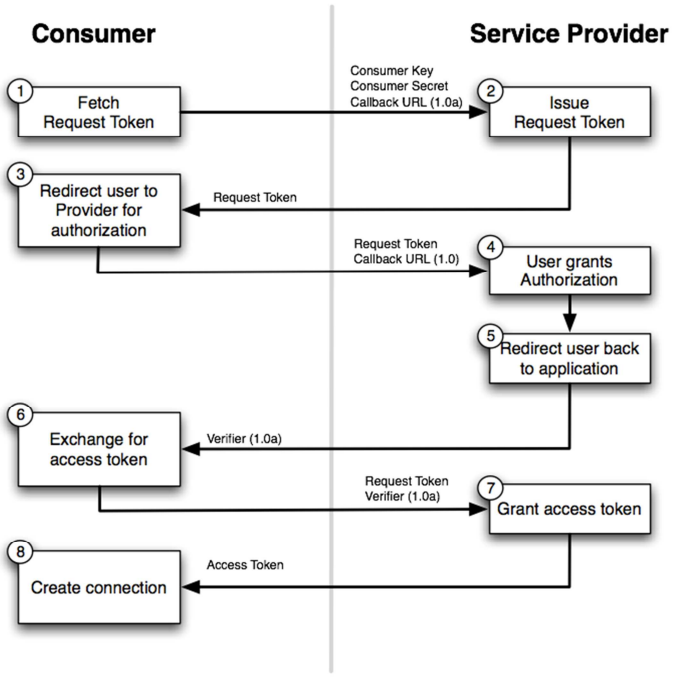

Figura 30 - Fluxo normal do OAuth [46]

Numa fase inicial o utilizador é forçado a proceder ao *login* no sistema alvo, caso ainda não o tenha feito, havendo a primeira troca de *tokens*, nomeadamente do *Request Token*.

De seguida, o utilizador terá de aceitar conceder à aplicação as permissões necessárias para o correto funcionamento da mesma, sendo estas permissões identificadas na página que é mostrada. Em algumas situações, como acontece com o Facebook, caso o utilizador aceite estas permissões uma primeira vez, não necessita de repetir este passo nas interações seguintes. Esta ação faz com que o utilizador receba um *Verifier* que poderá, num último passo, ser trocado por um *Access Token* que permitirá usar os serviços pretendidos.

No Anexo B encontra-se uma análise individual às interações necessárias para efetuar os processos de autenticação e autorização nos serviços utilizados na aplicação desenvolvida.

3.2.2.2. Google Picasa

Um dos serviços usados para o alojamento *online* das imagens e/ou vídeos obtidos na rede social selecionada é o Google Picasa. Este serviço também tem um processo de autenticação e autorização, embora seja diferente dos restantes processos usados na aplicação já que, em vez de utilizar o OAuth como é bastante comum junto dos serviços disponibilizados hoje em dia, utiliza uma solução proprietária do Google, denominada de Google AuthSub.

A partir de Abril de 2012 este método de autenticação e autorização foi substituído pelo OAuth 2.0, mas continuará a funcionar. Como os restantes exemplos já usam o OAuth e, no caso do Google Picasa, há a possibilidade de experimentar um método diferente de autenticar e autorizar o acesso à aplicação, optou-se por usar o Google AuthSub, até porque a migração para OAuth seria relativamente simples.

Aplicações *web* que necessitem de aceder a serviços protegidos por uma conta de utilizador Google têm de usar o serviço de *Authorization Proxy*. Para manter um nível alto de segurança, a interface de acesso a este serviço, o AuthSub, permite que a aplicação consiga o acesso sem ter de manipular qualquer tipo de informação associada ao *login* do utilizador.

O Google dá à pessoa que desenvolve a aplicação a opção de registar a aplicação, passando a ter de usar um ficheiro guardado com um certificado que permite assinar digitalmente os pedidos feitos. Para o âmbito deste projeto optou-se por usar uma abordagem não registada, dado que o processo de registo, além de moroso, criaria uma complexidade adicional ao trabalho desenvolvido.

Tal como aconteceu nos exemplos descritos anteriormente, antes de se poder interagir com o serviço desejado, é necessário proceder à autenticação e autorização. Neste caso, antes de conseguir aceder ao serviço, é obrigatório chamar o URL de autorização do Google, disponibilizando alguma informação necessária, tal como o nome do serviço a ser usado. Será mostrada uma página do Google para que o utilizador possa aceitar ou recusar o acesso à sua conta.

Caso o utilizador aceite as permissões pedidas, o Google redireciona o utilizador para a página da aplicação *web*, com a inclusão de um *token* de acesso para o serviço requisitado. A partir desse *token*, tal como nos casos anteriores, passa a ser possível interagir com o serviço.

Na Figura 31 está representado o processo de autorização do Google AuthSub.

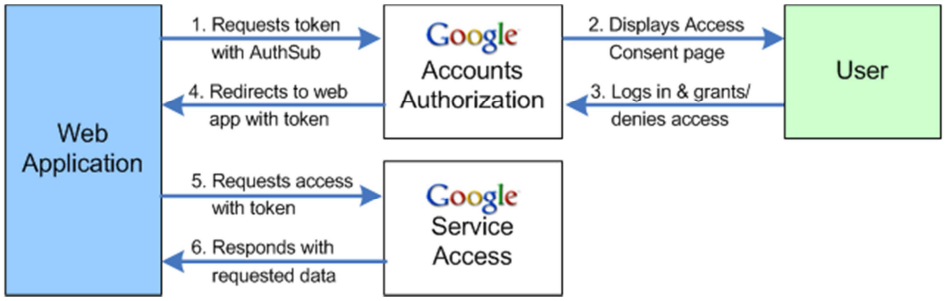

Figura 31 - Processo de autorização do Google AuthSub [47]

Esta autorização envolve uma sequência de interações entre três entidades: a aplicação *web*, os serviços do Google e o utilizador. Os passos que compõem esta sequência de interações são os seguintes:

1. Quando a aplicação *web* necessita de aceder a um serviço Google associado a uma conta de utilizador, faz uma chamada AuthSub ao serviço Google de *Authorization Proxy*;
2. O serviço de Autorização responde, mostrando uma página de pedido de acesso. Esta página gerida pelo Google pede ao utilizador para permitir ou negar o acesso aos seus serviços Google. Caso o utilizador não tenha *login* feito na sua conta Google, terá de o fazer;
3. O utilizador decide aceitar ou recusar o acesso à sua conta. Caso recuse o acesso, não será redirecionado para a aplicação *web* original, mas sim para uma página do Google;
4. Se o utilizador permitir o acesso, o serviço de Autorização redireciona o utilizador para a aplicação *web*. Em conjunto, será enviado um *token* que poderá ser usado para obter um *token* de acesso;
5. A aplicação *web* contacta o serviço Google com um pedido, usando o *token* anterior como agente para o utilizador;
6. Se o serviço Google reconhecer o *token*, providencia os dados pedidos.

Cada *token* de autorização é específico a uma única conta de utilizador, mas pode ser usado em múltiplos serviços. Existe a possibilidade de usar *tokens* válidos apenas para uma utilização, que permitem apenas uma única chamada ao serviço Google antes de serem anulados, ou de usar *tokens* de sessão, permitindo que seja feito um número ilimitado de chamadas aos serviços disponibilizados. No caso da aplicação desenvolvida, optou-se por usar *tokens* de sessão, mas, no final de cada utilização do Google Picasa, optou-se por revogar o *token*, de modo a impedir que o mesmo possa voltar a ser usado.

Desse modo, qualquer utilizador obtém um novo *token* de cada vez que utiliza a aplicação.

A integração do AuthSub numa aplicação *web* implica, então, os seguintes passos:

1. **Decidir se deve ou não registar-se a aplicação *web*:** Como já foi referido, as aplicações *web* podem ser registadas, garantindo um maior nível de segurança. Desse modo passam a ser reconhecidas pelo Google, sendo alterada a janela que permite ao utilizador fazer o *login*, caso seja necessário, e aceitar as permissões, passando as aplicações a ser identificadas por um nome bem elucidativo das mesmas, ao contrário de uma simples chamada de um URL;
2. **Decidir que tipo de *tokens* usar e como os gerir:** Um *token* de autorização recebido do Google tem como função o uso em interações futuras com o serviço requisitado. A escolha do *token*, seja para apenas uma interação ou para várias, depende das funcionalidades disponibilizadas pela aplicação. Por exemplo, um *token* de apenas uma utilização pode ser útil para utilizações esporádicas ou únicas. Alternativamente, como foi usado nesta aplicação, pode optar-se por obter *tokens* novos em cada utilização, desde que se configure um mecanismo de revogação de *tokens*, sendo que desse modo, para aumentar a segurança, passa a ser necessário ter o *login* feito e aceitar as permissões de cada vez que se acede à aplicação;
3. **Determinar o âmbito do serviço:** Tal como nos exemplos anteriores, também no AuthSub pode ser configurado o âmbito da interação. Alguns serviços disponibilizam um conjunto básico de interações, mas outras podem ser requisitadas, passando-as como um parâmetro de *scope*. No caso da interação da aplicação com o Google Picasa, no parâmetro *scope* passou-se apenas o endereço de dados deste serviço, nomeadamente "http://picasaweb.google.com/data/";
4. **Configurar um mecanismo para pedir e receber um *token* de autenticação:** O mecanismo configurado deve gerar uma chamada corretamente formada, denominada de AuthSubRequest, incluindo os valores obrigatórios de *next*, que identifica a localização da página para onde deve ser feito o redireccionamento, após um *login* correto e *scope* que, como já foi referido anteriormente, representa o âmbito da aplicação. Outros valores opcionais podem ser indicados como a existência ou não de autorização segura ou o tipo de *token* a utilizar. O mecanismo usado deve, ainda, ser

capaz de identificar os parâmetros recebidos após o redireccionamento da página, de modo a utilizá-los para gerar, posteriormente, um *token* de autenticação válido;

5. **Configurar mecanismos para pedir *tokens* de sessão e guardar ou revogá-los, se relevante:** Como foi referido anteriormente, optou-se por uma abordagem em que os *tokens*, depois de ser feita a interação, seriam revogados de modo a impedir que possam ser reutilizados. Desse modo não se torna necessário guardar os mesmos, mas passa a ser extremamente importante ter o cuidado de os manipular corretamente, garantindo que, após a interação, os mesmos são revogados corretamente;
6. **Configurar um mecanismo para requisitar o acesso a um serviço Google:** Tal como nos casos anteriores, depois de obtido um *token* de autenticação válido, passa a ser possível interagir com o serviço desejado, neste caso o Google Picasa. Para efetuar a interação, usou-se uma biblioteca que cria uma abstração sobre o Java das funcionalidades disponibilizadas pelas APIs existentes deste serviço. Esta biblioteca é disponibilizada pelo próprio Google e a sua documentação está disponível *online* para consulta [47,48].

3.2.3. Bibliotecas externas utilizadas

Após a obtenção dos *tokens* de acesso, descrita na subsecção anterior, passa a ser possível a interação com as redes sociais ou os serviços de alojamento *online*. Para evitar os pedidos explícitos através de endereços ou objetos JSON criados especificamente para esse efeito, surgem bibliotecas que criam uma abstração para linguagens de programação que permitem uma interação mais facilmente compreensível por quem desenvolve a aplicação. Neste capítulo serão analisadas as diferentes bibliotecas externas que foram utilizadas para proceder à interação com as redes sociais, com os serviços de alojamento, bem como as bibliotecas utilizadas para criar os ficheiros em RDF e para compreender e extrair a informação proveniente dos mesmos.

3.2.3.1. Scribe

Para ajudar na implementação das interações necessárias para efetuar a autenticação e a autorização nas redes sociais, excetuando o caso do Facebook, optou-se por utilizar uma biblioteca em Java, denominada de Scribe. Esta biblioteca é resumida, pelo autor, como uma biblioteca simples de OAuth para Java. O Scribe foi criado em Julho de 2010 por Pablo Fernandez que é, atualmente, engenheiro de *software* no LinkedIn.

O Scribe tem as seguintes vantagens:

- É extremamente simples, já que com apenas uma linha de código consegue configurar-se esta biblioteca para funcionar com a informação necessária à interação com uma rede social;
- Suporta várias *threads*, permitindo que a pessoa que desenvolve a aplicação consiga obter o máximo de performance possível;
- Suporta um grande número de APIs de OAuth 1.0 e 2.0 nativamente, nomeadamente provenientes do Google, Facebook, Yahoo, LinkedIn, Twitter, Foursquare, Vimeo, entre outras;
- É pequeno, sendo constituído por cerca de mil linhas de código e bastante simples de perceber. É ainda extremamente modular;
- Está preparado para a utilização no Sistema Operativo Android;
- É estável e foi submetido a diversos testes para garantir o seu correto funcionamento. Quando surge algum problema, tem mensagens de erro que, ao contrário de vários outros casos, dizem exatamente o que aconteceu de errado, quando e onde;
- Tem um grande suporte por parte de uma grande comunidade, inclusive tem uma secção dedicada no StackOverflow, um espaço de discussão para programadores profissionais e/ou entusiastas [49];
- Uma outra grande vantagem que tornou importante o uso desta biblioteca é o facto de ser facilmente extensível, permitindo a adição de novos serviços, desde que estes usem como esquema de autorização o OAuth. Um exemplo disso foi a fácil integração com o Personal Digital Repository.

3.2.3.2. RestFB

O RestFB é um cliente simples e flexível, escrito em Java e que interage com a API disponibilizada pelo Facebook. É *software open source* lançado sob os termos da Licença MIT. Foi criado por Mark Allen, com o patrocínio da empresa Transmogrify.

A versão mais recente, 1.6.9, lançada em Outubro de 2011 está disponível para *download* via Google Code.

As principais motivações, relativamente ao *design* deste cliente são:

- API pública minimalista;
- Máxima extensibilidade possível;
- Robustez para ir de encontro às frequentes alterações às APIs impostas pelo Facebook;
- Configuração simples orientada aos metadados;
- Não utilizar quaisquer dependências.

O RestFB também abrange outros aspetos que não tinham sido identificados como prioritários, mas que acabaram por ser bastante importantes. Esses aspetos são:

- Suporte para partes da API da Plataforma Facebook que não sejam intencionalmente acessíveis ao utilizador;
- Providenciar um mecanismo para obter chaves de sessão ou *tokens* de acesso OAuth;
- Utilizar o formato de transferência de dados XML em alternativa ao JavaScript Object Notation (JSON);
- Manter versões formalmente corretas de todos os métodos e códigos de erro pertencentes à API do Facebook.

Além destes aspetos, o RestFB implementa os objetos básicos que fazem parte de Graph API, a API disponibilizada aos utilizadores pelo Facebook. Esses objetos são os seguintes:

- **Album** – Permite o acesso a álbuns de fotografias;
- **Application** – Uma aplicação registada no Facebook;
- **Checkin** – Um *checkin* feito por um utilizador num certo local disponibilizado no Facebook;
- **Comment** – Um comentário feito num objeto da API;
- **Event** – Um evento criado no Facebook;
- **Group** – Um conjunto de utilizadores reunidos num grupo;
- **Link –** Uma ligação partilhada;

- **Note** – Uma nota criada por uma conta no Facebook;
- **Page** – Uma página de Facebook;
- **Photo** – Uma fotografia individual pertencente a um álbum;
- **Post** – Uma entrada individual na *feed* de um perfil do Facebook;
- **Status Message** – Uma mensagem de *status* presente no mural de um utilizador;
- **User** – O perfil de um utilizador;
- **Venue** – Possibilita a marcação de pontos de encontro, ou reuniões. Esta funcionalidade foi, entretanto, descontinuada pelo Facebook, não estando sequer documentada na página oficial;
- **Video** - Um vídeo individual.

Depois de se obter, através dos processos descritos no capítulo anterior, um *token* de acesso válido para o Facebook, o mesmo pode ser usado em conjunto com o RestFB de modo a proceder à interação com a rede social [50,51].

3.2.3.3. Twitter4J

O Twitter4J é uma biblioteca não oficial em Java para possibilitar a interação com as APIs disponibilizadas pelo Twitter. Através do Twitter4J passa a ser possível a integração de uma aplicação desenvolvida em Java com os serviços do Twitter.

Foi criada por Yusuke Yamamoto e as suas principais características são as seguintes:

- O facto de ser uma biblioteca 100% Java, o que faz com que funcione com qualquer plataforma Java;
- Está preparado para a utilização com a plataforma Android bem como com o Google App Engine;
- Não tem quaisquer dependências, pelo que não é necessário adicionar nenhum Java Archive (JAR) para além do Twitter4J;
- Inclui suporte embutido para OAuth;
- Inclui suporte nativo para GNU zip (Gzip).

Em termos de Sistema Operativo, o Twitter4J está preparado para funcionar em Windows ou em qualquer versão de Unix que suporte Java.

O Twitter4J é considerado *software open source* e pode ser utilizado em qualquer projeto, seja comercial ou não. Inclui *software* proveniente de JSON.org para interpretar as respostas em formato JSON provenientes do Twitter [52].

3.2.3.4. linkedin-j

O linkedin-j providencia uma abstração em Java para a interação com as APIs públicas disponibilizadas pelo LinkedIn. Esta biblioteca foi criada por Nabeel Mukhtar em Dezembro de 2009.

O linkedin-j é composto, na sua grande maioria, por código escrito em Java. Verificou-se um decréscimo considerável na atividade de desenvolvimento deste projeto nos últimos 12 meses, o que pode indicar que o interesse neste projeto esteja a ser cada vez menor, ou que o mesmo tenha atingido uma fase de maturação que requer cada vez menos alterações ao código.

Por curiosidade, este projeto teve cerca de 400 *commits*, ou seja, 400 envios de código alterado e/ou atualizado, segundo o modelo Constructive Cost Model (COCOMO), que não é mais do que um algoritmo que estima o custo de um modelo de *software*, o linkedin-j teve cerca de 122 anos de esforço e um custo estimado de cerca de 7 milhões de dólares.

No total, este projeto tem cerca de 54 mil linhas de código. A última atualização foi feita em Fevereiro de 2012 [53].

Tal como nas bibliotecas referidas anteriormente, para utilizar o linkedin-j é necessário ter um *token* de acesso válido, obtido através do processo de autenticação e autorização referente ao LinkedIn, explicado na subsecção anterior e no Anexo B. Depois de obtido este *token* passa a ser possível interagir com as APIs disponibilizada pelo LinkedIn. Para o primeiro passo é ainda necessário utilizar as duas chaves providenciadas pelo LinkedIn aquando da criação de uma aplicação dedicada para proceder às diferentes interações.

Os métodos disponibilizados pelo linkedin-j podem ser resumidos em 5 categorias diferentes:

- **Perfil:** Estes métodos permitem aceder à informação básica presente no perfil do utilizador autorizado. Permite, entre outras operações, obter o nome do utilizador, o seu número de identificação na rede social, a sua fotografia e o título em destaque na sua página;

- **Ligações:** Através dos métodos referentes às ligações do utilizador, é possível obter todos os contactos que o utilizador adicionou na sua rede, bem como a informação de perfil disponível de cada um deles;
- **Atualizações da rede:** Estes métodos são responsáveis por obter os diferentes conteúdos partilhados pelo utilizador na sua página de LinkedIn, bem como os comentários que fez;
- **Pesquisa:** Existem métodos que permitem pesquisar utilizadores através de palavras escolhidas. No caso do trabalho desenvolvido não foi necessário utilizar esta funcionalidade;
- **Mensagens:** É ainda possível obter as mensagens pessoais do utilizador, bem como enviar a um utilizador mensagens e/ou convites para adicionar à rede de contactos do utilizador [54].

3.2.3.5. Google Data Java Client

No caso da interação com o Google Picasa, nomeadamente com o serviço Picasa Web Albums, as bibliotecas em Java necessárias são fornecidas pelo próprio Google.

Depois de feito o *download* das bibliotecas, a utilização das mesmas exige a utilização de várias dependências necessárias para o correto funcionamento. Uma das dependências necessárias é o Guava, um projeto desenvolvido pelo Google e que contém várias bibliotecas reunidas num só conjunto. Esta biblioteca é obrigatória para que possa ser feita a interação com o serviço desejado.

As principais características do Google Data Java Client são:

- Integração do Java 5 e aplicações *web*, incluindo o Google App Engine;
- Protocolo Atom XML;
- Utilização de OAuth, bem como outros métodos de autenticação como o ClientLogin ou o AuthSub, usado neste projeto;
- Utiliza o java.net.HttpURLConnection como biblioteca HTTP de baixo-nível.

Esta biblioteca deixou de ser ativamente desenvolvida, exceto no caso do aparecimento de erros graves ou a adição de suporte a novas APIs disponibilizadas pelo Google. No entanto a mesma não foi deprecada, sendo considerada uma escolha estável. Existem alternativas, como o Google APIs Client Library for Java, criado para possibilitar a integração das APIs do Google com o Sistema Operativo Android, mas que também possibilita o acesso às funcionalidades necessárias para este projeto, mas

como, apesar de esta biblioteca ser ainda bastante utilizada, optou-se por prosseguir o desenvolvimento usando-a.

Este cliente em Java permite efetuar as seguintes operações:

- **Autenticação de utilizador único:** Podem ser introduzidas diretamente na aplicação desenvolvida as credenciais de acesso de um utilizador mas, por questões de segurança e também por esta possibilidade não fornecer as funcionalidades desejadas, esta operação não é necessária para este trabalho;
- **Autenticação de múltiplos utilizadores:** Este método de autenticação foi o selecionado para este trabalho. Implica a obtenção das credenciais de acesso para o utilizador, tal como descrito no capítulo anterior sendo, de seguida, necessário criar um objeto do tipo PicasawebService que receberá o parâmetro de autenticação obtido. A partir do momento em que é criado este objeto, deixa de ser necessário lidar explicitamente com o *token* de acesso, já que a biblioteca trata de o incluir nos pedidos seguintes;
- **Pedir uma lista de álbuns:** Com um processo de autenticação bem-sucedido passa a ser possível obter, por exemplo, a lista de álbuns existentes na conta de um utilizador. No caso da aplicação desenvolvida, não será necessário obter esta informação;
- **Adicionar um álbum:** Torna-se também possível criar novos álbuns. Para tal, depois de feita a autenticação, apenas é necessário definir qual o nome do novo álbum, sendo as restantes tarefas necessárias geridas pela biblioteca. Mais uma vez, neste trabalho, não é necessário usar esta funcionalidade, optando-se por colocar as imagens e vídeos num álbum existente por omissão que permite o *upload* dessa informação diretamente, sem necessitar da criação de álbuns. No serviço Google Picasa, o nome do álbum aparecerá como “Drop Box”;
- **Modificar as propriedades de um álbum:** É possível atualizar a informação de um álbum, nomeadamente o nome dado ao mesmo;
- **Apagar um álbum:** Através desta operação um álbum pode, facilmente, ser apagado, o que apagará também todos os ficheiros multimédia associados a esse álbum;
- **Pedir uma lista de fotografias:** Tendo a informação do nome do utilizador e da identificação do álbum torna-se possível aceder à lista de todas as fotografias existentes nesse álbum. Simultaneamente é, também, possível

obter a lista das fotografias mais recentemente adicionadas pelo utilizador. Por último, é possível listar as fotografias com base numa pesquisa feita. Nenhuma destas funcionalidades é necessária na aplicação;

- **Aceder à informação de fotografias:** Tal como é possível listar todas as fotografias, também é possível aceder à informação individual de cada fotografia;
- **Efetuar o *upload* de uma fotografia:** Existem duas opções disponíveis. O *upload* de uma fotografia com metadados associados ou um *upload* sem necessitar de passar qualquer informação relativa à fotografia. No primeiro caso, além da localização da fotografia, é necessário indicar à biblioteca o título da mesma, bem como uma descrição do conteúdo. No segundo caso apenas é necessário indicar a localização da fotografia. Na aplicação optou-se por usar a segunda abordagem, já que as fotografias serão guardadas com o intuito de garantir a sua segurança, não sendo relevante a navegação através das mesmas. Tal como foi referido anteriormente, para evitar que o utilizador seja forçado a escolher o nome dos álbuns que deseja usar de cada vez que recorrer a este serviço, as imagens e/ou vídeos são guardados num álbum criado por omissão;
- **Efetuar o *upload* de um vídeo:** Para efetuar o *upload* de um vídeo, não existem as duas opções referidas no caso das fotografias, sendo obrigatório indicar toda a informação acerca do vídeo. Para evitar a aborrecida tarefa de descrever pormenorizadamente qualquer vídeo existente na conta de utilizador, optou-se por escrever, nestes campos, apenas o nome do ficheiro sendo possível, desse modo, enviar um vídeo tendo apenas o nome do ficheiro, tal como acontece com as fotografias. Neste caso, o processo de *upload* é mais demorado do que acontece com as fotografias, podendo, por vezes, ser necessário algum tempo até o vídeo ser totalmente processado;
- **Atualizar uma fotografia:** É possível atualizar a informação associada a uma fotografia mas, como foi referido anteriormente, não se utiliza a informação associada a cada fotografia, pelo que se torna desnecessário o uso desta funcionalidade;
- **Apagar uma fotografia:** É possível apagar uma fotografia existente na conta de utilizador;
- **Trabalhar com *tags*:** É possível, de modo a organizar melhor os ficheiros existentes, atribuir *tags* a cada ficheiro. Existem também funcionalidades para

obter as *tags* existentes. Esta funcionalidade é bastante útil mas, no âmbito deste trabalho, não é necessária;

- **Comentários:** O Google Picasa suporta, ainda, a adição de comentários a uma fotografia por parte de utilizadores. Mais uma vez esta funcionalidade não é necessária já que, as imagens transferidas para este serviço, estarão apenas acessíveis ao utilizador [48,55].

3.2.4. Apache Jena

Apache Jena providencia uma *framework* completa para construir aplicações de *Web* Semântica em Java. Entre outras funcionalidades, disponibiliza *parsers* para lidar com conteúdo RDF/XML, bem como para o formato de triplos que foi descrito no capítulo referente ao RDF e uma API de programação em Java. Esta biblioteca segue todas as recomendações para o RDF e tecnologias relacionadas por parte do W3C [56].

No Jena, toda a informação proveniente de uma coleção de triplos RDF está contida numa estrutura de dados, definida como *Model*. Este modelo representa um grafo em RDF, já que contém um conjunto de nós, ligados entre si através de relações com nomes, pelo que, como foi visto nos exemplos do RDF, tendo um triplo como ‘example:abc foaf:name “Tito”’, este pode ser lido como “o recurso example:abc tem uma propriedade foaf:name com o valor “Tito””, mas o inverso não é verdadeiro.

No Java, é usada a classe *Model* como o principal recetor da informação RDF. Esta classe foi desenhada de modo a ter uma API muito rica, com muitos métodos que permitem uma fácil escrita de programas e aplicações baseados em RDF. Outra funcionalidade de *Model* é dar uma abstração sobre as diferentes formas de guardar os nós e relações RDF.

Existe ainda uma abstração bastante mais simples, o *Graph*, que é usado pelo Jena como interface comum para a salvaguarda de RDF a um nível mais baixo. Esta abstração tem uma API bastante mais simples, sendo mais fácil a sua reimplementação.

Para um programador, o conceito mais importante relativamente aos diferentes recetores de RDF disponíveis no Apache Jena é o de *Model* que representa uma API em Java extremamente rica e com vários métodos para interagir com dados em RDF.

Tal como foi referido na secção própria, no RDF existem dois tipos de nós, as referências URI, que descrevem os recursos, e os literais. No Jena, a interface Java

Resource representa os recursos e a interface *Literal* representa os literais. Como ambas podem aparecer em nós, a interface comum *RDFNode* é uma superclasse das duas.

Tal como referido, as ligações entre dois recursos ou entre um recurso e um literal têm sempre um nome que identifica a propriedade. Tal como o RDF usa URIs como nomes para recursos, também no caso das propriedades acontece isso, sendo estas propriedades apenas um caso especial de recursos RDF. No Jena estas propriedades são identificadas pelo objeto *Property* que é uma subclasse Java de *Resource*.

Para representar um único triplo, o Jena utiliza uma classe denominada de *Statement*. De acordo com a especificação do RDF, apenas recursos podem ser o sujeito de um triplo RDF, sendo que o objeto pode ser um recurso ou um literal. Os métodos principais para extrair os elementos de um *Statement* são:

- getSubject() que retorna um *Resource*;
- getObject() que retorna um *RDFNode*;
- getPredicate() que retorna um *Property*;

Na Tabela 8 está um resumo das várias *packages* que integram o projeto Apache Jena, permitindo uma melhor compreensão das suas funcionalidades [57].

Package	Descrição
chh.jena.rdf.model	O núcleo do Jena. Permite criar e manipular grafos RDF.
chh.jena.datatypes	Providencia as interfaces principais através das quais os tipos de dados são descritos para o Jena.
chh.jena.ontology	Abstrações e classes convenientes para aceder e manipular ontologias representadas em RDF.
chh.jena.rdf.arp	Um *parser* para RDF/XML.
chh.jena.rdf.listeners	Verificar alterações nas frases de um modelo.
chh.jena.reasoner	Suporta um conjunto de inferências que permitem derivar informação adicional de um modelo RDF.
chh.jena.shared	Classes de utilidade comuns.
chh.jena.vocabulary	Classes constantes com objetos pré-definidos constantes para classes e propriedades definidas em vocabulários conhecidos.
chh.jena.xmloutput	Escrever RDF/XML.

Tabela 8 - *Packages* do Apache Jena [57]

Como exemplo, o ficheiro RDF mostrado na Figura 8, usado para descrever esta dissertação poderia ser representado, no Apache Jena, pelo código representado na Figura 32.

```
Model model = ModelFactory.createDefaultModel();

String ResumeRDFNS = "http://purl.org/captsolo/resume-rdf/0.2/cv#";
String TeachNS = "http://linkedscience.org/teach/ns#";

model.setNsPrefix("foaf", FOAF.NS);
model.setNsPrefix("dc", DCTerms.NS);
model.setNsPrefix("cv", ResumeRDFNS);
model.setNsPrefix("teach", TeachNS);

Property cvStudiedIn = model.createProperty(ResumeRDFNS + "studiedIn");
Property teachTeacher = model.createProperty(TeachNS + "teacher");

Resource recurso = model.createResource()
        .addProperty(DCTerms.creator, model.createResource()
        .addProperty(FOAF.name, "Tito Azevedo")
        .addProperty(VCARD.EMAIL, "titoazevedo@ua.pt")
        .addProperty(cvStudiedIn, "Universidade de Aveiro"))
        .addProperty(DCTerms.title, "Repositório de dados pessoais das Redes Sociais")
        .addProperty(teachTeacher, "Joaquim Arnaldo Carvalho Martins")
        .addProperty(DCTerms.date, "01/01/2012");


model.write(System.out, "RDF/XML-ABBREV");
```

Figura 32 - Código em Java para representar um documento RDF

Neste código começa-se por criar um modelo, que, tal como foi descrito anteriormente, é o elemento fundamental na criação de conteúdo em RDF.

As duas linhas seguintes tratam de definir qual é o prefixo dado aos *namespaces* utilizados. Em alguns casos esta biblioteca já o faz, mas, para evitar o uso de *namespaces* fictícios como "j.0" ou "j.1", deve atribuir-se o prefixo correto para os casos em que tal não aconteça.

De seguida cria-se o recurso principal ao qual se vai adicionar uma propriedade com a informação do criador. De seguida são adicionados três recursos com propriedades relativas ao criador, nomeadamente o nome, o *e-mail* e o local de ensino. Poderia, eventualmente, ser adicionada mais informação acerca do criador. Por último são adicionadas três propriedades, já fora das propriedades relativas ao criador do documento, com o título, o orientador da dissertação e a data que representa, neste caso, o ano.

No final é feito um *write* que irá mostrar o resultado obtido. Este *write* recebe, como parâmetros, em primeiro lugar o destino para onde deve escrever a informação. Neste caso, como estamos na presença de um simples teste, optou-se por escrever

diretamente no System.out, ou seja, no ecrã. No caso da aplicação desenvolvida, passa-se como parâmetro um OutputStream que será o responsável por escrever a informação no ficheiro correspondente. Em segundo lugar recebe o tipo de escrita. Neste caso, além de se escrever em RDF/XML, usou-se também o formato "ABBREV" que representa um ficheiro RDF mais legível, já que o Jena tem, por omissão, representações que não são tão facilmente percetíveis.

O resultado final deste código está representado na Figura 33.

```
run:
<rdf:RDF
    xmlns:teach="http://linkedscience.org/teach/ns#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
    xmlns:dc="http://purl.org/dc/terms/"
    xmlns:cv="http://purl.org/captsolo/resume-rdf/0.2/cv#">
  <rdf:Description>
    <dc:date>01/01/2012</dc:date>
    <teach:teacher>Joaquim Arnaldo Carvalho Martins</teach:teacher>
    <dc:title>Repositório de dados pessoais das Redes Sociais</dc:title>
    <dc:creator rdf:parseType="Resource">
      <cv:studiedIn>Universidade de Aveiro</cv:studiedIn>
      <vcard:EMAIL>titoazevedo@ua.pt</vcard:EMAIL>
      <foaf:name>Tito Azevedo</foaf:name>
    </dc:creator>
  </rdf:Description>
</rdf:RDF>
BUILD SUCCESSFUL (total time: 1 second)
```

Figura 33 - Resultado obtido com o código Java descrito

Como é possível observar, o resultado final é semelhante ao ficheiro RDF descrito na Figura 8. Há ligeiras diferenças relativamente à ordem em que a informação aparece mas, como a ordem não é importante num ficheiro RDF, nem podem ser dadas quaisquer garantias relativamente à mesma, o conteúdo dos dois ficheiros é exatamente o mesmo.

De notar que o Apache Jena tem inúmeras dependências que têm, obrigatoriamente, de estar no projeto desenvolvido. Este acaba, talvez, por ser o ponto mais negativo desta biblioteca.

3.2.5. Java DOM Parser

Para efetuar o *parsing*, ou seja a leitura e interpretação, dos ficheiros RDF, como foi referido anteriormente, é possível usar o Apache Jena. No entanto, após a execução de alguns testes com os resultados obtidos nos passos anteriores, verificou-se que não disponibilizava as funcionalidades pretendidas. Além de não tornar possível a obtenção dos atributos de diferentes recursos encadeados, não possibilitava a ordenação, por exemplo pela data, dos conteúdos partilhados de uma forma simples e fácil.

Desse modo optou-se por pesquisar uma biblioteca diferente que fosse capaz de efetuar o *parsing* de um ficheiro RDF de modo simples e eficaz. Foi escolhido o Document Object Model (DOM).

O DOM não é mais do que uma API para documentos HTML e XML corretamente formados. Define a estrutura lógica dos documentos e o modo como um documento é acedido e manipulado.

O DOM surgiu como uma especificação para possibilitar que programas em JavaScript e Java pudessem ser portáveis em *browsers web*. No entanto, quando o grupo de trabalho DOM foi formado no W3C, juntaram-se pessoas de outros domínios, como o HTML e o XML, acabando por influenciar o desfecho desta biblioteca.

Um dos pontos fortes do DOM é o facto de ser uma especificação do W3C. Como tal, um dos objetivos principais é providenciar uma interface de programação *standard* que pode ser usada em diferentes ambientes e aplicações. Foi desenhado para permitir o uso de qualquer linguagem de programação. No caso deste trabalho, como nas situações anteriores, optou-se por usar o Java.

O DOM está separado em 3 níveis distintos, definindo os objetos e propriedades de todos os elementos, bem como os métodos para aceder aos mesmos:

- *Core* DOM – modelo *standard* para qualquer documento estruturado;
- XML DOM – modelo *standard* para documentos XML;
- HTML DOM – modelo *standard* para documentos HTML.

Como a notação de um ficheiro RDF, numa análise generalizada, pode ser vista como XML, interessa-nos a parte responsável por este, neste caso o XML DOM. Este é caracterizado por ser, como já foi referido em parte:

- Um modelo de objeto *standard* para XML;
- Uma interface de programação *standard* para XML;
- Independente da plataforma e da linguagem de programação;
- Um *standard* W3C.

Resumidamente, o XML DOM é um *standard* para obter, alterar, adicionar e remover elementos XML.

Tendo, por exemplo, o XML representado na Tabela 9, o XML DOM transforma essa informação numa árvore DOM como está representada na Figura 34.

```
<book>
        <title>Título do Livro</title>
        <author>Autor do Livro</author>
        <ISBN>Número ISBN do Livro</ISBN>
</book>
```

Tabela 9 - Ficheiro XML

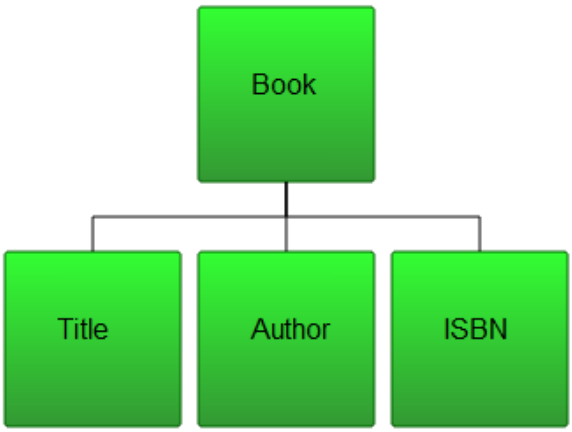

Figura 34 - Árvore DOM do ficheiro XML [58]

Depois de ser obtida esta árvore DOM, a mesma pode ser manipulada como qualquer outra árvore ou grafo, podendo aceder aos seus elementos.

De acordo com o DOM, tudo o que está num documento XML é um nó. Desse modo, o DOM indica-nos que:

- O documento inteiro é um nó de documento;
- Qualquer elemento XML é um nó de elemento;
- O texto presente nos elementos XML é um nó de texto;
- Qualquer atributo é um nó de atributo;
- Comentários são nós de comentários.

Um erro comum, quando se usa o DOM, é esperar que um nó de elemento contenha texto já que, como foi referido anteriormente, o texto será sempre guardado em nós de

texto. Como exemplo, se tivermos “<year>2012</year>”, o nó de elemento “<year>” contém um nó de texto com o valor “2012”. “2012” não é o valor do elemento “<year>”.

O objeto DOM *Document* representa um documento XML. Quando se efetua o *parsing* de um ficheiro XML usando o Java DOM Parser, obtém-se um objeto *Document*.

As duas características mais utilizadas do DOM são:

1. Aceder a elementos-filho de um elemento;
2. Aceder a atributos de um elemento.

Inicialmente, um *Document* contém apenas um elemento de raiz, que pode ser obtido utilizando o método getDocumentElement().

Este elemento raiz pode ter filhos que podem ser outros elementos, comentários, instruções de processamento, caracteres, entre outros. Para se obter os filhos de um elemento, existe um método com o nome getChildNodes que retorna um objeto NodeList, que não é mais do que uma lista de nós, que pode ser percorrida para obter a informação desejada.

Para obter os atributos de um elemento, existem dois métodos especialmente criados para este efeito, o getAttribute e o getAttributeNode. Na maioria das vezes será suficiente usar o getAttribute, mas podem surgir casos em que seja necessário passar o atributo obtido para um ou mais métodos que necessitem de aceder a mais informação acerca do atributo para o processar, desse modo passa a ser necessário utilizar a segunda opção [58,59,60].

Para mostrar um exemplo funcional desta biblioteca será usada, mais uma vez, a informação presente na Figura 8. O código usado para este exemplo está na Figura 35.

```
File ficheiroRDF = new File("c:\\tito\\rdfTese.rdf");
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(ficheiroRDF);
doc.getDocumentElement().normalize();

NodeList nListNome = doc.getDocumentElement().getElementsByTagName("foaf:name").item(0).getChildNodes();
NodeList nListEmail = doc.getDocumentElement().getElementsByTagName("vcard:EMAIL").item(0).getChildNodes();
NodeList nListEnsino = doc.getDocumentElement().getElementsByTagName("cv:studiedIn").item(0).getChildNodes();
NodeList nListTitulo = doc.getDocumentElement().getElementsByTagName("dc:title").item(0).getChildNodes();
NodeList nListOrientador = doc.getDocumentElement().getElementsByTagName("teach:teacher").item(0).getChildNodes();
NodeList nListAno = doc.getDocumentElement().getElementsByTagName("dcterms:date").item(0).getChildNodes();
Node noNome = (Node) nListNome.item(0);
Node noEmail = (Node) nListEmail.item(0);
Node noEnsino = (Node) nListEnsino.item(0);
Node noTitulo = (Node) nListTitulo.item(0);
Node noOrientador = (Node) nListOrientador.item(0);
Node noAno = (Node) nListAno.item(0);

String nome = noNome.getNodeValue();
String email = noEmail.getNodeValue();
String ensino = noEnsino.getNodeValue();
String titulo = noTitulo.getNodeValue();
String orientador = noOrientador.getNodeValue();
String ano = noAno.getNodeValue();
System.out.println("Nome do autor: " + nome);
System.out.println("E-mail do autor: " + email);
System.out.println("Local de ensino: " + ensino);
System.out.println("Título: " + titulo);
System.out.println("Orientador: " + orientador);
System.out.println("Ano: " + ano);
```

Figura 35 - Código Java para exemplificar o funcionamento do DOM Parser

De notar que este exemplo é meramente para mostrar o funcionamento desta biblioteca já que, por exemplo, no trabalho final existe um método para o processo que vai desde a obtenção da NodeList ao retorno do NodeValue.

Neste exemplo começa-se por definir o ficheiro que contém o documento RDF. Como foi referido, neste ficheiro está o conteúdo representado na Figura 8.

De seguida são executados 4 métodos que permitem a construção de um novo *Document*, o início do *parsing* e a normalização do ficheiro.

Seguidamente é feito o acesso aos elementos pretendidos, neste caso pretende-se obter os valores representados por “foaf:name”, “vcard:EMAIL”, “cv:studiedIn”, “dc:title”, “teach:teacher” e “dcterms:date”, obtendo desse modo todos os nós-filho existentes. O valor estará guardado no primeiro elemento, já que só temos um valor para cada um destes elementos. No caso de existir mais do que um poderia ser feito um ciclo para obter toda a informação pretendida.

Por último é feito um getNodeValue, para obter o valor da informação pretendida. Neste passo poderia ser obtida toda a informação associada a um nó, mas como se pretende apenas o seu valor, não foi necessário obter qualquer informação adicional.

No final é mostrada a informação pretendida no ecrã. O resultado deste pequeno programa está representado na Figura 36.

```
run:
Nome do autor: Tito Azevedo
E-mail do autor: titoazevedo@ua.pt
Local de ensino: Universidade de Aveiro
Título: Repositorio de Dados das Redes Sociais
Orientador:  Joaquim Arnaldo Carvalho Martins
Ano:  01/01/2012
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figura 36 - Resultado da execução do programa anterior

Como se pode ver, tornou-se possível extrair a informação pretendida de uma forma rápida e simples.

3.3. Camada de Dados

Esta camada, como o nome indica, é responsável por providenciar o acesso a dados guardados dentro dos limites do sistema, bem como dados acessíveis através de outros sistemas ligados a este através de, por exemplo, serviços. Esta camada expõe interfaces genéricas que podem ser consumidas na Camada de Negócio.

No caso específico da aplicação desenvolvida, há uma necessidade de salvaguardar a informação obtida de um modo seguro e persistente, usando-se para isso um projeto denominado de Personal Digital Repository [34,35] e que trata de guardar os conteúdos provenientes das redes sociais em Bases de Dados NoSQL, nomeadamente utilizando o MongoDB e o Neo4j.

Relativamente às APIs disponibilizadas pelo Repositório Digital Pessoal, estas são acedidas, mais uma vez, através de REST. Depois de ser feita a autenticação e autorização do utilizador existem três métodos que podem ser acedidos para guardar ou receber os documentos RDF envolvidos na interação.

O primeiro método, storeContent, como o nome indica, permite guardar na Base de Dados NoSQL do Repositório Digital Pessoal os documentos RDF obtidos da rede social escolhida pelo utilizador. Este método recebe os seguintes parâmetros:

- contentHandler: Este parâmetro define qual vai ser o plugin que vai lidar com o pedido efetuado. Cada *plugin* lida com um tipo diferente de dados, e a presença deste parâmetro simplifica o processamento que teria de ser feito sobre o conteúdo para se determinar um *plugin* que pode processar o conteúdo enviado;

- content: O tipo deste parâmetro é variável consoante o tipo definido no contentHandler anterior e pode ir desde um ficheiro normal a dados presentes no tipo JSON.

Existe, tal como referido anteriormente, um outro método que suporta as necessárias interações da aplicação, nomeadamente o retrieveContent que permite obter o conteúdo previamente guardado na Base de Dados NoSQL do Repositório. Este método tem os seguintes parâmetros:

- contentHandler: Mais uma vez este método tem um contentHandler que define o *plugin* utilizado neste método;
- offset: Este parâmetro é do tipo inteiro e vai ser utilizado para efetuar a paginação dos resultados. Este parâmetro é opcional e, no âmbito deste projeto, não é necessário;
- limit: O parâmetro limit permite estabelecer um limite máximo para os resultados gerados. Mais uma vez estamos na presença de um parâmetro opcional que não é usado neste projeto;
- beginDate: Pode ser definida uma data inicial, de modo a obter apenas os dados a partir de uma data definida;
- endDate: Tal como no parâmetro anterior, também pode ser definida uma data final para efetuar a filtragem da informação.

Como cada rede social cria um número variável de documentos RDF, para um correto funcionamento do processo, optou-se por receber esta informação, no método retrieveContent, através de um ficheiro comprimido, neste caso com a extensão zip, que apenas terá de ser gerido no lado da aplicação de modo a extrair a informação presente no mesmo.

Além dos dois métodos referidos anteriormente, há ainda um terceiro método, showContent, que vai permitir obter um único item baseado no identificador passado. Este método recebe os seguintes parâmetros:

- contentHandler: Mais uma vez utiliza um contentHandler com o mesmo objetivo dos casos anteriores;
- id: Recebe também um identificador que permite identificar inequivocamente a informação desejada.

Apesar da possível utilidade do método showContent descrito anteriormente, optou-se por usar apenas os dois primeiros métodos nesta aplicação dado que o utilizador, nas funcionalidades desenvolvidas, terá apenas interesse em guardar toda a informação gerada na rede social escolhida e obter também toda a informação.

É importante referir que, mesmo sendo a informação enviada para o Repositório Digital Pessoal, é mantida uma cópia local da informação. Caso o utilizador opte por guardar as imagens e/ou vídeos provenientes da rede social no disco, a cópia local será mantida nessa mesma pasta, caso contrário será mantida numa pasta denominada de "Temp", criada na raiz do disco. Deste modo reduz-se o risco da perda desta informação, garantindo que, mesmo que haja uma falha de comunicação com a Base de Dados NoSQL presente no Repositório Digital Pessoal que vai guardar os documentos, os dados são guardados localmente.

4. Repositório de dados pessoais das Redes Sociais

Neste capítulo será feita uma apresentação do trabalho desenvolvido e dos resultados obtidos pelas interações escolhidas pelo utilizador. Como existem múltiplas possibilidades de interação com as redes sociais e/ou com os serviços disponibilizados, apenas será mostrada um possível cenário, tentando agrupar o máximo de funcionalidades, de modo a criar um exemplo adequado. No exemplo selecionado, começa-se por obter informação proveniente de uma conta de Twitter para, de seguida, migrar essa informação para uma conta de Facebook, explorando a possibilidade de mudar o conteúdo entre contas de redes sociais diferentes.

Numa fase inicial, quando se abre a página, é apresentado ao utilizador o menu superior, já referido anteriormente, bem como uma breve ajuda à navegação (Figura 37).

Figura 37 - Página inicial da aplicação *web*

No menu superior é visível a opção de regressar à página principal, o acesso às diferentes redes sociais e os botões para obter informação e contactar o autor. A existência deste menu em todas as páginas integradas na aplicação permite uma fácil navegação entre todos os elementos da aplicação. Na Figura 38 é visível o comportamento do menu superior quando o ponteiro do rato se situa sobre uma das opções disponíveis, tal como já tinha sido descrito anteriormente, com o aparecimento do texto identificativo de cada opção, de modo a evitar quaisquer dúvidas.

Figura 38 – Menu superior com opção selecionada

Na Figura 39 está representado um diagrama de sequência que representa as interações que o utilizador necessita de efetuar para obter a informação proveniente de uma conta do Twitter.

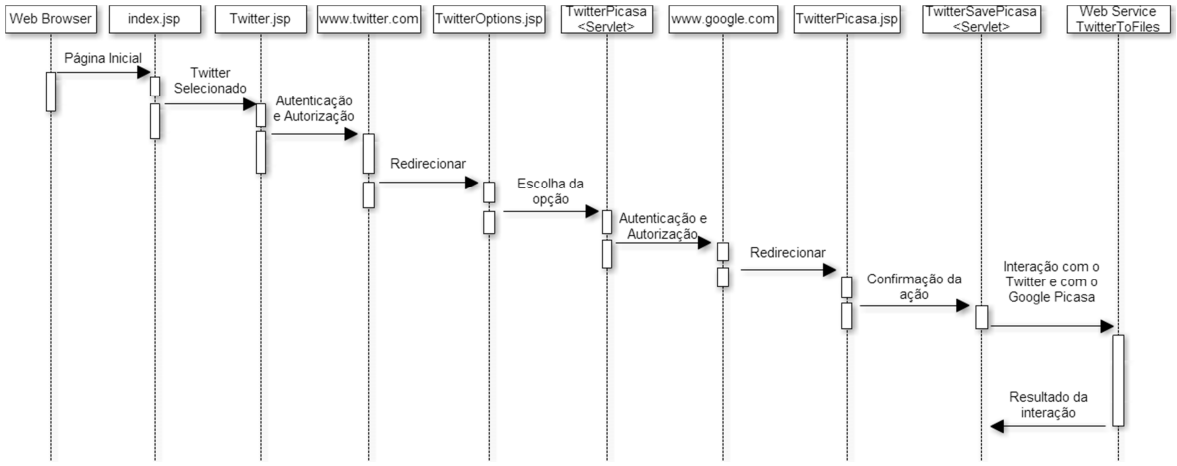

Figura 39 - Interações para a obtenção do conteúdo da conta de Twitter

Numa fase inicial o utilizador encontra-se na página inicial, mostrada na Figura 37. Escolhendo o símbolo do Twitter no menu superior, o utilizador necessitará de se autenticar na rede social e autorizar o acesso à sua informação. De seguida o utilizador poderá escolher uma das opções disponíveis para interagir com esta rede social, nomeadamente guardar os dados no disco, guardar no serviço de alojamento de imagens e vídeos Google Picasa ou fazer o upload de informação previamente guardada para o Twitter. Escolhendo, para efeitos da demonstração, a opção de guardar os ficheiros multimédia no Google Picasa será, mais uma vez, necessário proceder à autenticação e posterior autorização neste serviço. De seguida, estando todos os requisitos para a interação preenchidos, é mostrada ao utilizador uma descrição do que acontecerá de seguida, de modo a evitar que o utilizador seja confrontado com um tempo de espera indesejado. Assim que o utilizador confirma que deseja proceder à interação, aparecerá uma animação dando conta do processamento da informação. No final deste processo,

será mostrada a informação de que o processo foi bem-sucedido ou, eventualmente, o tipo de erro encontrado.

No caso selecionado, quando a interação é bem-sucedida, os ficheiros multimédia estarão disponíveis para acesso no Google Picasa e estarão guardados na Base de Dados NoSQL do Repositório Digital Pessoal os ficheiros RDF gerados com o conteúdo do utilizador na rede social. Serão cinco os ficheiros RDF criados, nomeadamente o twitterDirectMessages.rdf com as mensagens pessoais recebidas e enviadas pelo utilizador, o twitterFollowers.rdf com todos os utilizadores que são seguidos pelo dono da conta acedida, o twitterFollowing.rdf com todos os utilizadores que seguem o dono da conta, o twitterPersonal.rdf com a informação pessoal do utilizador e o twitterStatus.rdf com todas as mensagens públicas criadas pelo utilizador, denominadas de *tweets*.

Na Figura 40 é analisado o primeiro documento referido, twitterDirectMessages.rdf.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:sioc="http://rdfs.org/sioc/ns#">
  <rdf:Description rdf:about="http://www.twitter.com/TesedoTito">
    <sioc:Post>
      <rdf:Description rdf:about="http://www.twitter.com/259667675674984448">
        <vcard:UID>259667675674984448</vcard:UID>
        <sioc:content>Boas amigo, tudo bem?</sioc:content>
        <sioc:addressed_to>bLiNdPT</sioc:addressed_to>
        <dcterms:creator>TesedoTito</dcterms:creator>
        <dcterms:created>Sat Feb 25 15:49:32 BST 2012</dcterms:created>
      </rdf:Description>
    </sioc:Post>
    <sioc:Post>
      <rdf:Description rdf:about="http://www.twitter.com/259667908890861568">
        <vcard:UID>259667908890861568</vcard:UID>
        <sioc:content>Também, obrigado :)</sioc:content>
        <sioc:addressed_to>bLiNdPT</sioc:addressed_to>
        <dcterms:creator>TesedoTito</dcterms:creator>
        <dcterms:created>Sat Feb 25 15:50:28 BST 2012</dcterms:created>
      </rdf:Description>
    </sioc:Post>
    <sioc:Post>
      <rdf:Description rdf:about="http://www.twitter.com/259667792616357888">
        <vcard:UID>259667792616357888</vcard:UID>
        <sioc:content>Tudo bem. E contigo?</sioc:content>
        <sioc:addressed_to>TesedoTito</sioc:addressed_to>
        <dcterms:creator>bLiNdPT</dcterms:creator>
        <dcterms:created>Sat Feb 25 15:50:00 BST 2012</dcterms:created>
      </rdf:Description>
    </sioc:Post>
  </rdf:Description>
</rdf:RDF>
```

Figura 40 - twitterDirectMessages.rdf

Inicialmente, tal como em qualquer documento RDF, são declarados os *namespaces* utilizados neste documento específico. Além dos mais habituais, optou-se por usar também o vcard quer permite representar em RDF objetos vCard, um formato normalizado para a representação de informação pessoal de utilizadores [61,62], o dcterms que representa os termos de metadados mantidos pelo Dublin Core Metadata

Initiative (DCMI) [63] e o SIOC que pretende permitir a representação de informação proveniente de comunidades *online* [64].

De seguida é criado um rdf:Description contendo o *link* do perfil do utilizador. Por cada mensagem existente é criado um sioc:Post que representa a mensagem. Dentro dele surgem vários atributos, nomeadamente o vcard:UID que contém o número de identificação da mensagem, o sioc:content representando o conteúdo da mensagem original, o sioc:addressed_to que representa o recetor da mensagem, o dcterms:creator que representa o criador da mensagem e o dcterms:created com informação relativa ao momento da criação da mensagem.

Na Figura 41 está disponível o conteúdo do documento twitterFollowers.rdf

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#">
  <foaf:Person rdf:about="http://www.twitter.com/bLiNdPT">
    <vcard:UID>47093428</vcard:UID>
    <vcard:FN>bLiNdPT</vcard:FN>
  </foaf:Person>
  <foaf:Person rdf:about="http://www.twitter.com/iluisoliveira">
    <vcard:UID>558192975</vcard:UID>
    <vcard:FN>iluisoliveira</vcard:FN>
  </foaf:Person>
</rdf:RDF>
```

Figura 41 - Representação do documento twitterFollowers.rdf

Neste caso, além de usar o *namespace* relativo ao vcard, definido anteriormente, também é usado o *namespace* do projeto Friend Of A Friend (FOAF) que permite guardar informação pessoal, bem como de relações existentes entre pessoas [65].

No documento representado é usado o foaf:Person para representar cada pessoa que consta na lista de seguidores do utilizador. Dentro desta definição é guardada a identificação do utilizador, através do vcard:UID e o nome do utilizador, através do vcard:FN.

Na Figura 42 está representado o documento twitterFollowing.rdf que representa a informação dos utilizadores que o dono da conta segue.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
    xmlns:dcterms="http://purl.org/dc/terms/">
  <rdf:Description rdf:about="http://www.twitter.com/bLiNdPT">
    <vcard:UID>47093428</vcard:UID>
    <vcard:FN>bLiNdPT</vcard:FN>
  </rdf:Description>
</rdf:RDF>
```

Figura 42 - Documento twitterFollowing.rdf

Como é facilmente visível, a estrutura deste documento é, em tudo, semelhante à do documento analisado anteriormente. Na verdade a informação presente nestes dois documentos é semelhante, pelo que não existe necessidade de alterar a estrutura dos mesmos.

Na Figura 43 é representado o documento twitterPersonal.rdf que contém a informação pessoal do utilizador.

```
<rdf:RDF
    xmlns:ma="http://www.w3.org/ns/ma-ont#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:sioc="http://rdfs.org/sioc/ns#">
  <rdf:Description rdf:about="http://www.twitter.com/TesedoTito">
    <vcard:Locality>Aveiro</vcard:Locality>
    <foaf:homepage>http://www.ua.pt</foaf:homepage>
    <dcterms:created>Tue Feb 21 13:28:14 BST 2012</dcterms:created>
    <dcterms:bibliographicCitation>Ola! Sou um teste!!!</dcterms:bibliographicCitation>
    <vcard:Region>Europe/Lisbon</vcard:Region>
    <ma:hasLanguage>pt</ma:hasLanguage>
    <vcard:UID>587384348</vcard:UID>
    <vcard:PHOTO>https://lh5.googleusercontent.com/.../fotocartaoestudante.jpg</vcard:PHOTO>
    <foaf:accountName>TesedoTito</foaf:accountName>
  </rdf:Description>
</rdf:RDF>
```

Figura 43 - Documento twitterPersonal.rdf

Neste documento surge mais um *namespace*, neste caso o ma que facilita a integração de informação relativa a objetos multimédia na *Web*, nomeadamente vídeos, imagens e áudio [66].

Depois de declarados os *namespaces*, é criado um rdf:Description, mais uma vez com o URL que permite aceder à conta do utilizador. Na propriedade vcard:Locality é guardada a localização do utilizador, em foaf:homepage é guardada a página definida pelo utilizador, em dcterms:created é guardada a data de criação da conta, em dcterms:bibliographicCitation é guardada a informação definida pelo utilizador para o campo "Bio", em vcard:Region está a região em que o utilizador se encontra, na propriedade ma:hasLanguage é guardada a língua utilizada pelo utilizador, em vcard:UID, mais uma vez, é guardado o número de identificação do utilizador e em vcard:PHOTO é

guardada a localização da imagem de perfil do utilizador. Para efeitos de apresentação, no documento representado não está presente o URL completo da imagem, mas este possibilitaria aceder diretamente à imagem guardada no Google Picasa. Por último é usada a propriedade foaf:accountName para guardar o nome da conta utilizada.

O último documento criado é o twitterStatus.rdf que contém todos os *tweets* escritos pelo utilizador. Este documento está representado na Figura 44

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:sioc="http://rdfs.org/sioc/ns#">
  <rdf:Description rdf:about="http://www.twitter.com/TesedoTito">
    <sioc:Post>
      <rdf:Description rdf:about="http://twitter.com/TesedoTito/status/230792123623022592">
        <sioc:content>Boas malta dos twitters!</sioc:content>
        <dcterms:creator>Tese do Tito</dcterms:creator>
        <dcterms:created>Sat Feb 25 23:28:24 BST 2012</dcterms:created>
        <dcterms:type>status</dcterms:type>
      </rdf:Description>
    </sioc:Post>
    <sioc:Post>
      <rdf:Description rdf:about="http://twitter.com/TesedoTito/status/230792923204161536">
        <foaf:img>https://lh5.googleusercontent.com/.../401103_376890602338589_1527230581_n.jpg</foaf:img>
        <sioc:content>Teste de imagem! http://t.co/dEw7OoJ9</sioc:content>
        <dcterms:creator>Tese do Tito</dcterms:creator>
        <dcterms:created>Sat Feb 25 23:31:35 BST 2012</dcterms:created>
        <dcterms:type>photo</dcterms:type>
      </rdf:Description>
    </sioc:Post>
  </rdf:Description>
</rdf:RDF>
```

Figura 44 - Documento twitterStatus.rdf

Por cada *tweet* existente é criado um sioc:Post com um rdf:Description com o *link* do *tweet*. Em cada *tweet* é identificado o conteúdo do mesmo através de um sioc:content, o criador dele através do dcterms:creator, a data de criação através do dcterms:created e o tipo de mensagem através do dcterms:type. No caso do *tweet* atual ter uma imagem associada é criado ainda um foaf:img com o URL que permite aceder à imagem ou, caso a imagem esteja guardada no disco, o caminho para a sua localização. Mais uma vez, para que o texto pudesse ser legível, o URL da imagem foi alterado.

Na Figura 45 encontra-se o conjunto de interações necessárias para efetuar a migração da conta de Twitter cuja informação foi obtida no exemplo anterior para uma conta de Facebook criada.

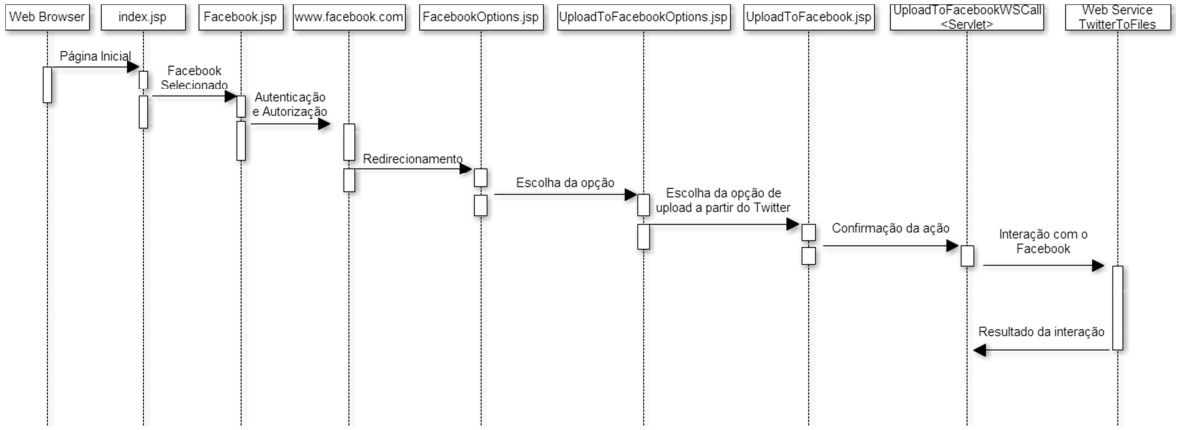

Figura 45 - Colocação de informação proveniente do Twitter no Facebook

A página inicial é semelhante à do processo anterior, mas desta vez é escolhido o Facebook como rede social. Depois de, mais uma vez, ser feito o processo de autenticação e autorização, o utilizador é redirecionado para as diferentes opções onde deverá escolher a opção de efetuar o *upload* da informação guardada para o Facebook. Para que sejam apenas permitidas as interações que façam sentidos, o utilizador terá, de seguida, de escolher a origem da informação de entre as opções disponibilizadas, onde deverá escolher o Twitter, dado ter sido esta a rede social escolhida para obter a informação. De seguida o processo é em tudo semelhante ao anterior, havendo, no último passo, a chamada ao Web Service responsável pela colocação da informação no Facebook, o FilesToFacebook.

No Anexo C encontra-se uma representação visual dos passos necessários para a obtenção dos resultados descritos anteriormente, bem como do aspeto final da conta usada para efetuar a migração.

Para além dos resultados obtidos com as interações descritas, é relevante mostrar também o resultado obtido através da salvaguarda da informação associada com uma conta de Facebook, dada a diversidade existente nesta rede social. Relativamente à informação dos amigos do utilizador e das mensagens privadas trocadas pelo mesmo, a informação é semelhante à que está presente nos documentos anteriores, mas relativamente aos restantes documentos, será mostrado um exemplo de cada um deles.

Na Figura 46 está representado um exemplo do documento facebookPersonal.rdf, que representa a informação pessoal do utilizador da rede social.

```
<rdf:RDF
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:aaiso="http://purl.org/vocab/aiiso/schema#"
    xmlns:rss="http://purl.org/rss/1.0/"
    xmlns:scf="http://www.iwi-iuk.org/material/RDF/Schema/Class/scf#"
    xmlns:ma="http://www.w3.org/ns/ma-ont#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:cv="http://purl.org/captsolo/resume-rdf/0.2/cv#"
  <rdf:Description rdf:about="http://www.facebook.com/100003255198822">
    <vcard:EMAIL>titoazevedo@ua.pt</vcard:EMAIL>
    <vcard:NICKNAME>tesetito</vcard:NICKNAME>
    <cv:employedIn>inexistente</cv:employedIn>
    <aaiso:description>2012</aaiso:description>
    <cv:jobTitle>President</cv:jobTitle>
    <rss:link>http://www.facebook.com/tesetito</rss:link>
    <foaf:interest>female</foaf:interest>
    <foaf:birthday>01/27/1986</foaf:birthday>
    <aaiso:name>University of Aveiro</aaiso:name>
    <ma:hasLanguage>British English</ma:hasLanguage>
    <cv:birthPlace>Salreu, Aveiro, Portugal</cv:birthPlace>
    <foaf:gender>male</foaf:gender>
    <dc:date>Sat Jul 07 13:35:51 BST 2012</dc:date>
    <foaf:firstName>Tese</foaf:firstName>
    <vcard:Country>pt_PT</vcard:Country>
    <vcard:Region>1.0</vcard:Region>
    <cv:maritalStatus>Divorced</cv:maritalStatus>
    <vcard:NOTE>Teste&#xD;
Citação 2&#xD;
Citação 3</vcard:NOTE>
    <vcard:PHOTO>c:\tito\foto.jpg</vcard:PHOTO>
    <scf:Religion>Religion</scf:Religion>
    <foaf:family_name>Tito</foaf:family_name>
    <aaiso:name>Escola Secundária de Albergaria-a-Velha</aaiso:name>
    <cv:Locality>Aveiro, Portugal</cv:Locality>
    <ma:hasLanguage>Português</ma:hasLanguage>
    <vcard:Locality>Albergaria-a-Velha</vcard:Locality>
    <aaiso:description>2005</aaiso:description>
    <aaiso:teaches>High School</aaiso:teaches>
    <scf:Politics>Undefined</scf:Politics>
    <aaiso:teaches>College</aaiso:teaches>
    <cv:startDate>Fri Jan 01 00:00:00 GMT 2010</cv:startDate>
    <dcterms:bibliographicCitation>Olá amigos!</dcterms:bibliographicCitation>
  </rdf:Description>
</rdf:RDF>
```

Figura 46 - Exemplo do documento facebookPersonal.rdf

Neste documento é visível a grande diversidade de informação que o Facebook possibilita, começando por ser visível na elevada quantidade de *namespaces* presentes neste documento RDF. Além dos que foram utilizados nos exemplos anteriores, temos neste exemplo o dc cujo *namespace* aponta para o mesmo do dcterms analisado anteriormente [63], o aiiso que permite representar informação académica, o que é bastante útil dado que muita da informação pessoal partilhada no Facebook está relacionada com a carreira académica do utilizador [67], o rss que permite definir canais

que são constituídos por itens que podem ser obtidos através de URLs [68], o scf que significa “Class Scheme for Science Fields” e permite, no caso específico da aplicação desenvolvida, representar as ideologias religiosas e políticas do utilizador [69] e o cv (ResumeRDF) que permite representar todo o tipo de informação associada a um Curriculum Vitae de um utilizador, nomeadamente, no caso desta aplicação, do histórico profissional do utilizador bem como de outro tipo de informação pessoal [70]. O tipo de informação pessoal, tal como já foi referido, é mais abundante do que nas outras redes sociais usadas neste projeto, o que pode trazer alguns problemas. Em primeiro lugar causou dificuldades dado que alguns tipos de informação não tinham *schemas* em RDF disponíveis para usar, obrigando a alguma pesquisa. Obriga também à utilização de mais recursos, fazendo com que o utilizador tenha de esperar mais algum tempo comparativamente ao tempo de espera das restantes redes sociais. Na Figura 47 está um exemplo do conteúdo presente no documento facebookStatus.rdf.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:pac="http://dig.csail.mit.edu/2008/PAC/ontology/pac#"
    xmlns:og="http://ogp.me/ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:sioc="http://rdfs.org/sioc/ns#">
  <rdf:Description rdf:about="http://www.facebook.com/100003255198822">
    <sioc:Post>
      <rdf:Description rdf:about="http://www.facebook.com/photo.php?fbid=376890602338589&amp;set=p.376890602338589&amp;type=1">
        <foaf:img>c:\testeFacebookTese/401103_376890602338589_1527230581_n.jpg</foaf:img>
        <pac:PhotoAlbum>Sem Album</pac:PhotoAlbum>
        <sioc:content>Isto é o mundo!</sioc:content>
        <dcterms:creator>Tito Azevedo</dcterms:creator>
        <dcterms:created>Mon Feb 20 17:11:20 GMT 2012</dcterms:created>
        <dcterms:type>photo</dcterms:type>
      </rdf:Description>
    </sioc:Post>
    <sioc:Post>
      <rdf:Description rdf:about="http://www.facebook.com/100003255198822/posts/160228134095692">
        <og:video>c:\testeFacebookTese/379902862037363_35245.mp4</og:video>
        <sioc:content>Teste de um vídeo!</sioc:content>
        <dcterms:creator>Tito Azevedo</dcterms:creator>
        <dcterms:created>Fri Feb 24 10:47:04 GMT 2012</dcterms:created>
        <dcterms:type>video</dcterms:type>
      </rdf:Description>
    </sioc:Post>
    <sioc:Post>
      <rdf:Description rdf:about="http://www.facebook.com/100003255198822/posts/152750408176798">
        <sioc:content>Boas grande amigo :)</sioc:content>
        <dcterms:creator>Tito Azevedo</dcterms:creator>
        <dcterms:created>Mon Feb 13 22:33:39 GMT 2012</dcterms:created>
        <dcterms:type>status</dcterms:type>
      </rdf:Description>
    </sioc:Post>
  </rdf:Description>
</rdf:RDF>
```

Figura 47 - Exemplo do documento facebookStatus.rdf

Este documento introduz dois novos *namespaces,* nomeadamente o pac que singnifica Photo Access Control e que permite representar informação acerca de fotografias, o que acaba por ser bastante útil dado que o Facebook tem uma enorme componente dedicada a este tipo de conteúdo [71] e o og que representa Open Graph,

um protocolo bastante usado no Facebook e cujo *namespace* permite, nesta aplicação, representar por exemplo informação associada a um vídeo [72]. Neste ficheiro é visível que, ao contrário do que acontecia com o ficheiro de *status* do Twitter, além de haver suporte a mensagens de texto e mensagens que contêm imagens, temos ainda vídeos. Neste trabalho os vídeos também são guardados o que, apesar de ser bastante útil para um utilizador que use com frequência este formato, pode provocar algum tempo de espera acrescido, dado que os ficheiros de vídeo são, tipicamente, bastante maiores comparativamente com os ficheiros de imagem.

5. Conclusão

Neste capítulo vai ser feita uma análise crítica aos objetivos propostos no início do desenvolvimento do projeto, verificando se os mesmos foram, ou não, alcançados.

Seguidamente será feita uma análise às limitações encontradas no decorrer do desenvolvimento deste trabalho.

De seguida será feito um levantamento dos diferentes tipos de conhecimentos adquiridos durante o desenvolvimento deste projeto.

Por último serão analisadas as diferentes possibilidades de evolução do projeto num eventual desenvolvimento de trabalho futuro.

5.1. Objetivos concluídos

Os objetivos propostos no início do desenvolvimento do trabalho foram alcançados. Foi criada uma aplicação *web* totalmente funcional que permite salvaguardar o conteúdo multimédia partilhado por um utilizador numa rede social, bem como o restante conteúdo textual. Esta aplicação é de fácil utilização, intuitiva e facilita a interação com as redes sociais, algo que era extremamente importante no desenvolvimento do projeto.

Os conteúdos multimédia podem ser guardados no disco ou em serviços de alojamento *online*, outro dos objetivos pretendidos, enquanto os conteúdos textuais serão guardados em formato RDF, para uma fácil interpretação por outros sistemas, numa Base de Dados NoSQL a funcionar num projeto independente deste.

É ainda possível, com este trabalho, efetuar o inverso do que foi descrito, isto é, poder colocar a informação guardada, tanto textual como multimédia, de volta na rede social podendo, onde tal faça sentido, haver uma migração entre redes sociais.

Além destes objetivos, como foi sendo analisado durante este documento, foi possível criar uma aplicação que respeita os critérios de usabilidade identificados, sendo visualmente funcional e apelativa.

Por razões de segurança, são usados os métodos de autenticação e autorização integrados nos serviços utilizados, garantindo assim que a informação introduzida pelo utilizador será tratada da melhor maneira possível.

5.2. Limitações impostas pelas redes sociais

Durante o desenvolvimento deste projeto e, à medida que eram desenvolvidas as necessárias interações com as redes sociais, foram surgindo algumas limitações, quer ao nível das APIs disponibilizadas pelas redes sociais, quer ao nível do que é disponibilizado pelas empresas aos utilizadores.

Nesta subsecção é feita uma pequena análise de algumas limitações identificadas na interação com os serviços utilizados, fazendo uma separação das mesmas por rede social:

- **Facebook:** O Facebook faz questão de impedir que os utilizadores consigam, programaticamente, enviar mensagens privadas. Esta limitação é imposta para evitar utilizações incorretas desta plataforma, nomeadamente para o envio de mensagens de publicidade e/ou outro tipo de *spam*. Ao mesmo tempo também dificulta o acesso às mensagens privadas dos utilizadores, apenas possibilitando a leitura das últimas 25 mensagens trocadas com cada utilizador o que, em alguns casos, não permite recuperar todas as mensagens existentes. Esta rede social também não permite a adição programática de amigos, sendo necessária a sua adição manual. Por outro lado, como foi referido anteriormente, também não permite a adição programática de informação pessoal do utilizador, sendo esta limitação ultrapassada com o envio da informação sob a forma de uma nota apenas lida pelo utilizador;
- **Twitter:** Uma das grandes limitações do Twitter, embora possa ser vista também como uma grande vantagem, é o facto de cada mensagem poder ter, no máximo, 140 caracteres. Esta limitação obrigou à alteração do modo de lidar com a informação da rede social. Por exemplo, no caso de o utilizador desejar colocar informação numa conta de Twitter, torna-se impossível colocar uma mensagem semelhante ao que acontece, como no exemplo descrito no Anexo C, com o Facebook, com informações acerca da mensagem original, como o autor original e a data de criação da mensagem, já que o número de caracteres o impossibilita. Este facto torna, também, complicada a migração de uma conta de Facebook para uma de Twitter. Outra das limitações impostas por esta rede social é um limite de 350 chamadas à API por hora. Este número pode parecer, à primeira vista, pouco limitativo, mas tendo em conta que, por exemplo, cada interação com a rede social para a obtenção de uma mensagem é equivalente a uma chamada, pode compreender-se que

esta limitação poderá causar resultados indesejados na execução da aplicação desenvolvida. Eventualmente este processo poderá ser resolvido ao entrar em contacto com o *staff* responsável pelo Twitter com um pedido para a adição da aplicação a uma lista de aplicações com permissões especiais, mas este processo é, aparentemente, demorado e nem sempre concluído com sucesso;

- **LinkedIn:** No caso do LinkedIn também surgiram algumas limitações que obrigaram a tomar algumas decisões de modo a evitar a perda da informação guardada. Em primeiro lugar o LinkedIn não tem um modo programático de permitir a alteração da informação pessoal, desde o nome e data de nascimento, passando pelos contactos pessoais até à informação profissional do utilizador. Para ultrapassar esta limitação optou-se por enviar uma mensagem privada ao utilizador com esta informação estando o utilizador, posteriormente, obrigado a colocar manualmente as partes consideradas relevantes desta informação. Ao mesmo tempo também não existe, por parte do LinkedIn, maneira de introduzir diretamente a informação relativa às ligações que o utilizador tem, sendo esta limitação tratada do mesmo modo que aquela que foi descrita anteriormente, através do envio de uma mensagem privada com esta informação.

5.3. Aprendizagem

Para desenvolver este projeto e, à medida que ia sendo necessário adicionar novas funcionalidades à aplicação, tornou-se necessário obter conhecimentos relativos a diversos elementos fundamentais no desenvolvimento de aplicações *Web*.

Relativamente a linguagens, foi necessário aprender ou aprofundar conhecimentos em:

- HTML;
- CSS (Cascading Style Sheets);
- JSP;
- Java;
- JavaScript;
- AJAX (Asynchronous JavaScript and XML);
- XML;

- RDF.

Relativamente a ferramentas para o desenvolvimento deste trabalho, como já foi referido, utilizou-se o NetBeans IDE 7.2 que agrupa todo o conjunto de funcionalidades necessário para desenvolver e publicar uma aplicação *Web* como era pretendida neste projeto.

A necessidade de autenticação e autorização no acesso às redes sociais exigiu um estudo prévio considerável, já que implica uma forma de obter credenciais seguras diferente dos sistemas habituais. Numa fase inicial, muito do tempo de desenvolvimento do projeto foi gasto na tentativa de lidar com estes sistemas de autenticação e autorização. Foi necessário consultar documentação, ver exemplos e efetuar diversos testes até ser possível incluir estes sistemas na aplicação *Web*, o que, eventualmente, acabou por acontecer.

O uso do RDF também implicou um estudo prévio considerável. Por ser uma linguagem com uma sintaxe bastante específica, embora com semelhanças notórias com o XML, obrigou à pesquisa, por entre a documentação existente. Das muitas soluções existentes para manipular esta linguagem foram escolhidas as duas utilizadas neste trabalho, nomeadamente o Apache Jena e o Java DOM Parser, já que pareceram ser aquelas que produziam os melhores resultados.

Este trabalho também obrigou à aprendizagem da filosofia subjacente ao NoSQL, apesar de esta estar incluída num projeto distinto que não foi desenvolvido no âmbito deste trabalho, apenas foi ligeiramente alterado para conseguir interagir com os resultados obtidos com esta aplicação desenvolvida.

5.4. Trabalho Futuro

O objetivo principal deste trabalho, numa fase inicial, era o de efetuar uma aplicação que, não só permitisse aplicar os conhecimentos adquiridos ao longo dos anos de estudo no curso, em conjunto com alguns novos conhecimentos que teriam de ser adquiridos ou melhorados, mas sobretudo proporcionar a possibilidade de criar um projeto que fosse útil e pudesse ser usado por alguém.

Apesar de o trabalho realizado completar os objetivos propostos e permitir todas as interações definidas anteriormente, existem várias funcionalidades que poderão, eventualmente, ser adicionadas no futuro de forma a completar e melhorar a aplicação desenvolvida.

Uma das eventuais adições poderia ser um mecanismo capaz de efetuar o *download* dos conteúdos partilhados nas redes sociais de forma automática, entre períodos de tempo definidos pelo utilizador, tal como foi visto em algumas das soluções existentes no mercado, como por exemplo no Backupify. A verdade é que este serviço teria bastante utilidade, não sendo integrado neste projeto apenas por ser visto como uma funcionalidade adicional, não sendo fundamental a sua inclusão.

Com o passar do tempo será bastante normal que vão surgindo cada vez mais redes sociais. O trabalho está feito de modo a garantir que qualquer rede social possa ser adicionada. Não há nenhuma forma automática de incluir uma nova rede social, dado que todas elas diferem nos processos de autenticação e autorização, na comunicação com as APIs disponibilizadas, nas limitações impostas aos conteúdos, etc., mas uma nova rede social pode ser, com relativa facilidade, incluída neste trabalho dado que o projeto foi desenhado de forma a que cada rede social tenha o seu módulo independente das restantes. Este ponto também é válido para eventuais alterações nos tipos de conteúdos suportados por uma dada rede social, sendo possível atualizar, com relativa facilidade, os tipos de dados obtidos da rede social selecionada.

Algo que podia continuar a evoluir seria a exploração das redes sociais. Algumas das redes sociais usadas neste trabalho não eram usadas com muita regularidade previamente, o que implica uma possível falta de conteúdo partilhado. Apesar de terem sido criados diversos exemplos para testar as diferentes funcionalidades de cada rede social, é possível que não sejam exploradas todas as funcionalidades disponíveis, pelo que uma evolução a este trabalho poderia passar por melhorar as interações com algumas das redes sociais.

Também poderão ser adicionados serviços de alojamento adicionais sendo que, tal como acontece com as redes sociais, a sua integração não implica alterações ao nível dos restantes serviços.

Devido às limitações de um projeto desta natureza, não é possível testar a aplicação com diversos utilizadores. Foram feitos alguns testes com mais do que um utilizador, mas com um número limitado que não possibilita a obtenção de conclusões muito fidedignas, pelo que outro dos passos futuros passaria por um teste com utilizadores de modo a garantir o funcionamento correto da aplicação em diferentes cenários.

Referências

[1] Couts, Andrew (2012, Jun.). Facebook may cause serious mental health problems in kids, studies show [Em linha]. Disponível em: http://www.digitaltrends.com/social-media/facebook-may-cause-serious-mental-health-problems-in-kids/.

[2] Connolly, D. (2012, Sep.). A Little History of the World Wide Web [Em linha]. Disponível em: http://www.w3.org/History.html.

[3] Portela, David. (2012, Jun.). Social Networks [Em linha]. Disponível em: http://www.goliveheaven.com/social.php.

[4] Usenet Service Review. (2012, Jun.). *What is Usenet? – Usenet FAQ* [Em linha]. Disponível em: http://www.usenetservicereview.com/what-is-usenet.

[5] Haynes, Charles. (2012, Jun.). Learn About The WELL [Em linha]. Disponível em: http://www.well.com/aboutwell.html.

[6] Ostrow, Adam. (2012, Jun.). Geocities to Shutdown; What Was Geocities, You Ask? [Em linha]. Disponível em: http://mashable.com/2009/04/23/geocities-shutdown/.

[7] Lycos. (2012, Jun.). *About Tripod* [Em linha]. Disponível em: http://www.tripod.lycos.com/about.tmpl.

[8] Businessweek. (2012, Jun.). News Corp.'s Place in MySpace [Em linha]. Disponível em: http://www.businessweek.com/stories/2005-07-18/news-corp-dot-s-place-in-myspace.

[9] Selleck, Evan. (2012, Jun.). Facebook Visited More Than Google in 2010, Traffic Analyst Firm Says [Em linha]. Disponível em: http://www.slashgear.com/facebook-visited-more-than-google-in-2010-traffic-analyst-firm-says-30122076/.

[10] Bullas, Jeff. (2012, Jun.). Google+ vs Facebook – Infographic [Em linha]. Disponível em: http://www.jeffbullas.com/2012/06/14/google-vs-facebook-infographic/.

[11] Protalinski, Emil. (2012, Jun.). Facebook bans KDE applications, deletes photos uploaded with them [Em linha]. Disponível em: http://www.zdnet.com/blog/facebook/facebook-bans-kde-applications-deletes-photos-uploaded-with-them/1766.

[12] Decker, Stefan; Melnik, Sergey; Van Harmelen, Frank; Fensel, Dieter; Klein, Michel; Broekstra, Jeen; Erdmann, Michael and Horrocks, Ian. (2012, Jul.). The Semantic Web: The Roles of XML and RDF [Em linha]. Disponível em: http://classweb.gmu.edu/kersch/infs770/Semantic_Web_16_2/Semantic%20Web.pdf.

[13] Rouse, Margaret. (2012, Jul.). Resource Description Framework (RDF) [Em linha]. Disponível em: http://searchsoa.techtarget.com/definition/Resource-Description-Framework.

[14] Swoogle. (2012, Sep.). *Swoogle Semantic Web Search Engine* [Em linha]. Disponível em: http://swoogle.umbc.edu/.

[15] Penry, Andrew. (2012, Jul.). A Comparison of two major dynamic web platforms (LAMP vs. WISA) [Em linha]. Disponível em: http://www.shawnolson.net/a/302/a_comparison_of_two_major_dynamic_web_platforms_lamp_vs_wisa.html.

[16] Hanley, Jason M. (2012, Jul.). Web Development: A Comparison of Three Major Platforms [Em linha]. Disponível em: http://www.syllogisticsoftware.com/papers/Web_Development_Technology_Comparison.html.

[17] Facebook. (2012, Mar.). *Facebook: New Download Your Info* [Em linha]. Disponível em: http://pt.scribd.com/doc/38841132/Facebook-New-Download-Your-Info.

[18] SocialSafe. (2012, Mar.). *SocialSafe – Back up your social networks* [Em linha]. Disponível em: http://www.socialsafe.net/.

[19] Palsule, Mahendra. (2012, Mar.). How To Backup & Archive All Your Facebook Data [Em linha]. Disponível em: http://www.makeuseof.com/tag/how-to-backup-archive-all-your-facebook-data/.

[20] Backupify. (2012, Mar.). *Online Data Backup: Restore & Backup Gmail, Google Apps, & More* [Em linha]. Disponível em: https://www.backupify.com/.

[21] Salesforce.com. (2012, Sep.). *CRM – The Enterprise Cloud Computing Company* [Em linha]. Disponível em: http://www.salesforce.com/company/.

[22] Pick&Zip. (2012, Mar.). *Download Facebook Pictures and Videos* [Em linha]. Disponível em: http://www.picknzip.com/.

[23] BackupMyTweets. (2012, Mar.). *Backup Your Twitter Account* [Em linha]. Disponível em: http://backupmytweets.com/.

[24] Palmer, Sean B. (2012, Jul.). The Semantic Web: An Introduction [Em linha]. Disponível em: http://infomesh.net/2001/swintro/.

[25] Manola, Frank and Miller, Eric. (2012, Jul.). RDF Primer [Em linha]. Disponível em: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[26] McCreary, Dan and McKnight, William. (2012, Sep.). The CIO's Guide to NoSQL [Em linha]. Disponível em: https://s3.amazonaws.com/dataversity/documents/CIO_Guide_NoSQL_v4.pdf.

[27] Vogels, Werner. (2012, Sep.). Amazon's Dynamo [Em linha]. Disponível em: http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html.

[28] Browne, Julian (2012, Sep.). Brewer's CAP Theorem [Em linha]. Disponível em: http://www.julianbrowne.com/article/viewer/brewers-cap-theorem.

[29] Burd, Greg. (2012, Jul.). NoSQL [Em linha]. Disponível em: http://static.usenix.org/publications/login/2011-10/openpdfs/Burd.pdf.

[30] PHP24. (2012, Sep.). *NoSQL: Cassandra, MongoDB, CouchDB, Redis, Riak, HBase, Membase, Neo4j* [Em linha]. Disponível em: http://www.php42.com/index.php/database/72-essay-review/382-NoSQL-%20Cassandra,%20MongoDB,%20CouchDB,%20Redis,%20Riak,%20HBase,%20Membase,%20Neo4j?tmpl=component&print=1&page=

[31] Neo4j. (2012, Sep.). *What is Neo4j?* [Em linha]. Disponível em: http://neo4j.org/learn/

[32] Meier, J.D.; Hill, David; Homer, Alex; Taylor, Jason; Bansode, Prashant; Wall, Lonnie; Boucher Jr., Rob and Bogawat, Akshay, *Microsoft Application Architecture Guide*, 2nd Edition". Portland, OR: Microsoft Press, 2009.

[33] M.Horton, Helen. (2012, Jun.). Web Site Specification [Em linha]. Disponível em: http://helenmhorton.com/hwgcoursework/intro_to_webdesign/site_specification.html.

[34] Sousa, J.; Pereira, M. and Martins, J.A., “Improving browser history using semantic information,” *ICEIS 2012 - Proceedings of the 14th International Conference on Enterprise Information Systems*, vol. 2, pp. 305–311, Jun. 2012

[35] Soares, R.; Pereira, M. and Martins, J.A. (2012,Oct.). Recolha, preservação e contextualização de objectos digitais para dispositivos móveis com android, *Iberian Journal of Information Systems and Technologies* [Em linha]. *0(9)*, 75–89. Disponível em: http://ojs. academypublisher.com/index.php/risti/article/view/risti097589.

[36] Koetsier, John. (2012, Sep.). Social media demographics 2012: 24 sites including Twitter, Facebook and LinkedIn [Em linha]. Disponível em: http://venturebeat.com/2012/08/22/social-media-demographics-stats-2012/.

[37] Nielsen, Jakob. (2012, Jun.). Usability 101: Introduction to Usability [Em linha]. Disponível em: http://www.useit.com/alertbox/20030908.html.

[38] Nielsen, Jakob. (2012, Jun.). Ten Usability Heuristics [Em linha]. Disponível em: http://www.useit.com/papers/heuristic/heuristic_list.html.

[39] jQuery Project. (2012, Jul.). *jQuery Project* [Em linha]. Disponível em: http://jquery.org/.

[40] w3schools.com. (2012, Jul.). *Introduction to Web Services* [Em linha]. Disponível em: http://www.w3schools.com/webservices/ws_intro.asp.

[41] Kim, Yong Mook. (2012, Jul.). JAX-WS Tutorial [Em linha]. Disponível em: http://www.mkyong.com/tutorials/jax-ws-tutorials/.

[42] Rouse, Margaret. (2012, Jul.). JAX-WS (Java API for XML Web Services) [Em linha]. Disponível em: http://searchsoa.techtarget.com/definition/JAX-WS.

[43] Rouse, Margaret. (2012, Jul.). Authentication [Em linha]. Disponível em: http://searchsecurity.techtarget.com/definition/authentication.

[44] Carter, Robert. (2012, Sep.). Authentication vs. Authorization [Em linha]. Disponível em: http://www.duke.edu/~rob/kerberos/authvauth.html.

[45] Hammer-Lahav, Eran. (2012, Jul.). OAuth [Em linha]. Disponível em: http://oauth.net/about/.

[46] Spring Framework. (2012, Sep.). *Chapter 2. Service Providers* [Em linha]. Disponível em: http://www.springframework.net/social/refdoc/serviceproviders.html.

[47] Google Developers. (2012, Mar.) *AuthSub for Web Applications* [Em linha]. Disponível em: https://developers.google.com/accounts/docs/AuthSub.

[48] Google Developers (2012, Mar.). *Picasa Web Albums Data API: Developer's Guide: Java* [Em linha]. Disponível em: https://developers.google.com/picasa-web/docs/2.0/developers_guide_java.

[49] Fernandez, Pablo. (2012, Jan.). Scribe – Simple OAuth library for Java [Em linha]. Disponível em: https://github.com/fernandezpablo85/scribe-java.

[50] Allen, Mark. (2011, Nov.). RestFB – A Lightweight Java Facebook Graph API and Old REST API Client [Em linha]. Disponível em: http://restfb.com/.

[51] Facebook Developers. (2012, Jan.). *Graph API* [Em linha]. Disponível em: https://developers.facebook.com/docs/reference/api/.

[52] Yamamoto, Yusuke. (2012, Feb.). Twitter4J – A Java library for the Twitter API [Em linha]. Disponível em: http://twitter4j.org/en/index.html.

[53] Mukhtar, Nabeel. (2012, Feb.). linkedin-j: Project Summary [Em linha]. Disponível em: http://www.ohloh.net/p/linkedin-j.

[54] Mukhtar, Nabeel. (2012, Feb.). linkedin-j – A Java wrapper for LinkedIn APIs [Em linha]. Disponível em: http://code.google.com/p/linkedin-j/.

[55] Google Project Hosting (2012, Mar.). *Google Data Java Client Library* [Em linha]. Disponível em: http://code.google.com/p/gdata-java-client/.

[56] The Apache Software Foundation (2012, May). *Project Details for Apache Jena* [Em linha]. Disponível em: http://projects.apache.org/projects/jena.html.

[57] The Apache Software Foundation. (2012, May). *Apache Jena – The core RDF API* [Em linha]. Disponível em: http://jena.apache.org/documentation/rdf/.

[58] Jenkov, Jakob. (2012, Jun.). Java & XML Tutorial: The DOM Parser [Em linha]. Disponível em: http://tutorials.jenkov.com/java-xml/dom.html.

[59] w3schools.com. (2012, Jun.). *XML DOM Tutorial* [Em linha]. Disponível em: http://www.w3schools.com/dom/default.asp.

[60] Hégaret, Phillipe Le; Wood Lauren and Robie Jonathan. (2012, Jun.). What is the Document Object Model? [Em linha]. Disponível em: http://www.w3.org/TR/DOM-Level-2-Core/introduction.html.

[61] Dawson, F. (2012, Sep.). *vCard MIME Directory Profile* [Em linha]. Disponível em: http://www.ietf.org/rfc/rfc2426.txt.

[62] Halpin, Harry; Ianella, Renato; Brian Suda and Norman Walsh. (2012, Sep.). *Representing vCard Objects in RDF* [Em linha]. Disponível em: http://www.w3.org/TR/vcard-rdf/.

[63] Dublin Core Metadata Initiative. (2012, Sep.). *DCMI Metadata Terms* [Em linha]. Disponível em: http://dublincore.org/documents/dcmi-terms/.

[64] SIOC-Project. (2012, Sep.). *Semantically-Interlinked Online Communities* [Em linha]. Disponível em: http://sioc-project.org/.

[65] FOAF Project. (2012, Sep.). *The Friend of a Friend (FOAF) project* [Em linha]. Disponível em: http://www.foaf-project.org/.

[66] Media Annotations Working Group. (2012, Sep.). *Video, Audio, Images* [Em linha]. Disponível em: http://www.w3.org/2008/WebVideo/Annotations/.

[67] AIISO. (2012, Sep.). *Academic Institution Internal Structure Ontology* [Em linha]. Disponível em: http://vocab.org/aiiso/schema#.

[68] RDF Site Summary (RSS) 1.0. (2012, Sep.). *RDF Site Summary (RSS) 1.0* [Em linha]. Disponível em: http://web.resource.org/rss/1.0/spec.

[69] Class Scheme for Science Fields. (2012, Sep.). *Reference Description* [Em linha]. Disponível em: http://www.iwi-iuk.org/material/RDF/Schema/Class/scf.html.

[70] ResumeRDF Ontology Specification. (2012, Sep.). *Specification Document - 10 Jan 2007* [Em linha]. Disponível em: http://rdfs.org/resume-rdf/.

[71] Photo Access Control Project. (2012, Sep.). *Photo Access Control Project* [Em linha]. Disponível em: http://dig.csail.mit.edu/2008/PAC/doc/.

[72] The Open Graph protocol. (2012, Sep.). *The Open Graph protocol* [Em linha]. Disponível em: http://ogp.me/#.

[73] Facebook Developers. (2012, Jan.). *Authentication* [Em linha]. Disponível em: https://developers.facebook.com/docs/authentication/.

[74] Twitter Developers. (2012, Feb.). *Implementing Sign in with Twitter* [Em linha]. Disponível em: https://dev.twitter.com/docs/auth/implementing-sign-twitter.

[75] LinkedIn Developers. (2012, Feb.). *Authentication* [Em linha]. Disponível em: https://developer.linkedin.com/documents/authentication.

Anexos

Anexo A – Casos de uso

Como foi referido anteriormente, o sistema possui apenas um tipo de utilizador, pelo que não há necessidade de efetuar uma distinção entre as permissões obtidas pelo utilizador. Na Figura 48 estão representados os principais casos de uso identificados para um utilizador da aplicação desenvolvida.

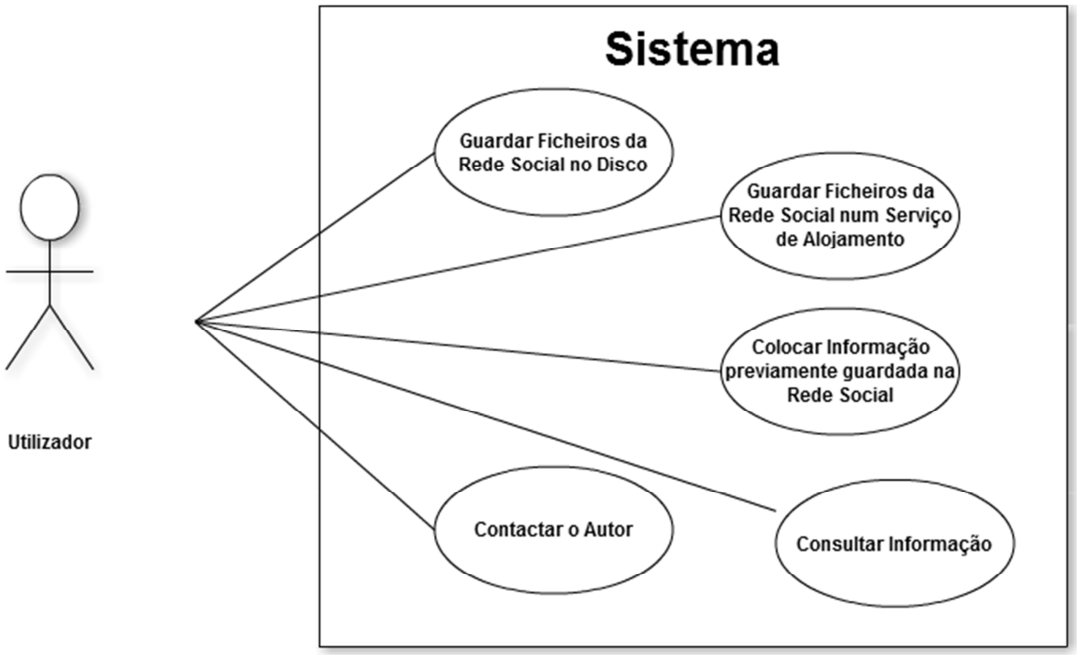

Figura 48 - Casos de uso do utilizador

Guardar Ficheiros da Rede Social no Disco

O utilizador, ao escolher a rede social, terá de efetuar os processos de autenticação e autorização descritos anteriormente. De seguida, quando surgem as diferentes opções disponíveis, o utilizador terá de selecionar a opção de guardar a informação no disco, sendo forçado a colocar uma localização válida para uma pasta no disco que pode ou não estar já criada. De seguida será necessário efetuar o *login* no serviço do Personal Digital Repository que será responsável por guardar os ficheiros RDF que contêm a informação dos conteúdos disponibilizados pelo utilizador na rede social.

Será então possível, para a aplicação, proceder à necessária interação, sendo o utilizador informado através de uma animação do processamento da informação. No final, será mostrada uma mensagem informando do sucesso, ou não, da interação.

Guardar Ficheiros da Rede Social num Serviço de Alojamento

Efetuando a interação anteriormente descrita, mas escolhendo a opção de guardar a informação num serviço de alojamento *online* será necessário, após este passo, efetuar mais um processo de autenticação e autorização, sendo de seguida possível, à aplicação, efetuar as necessárias interações. No final do processo de autenticação e autorização no serviço selecionado, caso este só suporte a receção de ficheiros multimédia, como acontece, por exemplo, com o Google Picasa, será necessário efetuar o *login* na Base de Dados NoSQL como aconteceu anteriormente.

Colocar Informação previamente guardada na Rede Social

Depois de escolhida a rede social, se for selecionada a opção de colocar a informação na rede social, é necessário proceder à identificação do utilizador na Base de Dados NoSQL, sendo, de seguida, necessário confirmar a intenção de efetuar a interação selecionada. Caso exista essa confirmação, o processo será iniciado, sendo mostrada uma animação como nos casos anteriores e, no final, a necessária mensagem informativa. Após esta mensagem as alterações já estarão disponíveis na rede social selecionada.

Contactar o Autor

Estará sempre disponível no menu superior a opção de Contactar o Autor do trabalho, que permitirá o envio de um e-mail usando, para isso, o cliente pré-definido pelo Sistema Operativo.

Consultar Informação

A qualquer momento estará, também, disponível no menu superior a opção de Consultar Informação, que permite consultar o âmbito do desenvolvimento deste projeto, bem como um pequeno guia de ajuda inicial ao utilizador.

Anexo B – Autenticação e autorização nas redes sociais usadas

Facebook

Como foi referido anteriormente, o processo de autenticação e autorização possibilita a obtenção da identidade de um utilizador, neste caso do Facebook e, permite ainda, ler e escrever dados através da API disponibilizada por esta rede social. A plataforma do Facebook usa OAuth 2.0 para a autenticação e autorização.

Uma autenticação correta resulta na obtenção de um *token* de acesso por parte da aplicação desenvolvida que poderá ser usado para proceder à interação com o Facebook.

Existem diversos fluxos de autenticação diferentes, como:

- Autenticação em aplicações Android nativas;
- Autenticação em aplicações iOS nativas;
- Autenticação em *Page Tab* no próprio Facebook;
- Autenticação em *Canvas Page* em apps.facebook.com;
- Autenticação para páginas *web* e aplicações móveis usando JavaScript (fluxo do lado do cliente);
- Autenticação para páginas *web* e aplicações móveis usando um servidor (fluxo do lado do servidor);
- Autenticação para dispositivos sem acesso a um browser.

Dadas as circunstâncias do desenvolvimento e as opções tomadas, optou-se por utilizar a autenticação para páginas *web* e dispositivos móveis usando JavaScript.

Por omissão, quando um utilizador permite o acesso de uma aplicação, este acesso apenas abrange a informação básica. Se for necessário o acesso a dados adicionais ou mesmo à escrita de informação no Facebook, torna-se necessário requisitar permissões adicionais, que têm de ser passados nos vários fluxos de autenticação usando um parâmetro denominado de *scope*.

No caso da aplicação desenvolvida, como se torna necessário aceder à totalidade da informação de um utilizador, bem como à possibilidade de escrita de informação, várias permissões adicionais têm de ser escolhidas. Estas permissões são as seguintes:

- **user_about_me**: Permite o acesso à informação disponibilizada no separador “Sobre”;
- **user_activities**: Permite o acesso às atividades em que o utilizador participa;
- **user_birthday**: Permite o acesso à data de nascimento do utilizador;

- **user_education_history**: Permite o acesso à informação escolar;
- **user_events**: Permite o acesso aos eventos em que o utilizador participa ou participou;
- **user_groups**: Lista de grupos de que o utilizador faz parte;
- **user_hometown**: Permite o acesso à naturalidade do utilizador;
- **user_interests**: Lista de interesses do utilizador;
- **user_likes**: Lista de *likes* que o utilizador fez;
- **user_location**: Localização atual do utilizador;
- **user_notes**: Acesso às notas adicionadas pelo utilizador;
- **user_photos**: Acesso às fotografias adicionadas pelo utilizador, bem como aquelas em que o mesmo foi identificado;
- **user_relationships**: Permite o acesso aos relacionamentos existentes na conta do utilizador, sejam familiares ou pessoais;
- **user_relationship_details**: Acesso aos detalhes dos relacionamentos;
- **user_religion_politics**: Acesso às afiliações políticas e religiosas do utilizador;
- **user_status**: Garante o acesso às mensagens publicadas pelo utilizador no seu mural;
- **user_videos**: Possibilita o acesso aos vídeos adicionados pelo utilizador, bem como aqueles em que o utilizador foi identificado;
- **user_website**: Permite o acesso a uma página *web* associada com a conta do utilizador, obtendo o URL correspondente;
- **user_work_history**: Permite o acesso ao histórico de trabalho associado com a conta de utilizador;
- **email**: Permite o acesso ao endereço de e-mail associado com a conta de Facebook.

Além das permissões referidas anteriormente, foi necessário utilizar, ainda, algumas permissões de um conjunto mais restrito, denominadas de *extended permissions*. Estas permissões poderão não ser aceites pelo utilizador, mas caso tal aconteça, algumas ações associadas com a interação das redes sociais não poderão ser concluídas. As permissões utilizadas que fazem parte deste conjunto foram:

- **read_mailbox**: Providencia o acesso à lista de mensagens privadas recebidas por parte do utilizador;

- **publish_stream**: Esta permissão é responsável por permitir que a aplicação escreva conteúdo, comentários e *likes* para uma conta de Facebook providenciada pelo utilizador.

Para conseguir utilizar a autenticação do Facebook foi, inicialmente, necessário criar uma aplicação na secção de desenvolvimento desta rede social. Ao criar uma aplicação são-nos dadas duas chaves, uma API Key e uma API Secret que são necessárias para o fluxo de interações que envolvem a autenticação e autorização.

Numa primeira fase, para iniciar a autenticação, terá de ser criado um URL formado por "http://www.facebook.com/dialog/oauth?client_id=", seguido da API Key disponibilizada na aplicação criada. De seguida terá de ser inserido o URI para onde a página, depois de efetuada a autenticação, deve apontar e, opcionalmente, as permissões necessárias para o correto funcionamento da aplicação. Por último deve adicionar-se um parâmetro "response_type=token". Todo este processo foi gerido via JavaScript de modo a possibilitar, de forma dinâmica, a interação com qualquer utilizador que use a aplicação. Na Figura 49 está um exemplo do fluxo de interações necessário para autenticar um utilizador.

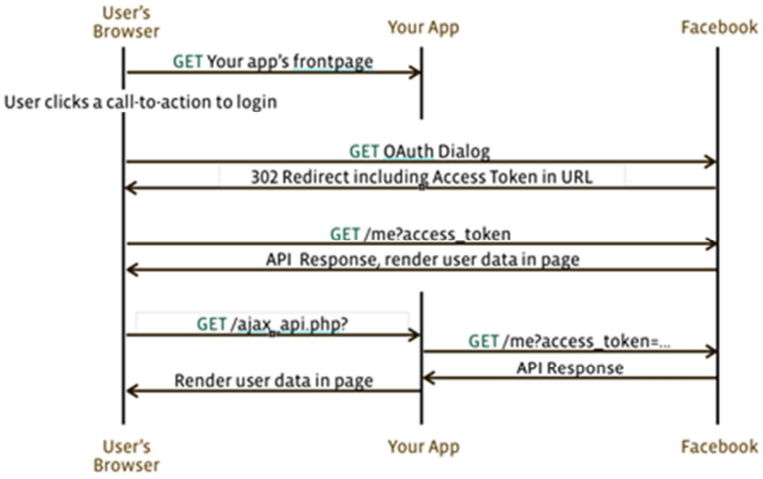

Figura 49 - Fluxo de interações para autenticar utilizador no Facebook [73]

Caso o utilizador não tenha ainda aceitado as permissões necessárias para a aplicação interagir com o Facebook, será confrontado com uma página como está representado na Figura 50. Terá, de seguida, de confirmar que aceita as permissões para que as interações necessárias possam decorrer. Se o utilizador já tiver aceitado anteriormente as permissões, não precisará de passar de novo por este processo.

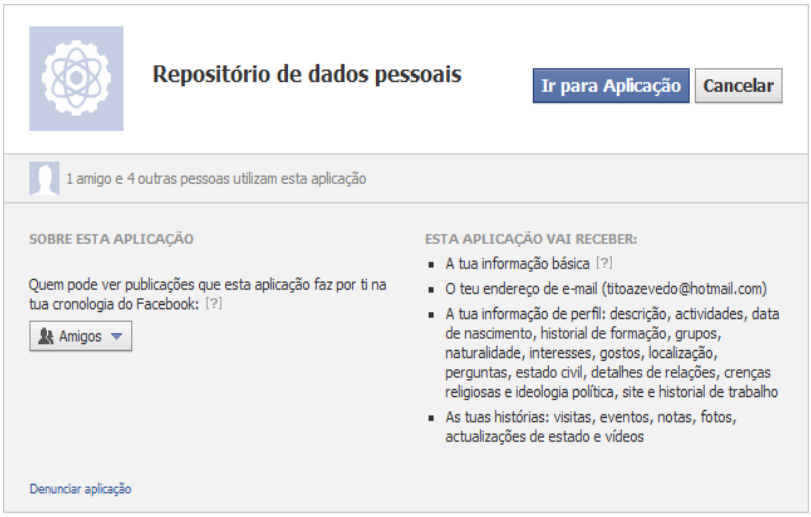

Figura 50 - Janela de autorização no Facebook

Um exemplo da janela que permite aceitar as permissões necessárias encontra-se na Figura 51.

Figura 51 - Janela com as permissões requisitadas pela aplicação

Caso o processo descrito anteriormente decorra de modo normal e o utilizador aceite as permissões necessárias, haverá um redireccionamento da página atual para o URI indicado anteriormente. Nesse URI irá também, como parâmetro, o *token* de autenticação. Desse modo passa a ser possível, usando o *token*, interagir com o Facebook, usando uma biblioteca, RestFB, que será descrita no capítulo referente às bibliotecas utilizadas [73].

Twitter

No caso do Twitter, o processo de autenticação e autorização é ligeiramente diferente.

Em primeiro lugar, torna-se necessário criar uma aplicação no próprio Twitter para que seja possível a interação com a página *web* desenvolvida, à imagem do que foi feito com o Facebook. Depois de registada a aplicação, é-nos dada uma *Consumer Key* e um *Consumer Secret* que serão usados no processo de autenticação.

Para obter um *request token*, a aplicação envia uma mensagem assinada para POST oauth/request_token. O único parâmetro necessário é o oauth_callback que vai definir a página para onde será feito o redireccionamento após ser completado o segundo passo desta interação. Neste processo são, ainda, adicionadas programaticamente as chaves obtidas no registo da aplicação que foram referidas anteriormente, de modo a possibilitar um conjunto correto de interações.

Na Figura 52 está representado o primeiro passo para a obtenção dos *tokens* de acesso ao Twitter, que foi descrito anteriormente.

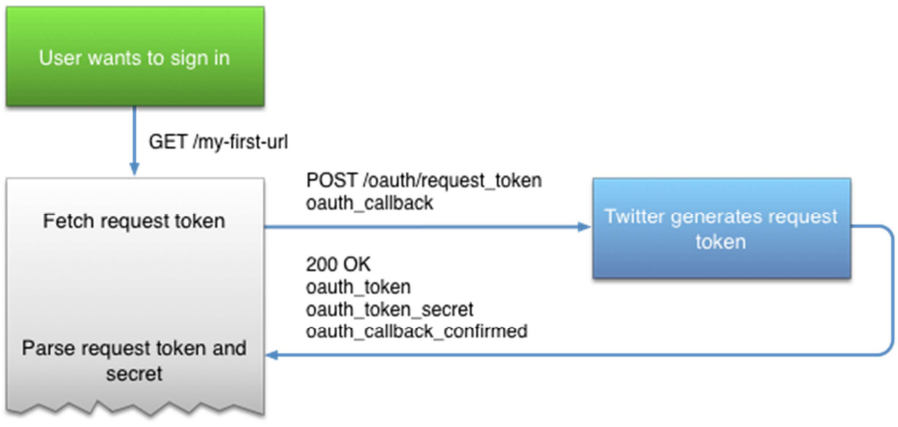

Figura 52 - Primeiro passo para autenticação no Twitter [74]

De seguida, a aplicação receberá dois *tokens*, o oauth_token e o oauth_token_secret, que deverão ser guardados para uma utilização futura.

No passo seguinte, o utilizador será direcionado para o Twitter de modo a autorizar a aplicação. Será, desta feita, efetuado um GET para oauth/authorize, passando o *request token* obtido no primeiro passo como parâmetro oauth_token. Caso o objetivo fosse apenas efetuar o *Sign In* do utilizador no Twitter, bastaria efetuar o GET para oauth/authenticate. Na Figura 53 está representado este segundo passo.

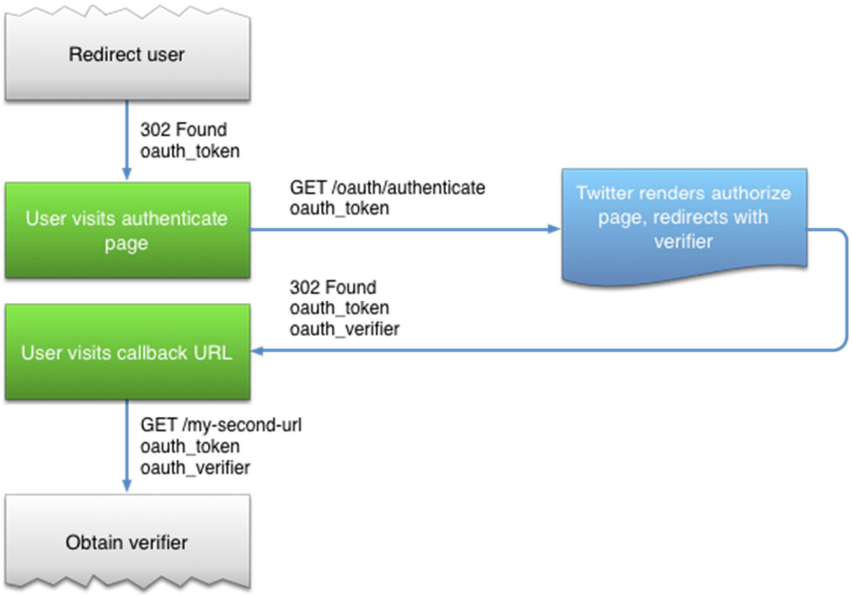

Figura 53 - Segundo passo para autorização no Twitter [74]

Caso este processo decorra com normalidade, serão recebidos dois parâmetros, um oauth_token que deverá ser igual ao usado no primeiro passo e um oauth_verifier que será usado para a obtenção dos *tokens* de acesso que permitirão a interação com o Twitter.

Um exemplo da janela que aparece neste passo está na Figura 54. Ao contrário do que acontecia com o Facebook, mesmo que o utilizador já tenha aceitado as permissões da aplicação, cada vez que efetuar este processo terá de o fazer de novo.

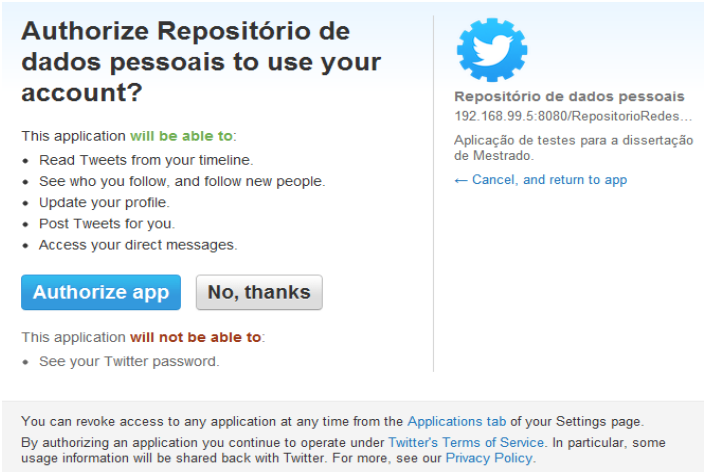

Figura 54 - Janela que permite aceitar as permissões no Twitter

Para o último passo, terá de ser feito um pedido para transformar o *request token* num *token* de acesso usável. Para isso será feito um POST para oauth/access_token contendo o oauth_verifier obtido anteriormente, passando ainda o *request token* como parâmetro oauth_token. Na Figura 55 está uma representação deste passo.

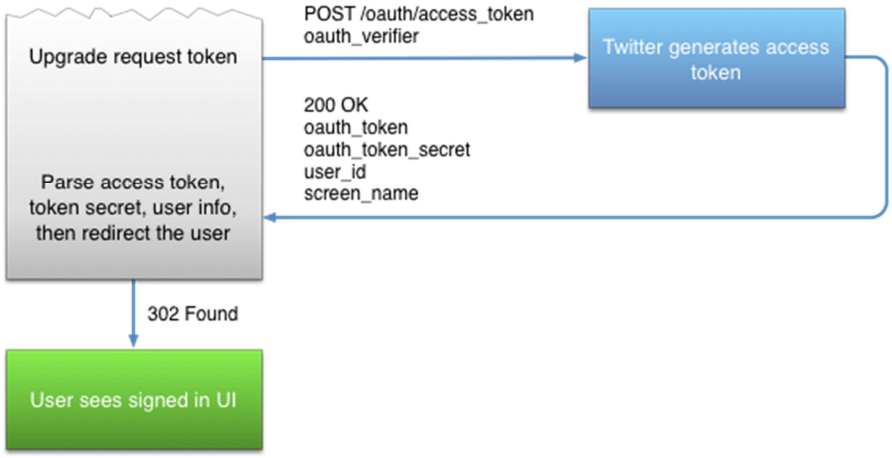

Figura 55 - Último passo para autenticação e autorização no Twitter [74]

Após este último passo, será possível obter os parâmetros oauth_token, oauth_token_secret, user_id e screen_name. Para a interação com o Twitter, guarda-se o oauth_token e o oauth_token_secret, que serão usados em conjunto com uma biblioteca destinada à integração da API do Twitter em Java, o Twitter4J, que será analisada na secção respetiva [74].

LinkedIn

O processo de autenticação e autorização do LinkedIn é bastante parecido com aquele usado no Twitter.

O LinkedIn cita três importantes aspetos que têm que ser garantidos quando uma aplicação interage com esta rede social. São os seguintes:

- A aplicação tem de conseguir identificar o membro;
- A privacidade do membro tem de ser protegida;
- O LinkedIn tem de ser capaz de saber qual a aplicação e qual o membro que efetua cada pedido.

Para conseguir garantir estes aspetos foi usado o OAuth de modo a dar às aplicações desenvolvidas acesso autorizado às APIs disponibilizadas pelo LinkedIn. Esta rede social usa o OAuth para negociar autorização e permitir o acesso a pedidos da API em nome de utilizadores do LinkedIn de modo rápido. Na Figura 56 é mostrada a interação normal entre o LinkedIn e o cliente, para os processos de autenticação e autorização.

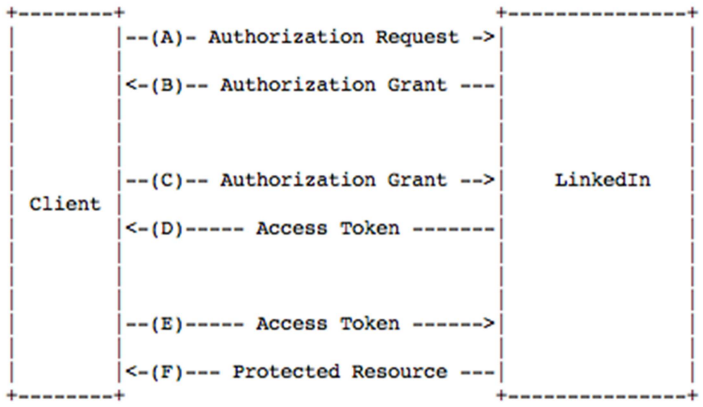

Figura 56 - Autenticação e autorização no LinkedIn [75]

Resumidamente, os fluxos OAuth usados no LinkedIn podem ser descritos do seguinte modo:

- Uma aplicação pede um conjunto de credenciais temporárias, conhecidas como *request token*. Nesta altura estas credenciais não estão associadas com nenhum utilizador específico;
- A aplicação redireciona o utilizador para uma janela de *login* onde autoriza essas credenciais temporárias de modo a que possam ser associadas com a sua conta de LinkedIn;

- A aplicação atualiza o *request token* para credenciais permanentes, também conhecidas como *access token*. Estas credenciais são necessárias para dar à aplicação acesso às APIs do LinkedIn e permite que se façam chamadas em nome do utilizador.

Tal como aconteceu nas redes sociais descritas anteriormente, também no LinkedIn foi necessário criar uma aplicação de modo a obter as chaves necessárias para iniciar o processo de autenticação e autorização, nomeadamente a API Key e a Secret Key.

Como foi referido, o primeiro passo é a obtenção de autorização por parte do utilizador. Usando as chaves obtidas após a criação da aplicação, é possível iniciar o fluxo de autorização, obtendo um *request token*. Neste processo é feito um POST para https://api.linkedin.com/uas/oauth/requestToken. Caso a aplicação pretenda que o utilizador aceite permissões adicionais deve, como em casos anteriores, passar essas permissões através de um parâmetro *scope*.

A partir do momento em que é obtido o *request token*, torna-se possível pedir permissão ao utilizador para efetuar chamadas à API em seu nome. Desse modo, a abordagem tradicional passa por redirecionar a página para o URL de autenticação, passando o oauth_token como parâmetro, de modo a possibilitar a identificação. O utilizador será confrontado com uma janela de autenticação onde pode escolher dar acesso à aplicação e, simultaneamente, ver que tipo de permissões é que a aplicação deseja obter. Assim que o utilizador autorize o acesso, o LinkedIn redirecionará a página de volta para a aplicação, passando um parâmetro com o nome oauth_verifier que actua como um PIN. Para efetuar este passo, será feito um POST para https://www.linkedin.com/uas/oauth/authenticate?oauth_token=XXXXX, onde "XXXXX" representa o *token* obtido anteriormente pelo utilizador.

Um exemplo da janela descrita anteriormente pode ser visto na Figura 57.

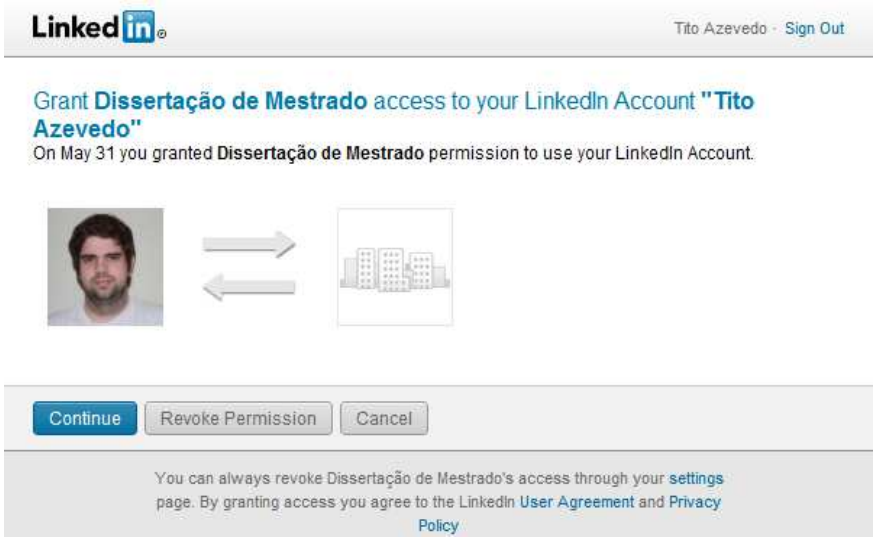

Figura 57 - Janela de permissões do LinkedIn

Através do *verifier* obtido, tal como acontecia no Twitter, passa a ser possível interagir com a rede social.

Com a API do LinkedIn existe a possibilidade de definir quais as permissões que o utilizador deve dar à aplicação. Estas permissões providenciam o seguinte:

- **Perfil básico por omissão:** Quando não são definidas permissões, será usado, por omissão, o perfil básico, que possuirá as permissões de acesso básico à conta do utilizador;
- **Tudo ou nada:** Os utilizadores terão de aceitar todas as permissões pedidas pela aplicação. Não é possível aceitar permissões individuais;
- **Otimizado para 3 permissões:** A experiência de utilizador foi otimizada para aplicações que peçam até 3 permissões. Pedir permissões adicionais é permitido, mas não é encorajado;
- **Reautenticação ou alteração de permissões:** Uma nova janela de *login* será mostrada ao utilizador caso haja uma alteração de permissões.

Na Tabela 10 estão representadas as diferentes permissões que a aplicação pode pedir ao utilizador.

Permissão	Descrição	*Scope*
Visualização do Perfil	Nome, foto, título principal e posições atuais	r_basicprofile
Perfil Completo	Perfil completo incluindo experiência, educação, competências e recomendações	r_fullprofile
Endereço de e-mail	O endereço de e-mail principal associado à conta do LinkedIn	r_emailaddress
Ligações	As ligações de 1º e 2º grau	r_network
Contacto	Endereço, número de telefone e contas ligadas	r_contactinfo
Atualizações da rede	Obter e colocar atualizações no LinkedIn em nome do utilizador	rw_nus
Discussão de grupo	Obter e colocar discussões de grupo em nome do utilizador	rw_groups
Convites e mensagens	Enviar mensagens e convites para ligação em nome do utilizador	w_messages

Tabela 10 - Permissões disponíveis no LinkedIn [75]

Como se pode ver, os processos de autenticação e autorização do LinkedIn têm alguns pontos em comum com os processos do Twitter. Tal como acontecia no Twitter, após a obtenção do *verifier* torna-se possível obter o *token* de acesso que permitirá a interação com a rede social. No caso do LinkedIn, optou-se por usar uma biblioteca de integração das APIs desta rede social em Java, com o nome linkedin-j e que será descrita no capítulo dedicado às bibliotecas usadas [75].

Anexo C – Navegação na aplicação desenvolvida

Continuando a descrição iniciada no Capítulo 4, depois de estarmos na página inicial, carregando no botão de Contacto é aberta uma janela para o envio de uma mensagem nova com o cliente de e-mail pré-definido pelo utilizador. Carregando no botão Informação será apresentada a página visível na Figura 58.

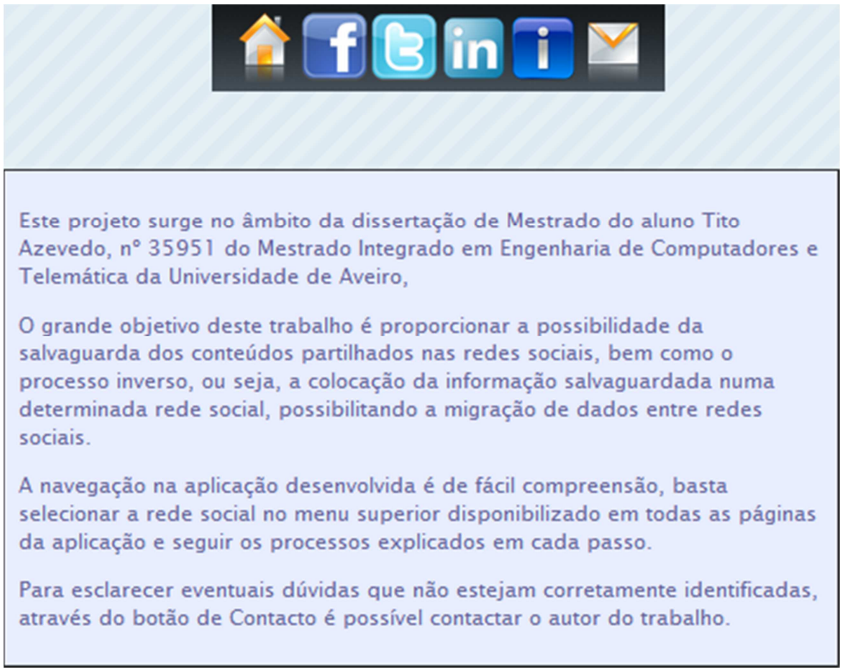

Figura 58 - Informação da aplicação desenvolvida

Escolhendo, para esta demonstração, a opção Twitter, é apresentada a página visível na Figura 59. Nesta página é visível um aviso relativamente ao processo que vai decorrer, de modo a que o utilizador esteja perfeitamente consciente do que acontecerá de seguida.

Figura 59 - Página mostrada após seleção da opção Facebook

Depois de o utilizador continuar o processo, caso não tenha iniciado sessão no Twitter, terá de o fazer (Figura 60).

Figura 60 - Iniciar sessão no Twitter

Ao contrário do que acontece no Facebook, ao efetuar o *login* neste ecrã o utilizador está, automaticamente, a aceitar as permissões necessárias para as interações que serão feitas.

Caso o utilizador efetue o *login*, aceitando desse modo as permissões, será possível continuar o processo, sendo mostrada a página visível na Figura 61 com as diferentes opções disponíveis para interagir com a rede social selecionada. Mais uma vez optou-se

por usar um menu animado, para proporcionar uma agradável utilização, mas também com texto a identificar cada serviço, para evitar potenciais enganos e/ou confusões.

Figura 61 - Opções de interação com a rede social

Caso o utilizador escolha a opção de guardar os ficheiros multimédia provenientes do Twitter no disco, será mostrada uma janela para a inserção da pasta onde o utilizador deseja guardar os ficheiros, como é visível na Figura 62. Nesta janela é feita uma verificação do que é inserido pelo utilizador, através da utilização de expressões regulares de modo a garantir que é colocada uma localização para uma pasta válida.

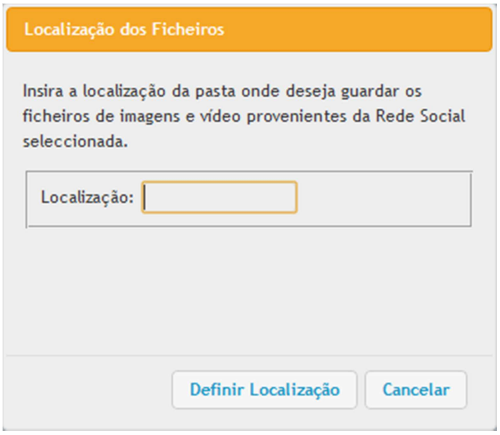

Figura 62 - Janela para a inserção da localização da pasta

Neste caso, para demonstrar, também, a interação com um serviço de alojamento *online*, é escolhida a opção de colocar a informação no Google Picasa. Depois de escolhida a opção é mostrada a página visível na Figura 63, informando o utilizador de que terá de efetuar o *login* na sua conta. Caso o utilizador não tenha uma conta criada,

terá de o fazer, pois o Google Picasa é um serviço que necessita de uma conta Google associada.

Figura 63 - Informação do processo de *login* na conta Google

Neste caso, depois de iniciar sessão na conta Google, o utilizador será levado para uma página de autorização do Google Picasa à aplicação (Figura 64).

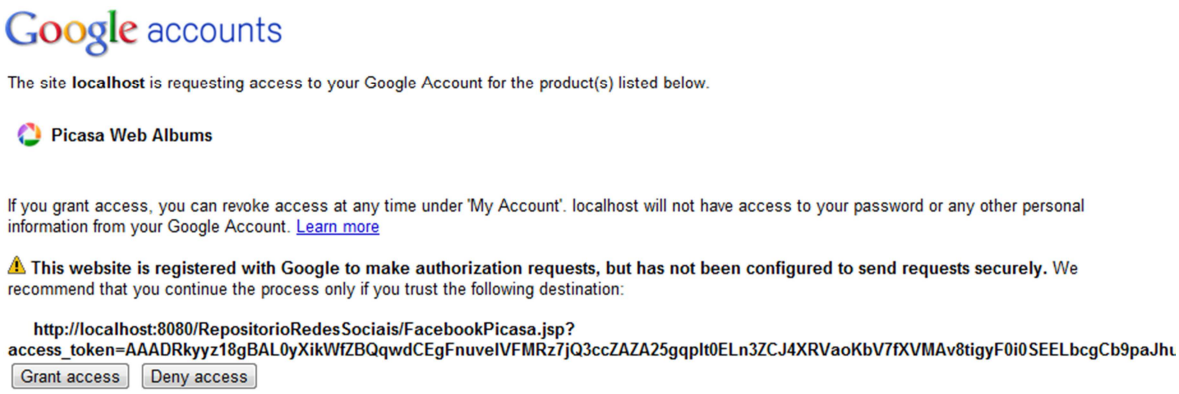

Figura 64 - Autorização do Google Picasa

Depois de ser dado o acesso, aparecerá uma nova janela que explica o processo seguinte, já que a interação com a rede social, dependendo da quantidade e tamanho da informação partilhada, pode ser demorada e, para evitar que o utilizador cancele o processo por achar que o mesmo não está a ser bem-sucedido, torna-se necessário dar uma explicação do processo (Figura 65).

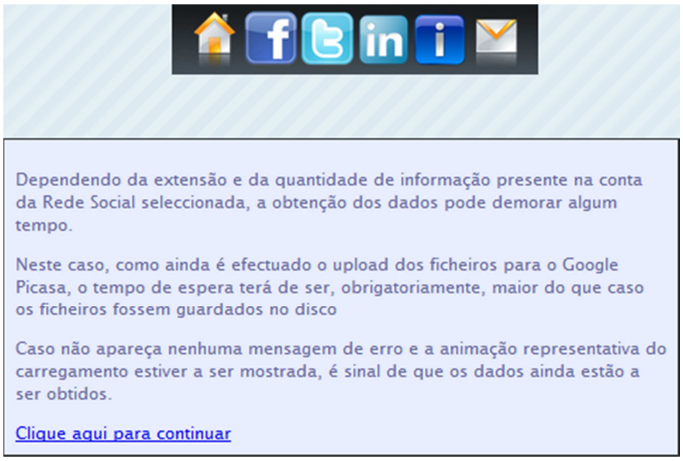

Figura 65 - Introdução à interação com a rede social

Caso o utilizador esteja de acordo com a introdução feita e decida prosseguir com o processo, carregando na ligação existente, será mostrada uma animação dando conta do processamento da informação, de modo a que o utilizador possa ter um *feedback* do que está a ser feito (Figura 66).

Figura 66 - *Feedback* da interação com a rede social

No fim deste processo, caso o mesmo decorra com sucesso, será apresentada a página visível na Figura 67. Caso existe algum erro no processo, será mostrada a mesma página ao utilizador, mas com a mensagem de erro e uma explicação do que se sucedeu, de modo a que o utilizador possa corrigir o problema, caso seja um problema seu, ou perceber o que houve de errado. Por exemplo, na Figura 68 é mostrada a mensagem de erro quando se tenta colocar a informação obtida do Twitter numa pasta onde o utilizador não tem quaisquer permissões.

Figura 67 - Sucesso na interação com a rede social

Figura 68 - Erro na interação com a rede social

Exemplificando agora o processo inverso, serão mostradas imagens da colocação de informação previamente obtida de uma conta para outra. Para efetuar este exemplo, bem como para efetuar os diversos testes necessários ao longo do desenvolvimento do projeto, foram criadas contas de teste, usadas para garantir o funcionamento correto das funcionalidades desenvolvidas. Escolhendo no menu superior que está presente em todas as páginas a opção Facebook será necessário proceder à autenticação, caso o utilizador ainda não o tenha feito, como é visível na Figura 69.

Figura 69 - Iniciar sessão no Facebook

De seguida será mostrada a página presente na Figura 70 que indica ao utilizador o tipo de permissões de que a aplicação necessita, podendo o utilizador avançar com o processo ou cancelá-lo.

Figura 70 - Janela com as permissões necessárias pela aplicação

Caso o utilizador opte por avançar com o processo, aparecerá a página representada na Figura 71 com a lista detalhada das permissões necessárias pela aplicação.

Figura 71 - Lista detalhada de permissões para a aplicação

No passo seguinte o utilizador terá uma janela com opções semelhante ao caso anteriormente analisado, mas desta vez terá de escolher a opção Facebook, de modo a proporcionar um envio de informação para esta rede social. Esta escolha está representada na Figura 72.

Figura 72 - Escolha da opção Facebook

De modo a evitar potenciais erros, como o *upload* de informação de uma rede social efetuado de modo incorreto, como identificado nas limitações encontradas descritas no capítulo seguinte, o utilizador terá de encontrar uma nova página com opções disponíveis para escolha, desta feita para escolher qual a origem dos dados que deseja colocar no Facebook. Esta janela está presente na Figura 73.

Figura 73 - Escolha da rede social original

No caso escolhido, como o utilizador fará a migração de uma conta de Twitter para uma de Facebook, nesta janela será escolhida a opção Twitter. De seguida será apresentada a janela presente na Figura 74, dando conta do processo que vai ser efetuado. Ao carregar na ligação para continuar com o processo será mostrada uma animação semelhante à da Figura 66 até à obtenção dos resultados pretendidos ou, eventualmente, do surgimento de um erro, sendo mostrada uma janela à semelhança do que aconteceu nas Figuras 67 e 68.

Figura 74 - Introdução ao processo de colocação de informação

Na Figura 75 está um exemplo do aspeto da conta de Facebook depois de colocada a informação obtida de outra conta. Neste exemplo é visível a colocação da informação pessoal através de uma nota que contém toda a informação. Como esta informação está visível a qualquer amigo do utilizador, não existe problema na publicação destes dados. É também visível a existência de mensagens com imagens associadas ou de simples mensagens textuais. Todas estas mensagens contêm a informação relativa à mensagem original como a data em que foram publicadas e o autor das mesmas. É visível que adicionou a imagem definida como imagem de perfil do Twitter a um álbum e a disponibilizou, embora não a tenha definido como imagem de perfil nesta conta, já que tal pode não ser desejado. Como o Facebook não suporta a alteração programática dos dados pessoais de um utilizador, estes foram passados através de uma nota apenas visível ao próprio utilizador. A mesma abordagem foi seguida para a salvaguarda das mensagens privadas, dado que, para evitar *spam*, o Facebook também não permite o envio de mensagens privadas através da sua API, obrigando a este tipo de soluções. Por último, também a informação das pessoas que o utilizador segue foi colocada numa mensagem apenas visível ao utilizador, dado que pode não querer que esta informação seja pública.

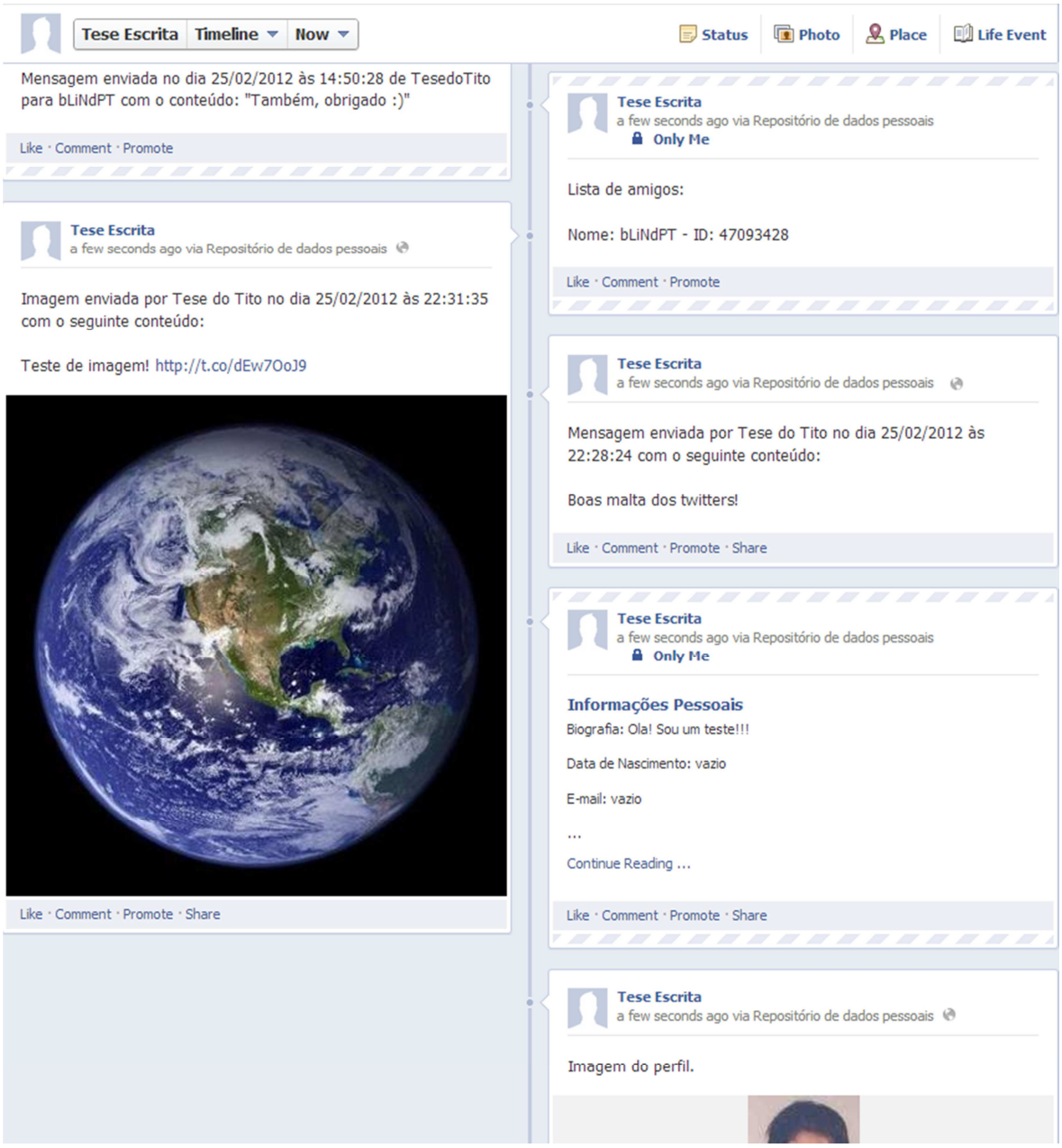

Figura 75 - Aspeto da página de Facebook depois da interação

Sempre que o utilizador utiliza o Repositório Digital Pessoal, seja para colocar nele os documentos RDF produzidos com a informação obtida na rede social, seja para receber esses documentos previamente guardados, terá de efetuar mais um processo de autenticação e autorização. Na Figura 76 representa-se a janela que o utilizador encontra quando utiliza este serviço.

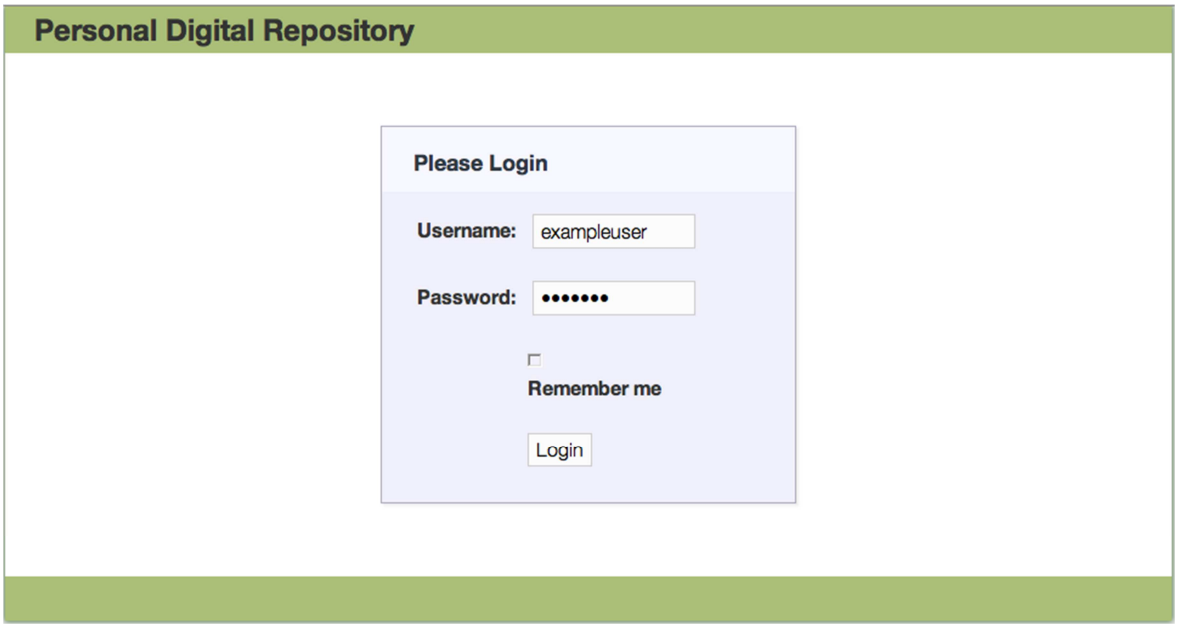

Figura 76 - Autenticação e Autorização no Repositório Digital Pessoal