Mestrado em Engenharia Informática
Estágio
Relatório Final

Bundlr: mecanismos de interacção social, pesquisa e exploração de conteúdo

Pedro Gaspar
pgaspar@student.dei.uc.pt

Orientadores:
Professor Doutor Luís Silva
Engenheiro Sérgio Santos

Data: 12 de Julho de 2012

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

Nos últimos anos, o aparecimento de serviços de publicação de conteúdo online provocou um crescimento no número de ferramentas de selecção, agregação e organização desse conteúdo. Neste estágio pretende-se desenvolver funcionalidades no Bundlr, uma dessas ferramentas, de forma a fomentar a sua utilização recorrente e a pesquisa e descoberta de novo conteúdo. Neste documento analisa-se a ferramenta tendo em conta outras semelhantes, faz-se um estudo da sua arquitectura, define-se a metodologia de desenvolvimento, planeia-se o trabalho e descrevem-se os módulos implementados. As temáticas que mais se destacam nesses módulos passam pela subscrição e visualização de conteúdo numa *Activity Stream*, pela criação de sugestões de conteúdo personalizadas e por funcionalidades de pesquisa *Full-Text*. Fazem-se também breves análises aos problemas de *Keyword Extraction* (apelidado de *Tag Extraction* neste documento) e de *Boilerplate Removal*.

Palavras-chave activity stream, categorização de conteúdo, curadoria, desenvolvimento web, full-text search, pesquisa, ruby on rails, sugestão de conteúdo, web social

Índice

Lista de Tabelas

Lista de Figuras

Lista de Acrónimos

API Application Programming Interface

CRUD Create Read Update Destroy

CSS Cascading Style Sheets

DNS Domain Name System

HTTP HyperText Transfer Protocol

IP Internet Protocol

HTML HyperText Markup Language

MVC Model-View-Controller

MVP Minimum Viable Product

ODM Object-Document Mapping

ORM Object-Resource Mapping

RDF Resource Description Framework

REST Representational State Transfer

RSS RDF Site Summary

URL Uniform Resource Locator

XML Extensible Markup Language

Capítulo 1

Introdução

Com a explosão de serviços de publicação de conteúdo online nos últimos anos houve também um grande aumento no número de ferramentas de selecção, agregação e organização desse conteúdo[1]. No entanto, ainda não houve nenhuma que se destacasse por ter aceitação pelo público em geral, da mesma forma que o Facebook ou o Google têm. Um dos principais desafios para qualquer uma destas ferramentas de selecção é a retenção de utilizadores, ou seja, aumentar o tempo que um utilizador despende no serviço.

Neste estágio pretende-se desenvolver funcionalidades num destes serviços de selecção, o Bundlr[1], com o intuito de fomentar a sua utilização recorrente e de facilitar a pesquisa e descoberta de novo conteúdo por parte dos utilizadores.

O planeamento dessas funcionalidades é feito através de módulos de trabalho que reflectem os objectivos práticos do estágio.

O processo de trabalho foi definido através de uma metodologia de desenvolvimento adaptada do Scrum desenvolvida especificamente para este projecto, principalmente por haver apenas um elemento na equipa. É também aplicada uma metodologia de teste para validar e melhorar a qualidade do software.

Durante o primeiro semestre fez-se um estudo do estado da arte a nível de soluções semelhantes ao Bundlr e implementaram-se as funcionalidades pagas e o sistema de subscrição de conteúdo. No segundo semestre introduziram-se funcionalidades de descoberta, categorização, sugestão e pesquisa de conteúdo.

O presente documento começa por fazer uma introdução ao Bundlr e aos objectivos do estágio. Segue-se o estudo e descrição dos aspectos mais importantes da arquitectura original da aplicação. Finalizada a análise do

[1] http://bundlr.com

serviço em questão, descrevem-se metodologias do estágio, o seu planeamento e apresenta-se o trabalho desenvolvido. Finalmente, apresentam-se as conclusões a que se chegou.

Os apêndices do documento contêm uma análise das formas mais comuns de resolver os problemas de *Tag Extraction* (apêndice I) e *Boilerplate Removal* (apêndice J), assim como um estudo comparativo de soluções para estes problemas (apêndice K). Inclui-se também uma análise mais profunda das alterações feitas em cada módulo a nível dos modelos e controladores do padrão MVC (apêndice E).

1.1 Bundlr

O Bundlr é uma ferramenta de selecção e organização de conteúdo online. Lançado de forma privada em Fevereiro 2011 e ao público em Junho do mesmo ano, é um serviço recente. A equipa de duas pessoas adoptou uma metodologia ágil de desenvolvimento de software, reunindo esforços para o lançamento de um *Minimum Viable Product*[2] em Fevereiro de 2011, estando desde então a melhorar vários aspectos do serviço e a tentar validar o modelo de negócio.

O principal foco do serviço é permitir aos utilizadores seleccionar, agregar e distribuir conteúdo em páginas temáticas, manualmente. O utilizador pode criar vários bundles, que são colecções temáticas de conteúdo. Depois de criar um bundle o utilizador poderá seleccionar conteúdo para lhe adicionar, na forma de clips. O serviço suporta vários tipos de clips, como vídeos do YouTube e do Vimeo, apresentações do Slideshare, entradas na Wikipedia, imagens e texto de qualquer website, entre outros. Em termos de descoberta do conteúdo agregado no serviço, os utilizadores têm acesso à pesquisa de bundles e de utilizadores, sendo apresentadas na *homepage* do serviço listas dos bundles mais populares no momento e dos bundles seleccionados pela equipa do Bundlr.

A figura 1.1 apresenta a página do bundle "WWDC 2012". Nesta, pode-se ver a grelha dos clips com conteúdo de imagens, vídeos e texto. Como se pode ver no canto superior direito, esta página temática pertence a um utilizador e, neste caso, foi composta com a ajuda de quatro colaboradores.

A figura 1.2 mostra a página de um utilizador, que lista todos os seus bundles. Quando o utilizador tem a sessão iniciada, esta é a sua página principal. Além de poder aceder aos seus bundles, pode criar novos e editar as suas definições.

A criação de clips é feita através de uma extensão do *browser*, que o utilizador activa quando está na página com o conteúdo que pretende guardar.

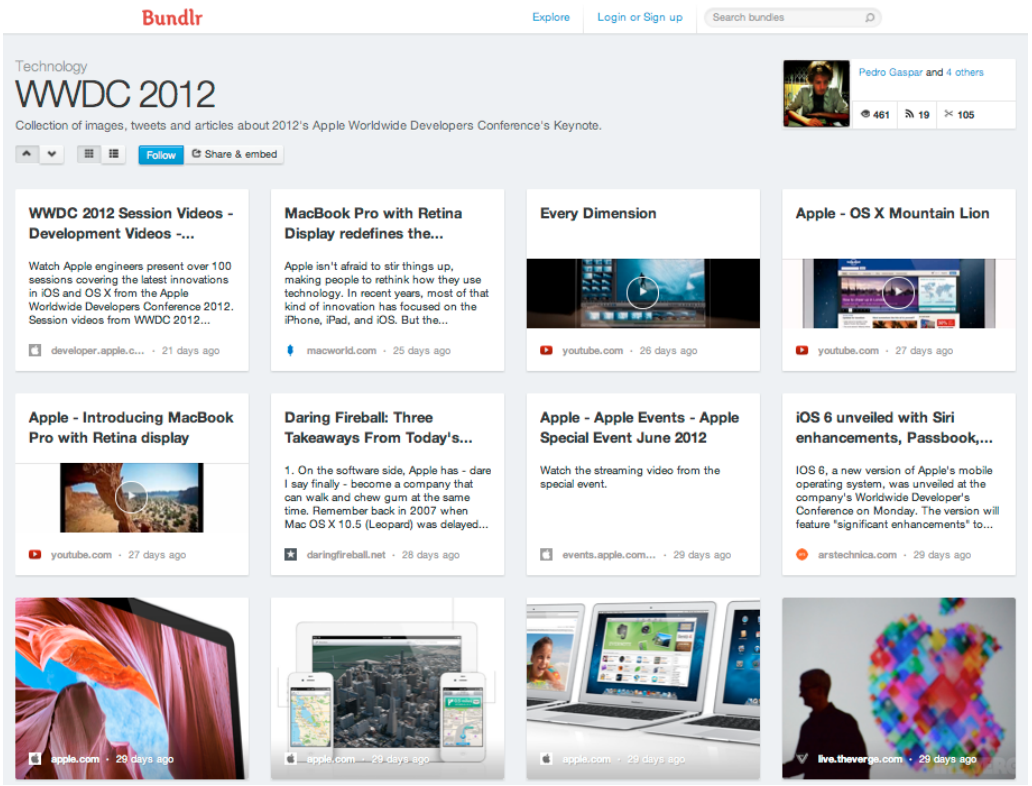

Figura 1.1: Página do bundle WWDC 2012, recolhida em Julho de 2012.

Antes de criar o clip, o utilizador tem de seleccionar qual o bundle onde o quer colocar.

Em termos de adesão, o serviço juntou até ao final de 2011 cerca de 7000 utilizadores registados, dos quais 20% utilizaram o serviço no mês de Dezembro. O público-alvo do serviço, inicialmente profissionais de informação, passou a ser estudantes, equipas de trabalho e pequenas empresas que trabalhem com informação web.

No Apêndice F inclui-se a análise do Bundlr no espectro das soluções mais conhecidas de agregação de *links* e conteúdo online, inicialmente apresentada no Relatório Intermédio.

1.2 Objectivos do Estágio

Do ponto de vista prático, os objectivos deste estágio são os seguintes:

- Desenvolver funcionalidades que farão parte dos planos pagos - bundles privados e limitação da colaboração;

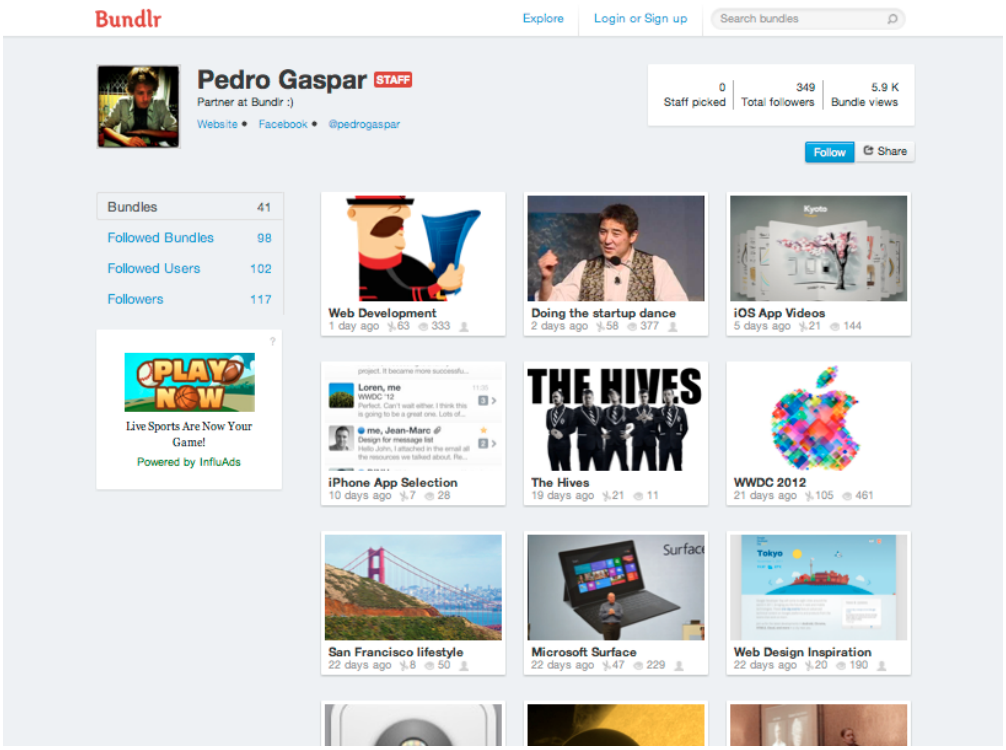

Figura 1.2: Página de utilizador, recolhida em Julho de 2012.

- Permitir aos utilizadores subscrever conteúdo produzido por outros utilizadores e em bundles específicos;
- Implementar mecanismos de exploração, descoberta e sugestão do conteúdo do Bundlr;
- Melhorar a pesquisa existente no serviço, implementando *Full-Text Search*.

Além destes, existem também objectivos relacionados com a correcta construção de um produto de software, isto é, os objectivos do ponto de vista de Engenharia:

- Cumprir os requisitos identificados nos módulos de trabalho identificados na tabela 4.1;
- Implementar *Acceptance Stories* para cobrir a totalidade dos requisitos dos módulos da tabela 4.1;

Dada a importância da definição do âmbito do estágio, apresentam-se de seguida alguns aspectos que não fazem parte dos objectivos do estágio e que, portanto, não serão abordados:

- A implementação do *front-end* a nível de design e de usabilidade;
- A elaboração e execução de testes não-funcionais a funcionalidades não implementadas e já existentes no sistema.

Capítulo 2

Arquitectura

A arquitectura da aplicação é a de um serviço web comum, contendo como principais componentes o Servidor e a Base de Dados. Além destes componentes básicos, incluem-se alguns serviços adicionais que apoiam as funcionalidades prestadas pela aplicação. Ao longo deste capítulo serão analisados esses vários componentes, tal como estavam antes do início deste estágio.

O esquema 2.1 apresenta uma visão geral da arquitectura da aplicação. Os componentes que estão marcados a cinzento foram adicionados ou alterados durante este estágio, e serão abordados no Capítulo 5. Os serviços externos como o servidor MongoDB serão descritos no presente capítulo.

No Apêndice E.1 encontram-se diagramas que descrevem o estado original do sistema em maior detalhe, sob o prisma da arquitectura MVC.

2.1 Servidor e *framework*

Ruby on Rails[1], uma *framework* de aplicações web lançada em 2004 por David Heinemeier Hansson[3] usando a linguagem de programação Ruby, é a *framework* usada pelo Bundlr. Como a maioria das *frameworks* deste tipo, baseia-se no padrão de desenho de software Model-View-Controller (MVC).

Utilização de uma *framework*

As vantagens da utilização de uma *framework* passam pela diminuição do tempo de desenvolvimento, já que estas incluem à partida funcionalidades que são usadas na maioria das aplicações web. Há também um aumento da segurança e da qualidade do sistema, já que o código introduzido pelas

[1] http://rubyonrails.org

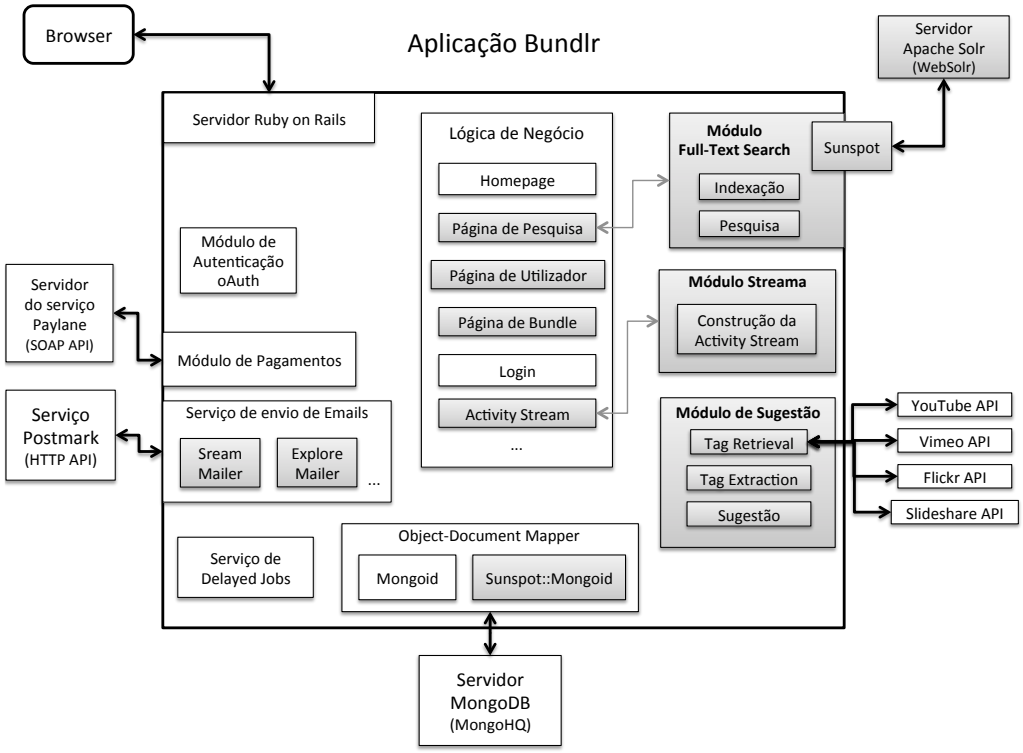

Figura 2.1: Diagrama simplificado da arquitectura da aplicação Bundlr. A cinzento estão marcados os componentes que foram adicionados ou alterados durante este estágio.

frameworks é testado em ambiente de produção por uma comunidade vasta de programadores que identifica rapidamente quaisquer falhas.

Por outro lado, as desvantagens são a menor flexibilidade para desenvolver código que vá contra o padrão MVC e alguma perda de performance, já que a solução final deixa de ser feita à medida para o problema em mãos. No entanto, mesmo essa perda de performance pode ser atenuada por estratégias de *caching*, entre outras. Relativamente a questões de escalabilidade, é sempre necessária, a certa altura, a intervenção do programador, já que cada aplicação tem necessidades diferentes e é sempre necessário fazer algum trabalho de adaptação da *framework*. Este tipo de situação levou algumas aplicações muito utilizadas a adoptarem soluções mais adequadas aos seus problemas específicos, como foi o caso do Twitter que mudou a sua infra-estrutura base para uma *framework* própria baseada em Scala, por facilitar o desenvolvimento em ambientes concorrentes, em 2009[4].

O Ruby on Rails é uma *framework* MVC popular[5], sendo suportada

por uma grande comunidade de programadores. Esta grande adesão significa que, através dos *plugins* desenvolvidos pela comunidade, é fácil encontrar extensões da *framework* para adicionar funcionalidades comuns a aplicações web. Além disso, o Ruby on Rails tem suporte interno para outras tarefas mais avançadas, como a utilização de múltiplas ORMs, sistemas de validação de dados e diferentes arquitecturas de testes de sistema.

A figura 2.2 apresenta uma simplificação da arquitectura interna de um servidor Ruby on Rails, sob o prisma do padrão MVC. Os componentes base de dados e ODM serão abordados na secção 2.2.

Alojamento - Heroku

O Heroku[2] é uma plataforma de alojamento que integra directamente com a *framework* Ruby on Rails. Integrando de forma transparente com o sistema de controlo de versões Git, facilita bastante o processo de envio de novas versões do código para produção.

A infraestrutura do Heroku é constituída por um conjunto de servidores quem contêm vários *dynos*[6], que correspondem a processos nas máquinas disponíveis. Os servidores partilham recursos tais como servidores de *caching* e a própria base de dados e proíbem qualquer tipo de operação de escrita local, permitindo-as apenas nos recursos partilhados. Isto torna possível associar uma aplicação a mais do que um *dyno* em tempo-real, permitindo assim que o administrador ajuste o número de processos conforme as necessidades da aplicação naquele momento. Cada *dyno* contém um servidor leve e rápido, já que não tem de lidar com sincronização de recursos e pedidos e por gerirem apenas um processo e uma *thread*, diminuindo assim os tempos de latência.

Esta arquitectura permite que a aplicação escale horizontalmente, através da adição de mais *dynos* para suportar períodos de maior carga, desde que os tempos de resposta da aplicação sejam baixos. O serviço também disponibiliza algumas ferramentas que diminuem a latência total, como soluções de *caching* ao nível da arquitectura.

[2] http://heroku.com

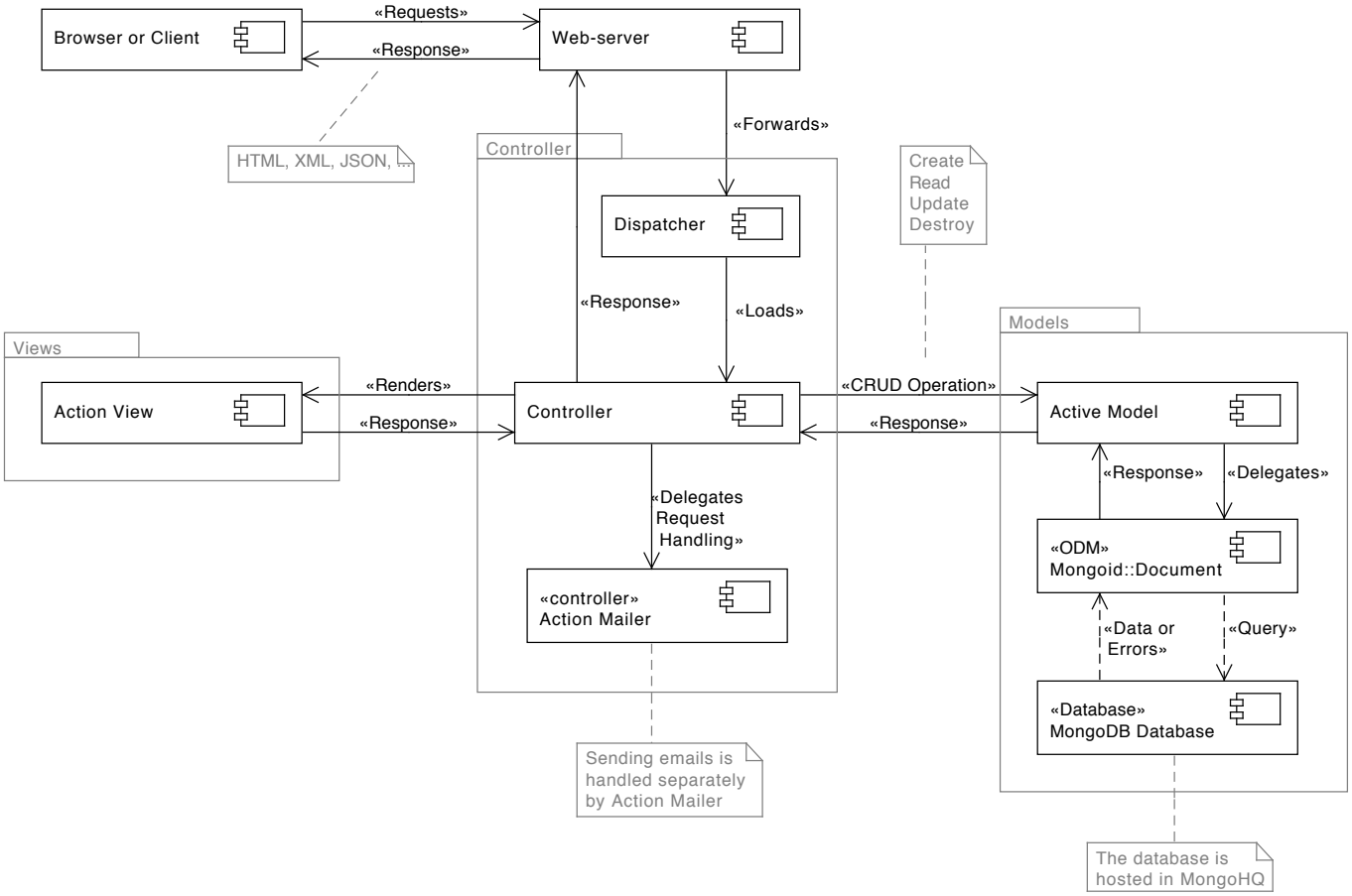

Figura 2.2: Diagrama de Componentes que representa de forma simplificada a divisão dos componentes do Ruby on Rails no modelo MVC.

As vantagens de utilizar um serviço como o Heroku em vez de adquirir fisicamente um servidor, ou mesmo alugar um servidor virtual privado, são o facto de não ser necessário fazer qualquer tipo de esforço de gestão do software instalado no servidor, já que, à excepção da própria aplicação, o restante software é da responsabilidade do serviço. Além disso, os servidores do Heroku estão geograficamente próximos dos servidores que alojam a base de dados, como é descrito na página 12 da secção que se segue.

Por outro lado, existem desvantagens no que toca à configurabilidade do servidor, que é reduzida. A título de exemplo, no Heroku é impossível alterar as definições de mapeamento ao nível do servidor, o que quer dizer que, por exemplo, todos os redireccionamentos têm de ser feitos ao nível da aplicação.

O Bundlr encontra-se alojado neste serviço em duas configurações: *stage* e produção. O primeiro ambiente serve como aplicação de testes privada, na qual são testadas as funcionalidades antes da passagem para o ambiente de produção, onde está a aplicação real.

2.2 Base de Dados

A base de dados é um dos principais componentes da arquitectura de uma aplicação web, já que possibilita o armazenamento do seu estado e dos seus dados. A que é utilizada pelo Bundlr é o MongoDB[3], uma base de dados não-relacional orientada a documentos.

Bases de Dados não-relacionais

Comparando com as base de dados relacionais, as mais comuns, uma não-relacional não exige a utilização de um esquema pré-definido. Esta opção faz sentido no contexto do serviço que se está a analisar, já que existe uma necessidade de suportar diferentes formatos de conteúdo, nomeadamente na formalização dos clips. Um clip pode ser tanto um link, como uma foto, como um vídeo, como uma série de outros tipos de conteúdos, o que iria significar a criação de tabelas de esquema bem definido (com todos os atributos previstos) para cada um destes tipos de clips, se se optasse pela opção de uma base de dados relacional. Qualquer alteração ou adição de um tipo de clip iria também exigir mudanças no esquema das tabelas afectadas.

Dentro das bases de dados não-relacionais existem três modelos principais:

- Chave-valor;
- Orientado a documentos;

[3] http://mongodb.org

- Baseado em grafos.

O modelo chave-valor permite associar informações em pares chave-valor, o modelo orientado a documentos cria colecções de dados com estrutura e tipo semelhantes (embora sem exigir um esquema fixo) e o modelo baseado em grafos ajuda a formalizar as relações existentes entre os vários tipos de dados. O MongoDB adopta o modelo orientado a documentos, adaptando-se ao tipo de conteúdo que é seleccionado pelos utilizadores do Bundlr.

Este modelo incentiva também à desnormalização de dados de forma a melhorar a sua performance, evitando pedidos adicionais à base de dados para aceder a documentos que estão relacionados com o que estamos a apresentar. Por exemplo, poder-se-ia manter alguma informação base do autor de um bundle na própria tabela de bundles, evitando ir buscar o registo completo do utilizador à tabela de utilizadores. Por outro lado, este tipo de optimizações significa uma necessidade acrescida de evitar incoerências entre diferentes documentos da base de dados.

As bases de dados não-relacionais, embora não dêem garantias de integridade dos dados, podem também seguir um esquema base. Em MongoDB, chama-se colecção a uma tabela e documento a cada registo. Podem-se especificar relações entre as colecções através de referência ou de incorporação (*embedding*). As primeiras são semelhantes às relações de base de dados relacionais, enquanto que as últimas incluem os atributos da colecção incorporada dentro de cada documento da tabela principal.

Mapeamento Objecto-Documento

Um Object-Document-Mapper (ODM) é uma biblioteca que faz a conversão dos documentos de uma base de dados não-relacional orientada a documentos para objectos em memória (necessita portanto de um ambiente orientado a objectos). O mapeamento de MongoDB para a *framework* Ruby on Rails pode então ser feito através de uma destas bibliotecas. As principais são o MongoMapper[4] e o Mongoid[5], sendo esta última utilizada pelo Bundlr.

Embora na altura da implementação inicial do Bundlr o Mongoid fosse a biblioteca com melhor suporte à versão 3.0 do Ruby on Rails, actualmente o MongoMapper é a que é aconselhada pela equipa do MongoDB[7].

[4] http://mongomapper.com
[5] http://mongoid.org

Alojamento da Base de Dados - MongoHQ

A base de dados encontra-se hospedada no MongoHQ[6], um serviço que tem integração automática com o Heroku e cujos servidores estão geograficamente próximos dos servidores do Heroku, estando ambos instalados no serviço Amazon EC2[7][8, 9].

A empresa por trás do MongoHQ gere o sistema de gestão de base de dados, retirando essa responsabilidade da equipa de desenvolvimento do Bundlr. A gestão por uma equipa especializada neste tipo de bases de dados, indica que tem melhores conhecimentos para fazer os ajustes necessários para obter melhor performance da base de dados em geral. Em termos de escalabilidade, existem vários planos de serviço que definem a performance e o espaço disponibilizados para a base de dados em questão e que podem ser facilmente alterados pela equipa do Bundlr, sem necessidade de alterar configurações ou código.

Por outro lado, a delegação da manutenção da base de dados num serviço externo implica uma limitação na configurabilidade do sistema de base de dados, tornando impossível fazer pequenos ajustes que aumentem a performance de uma aplicação específica.

Actualmente estão alojadas neste serviço três bases de dados: desenvolvimento, *stage* e produção. A base de dados usada no desenvolvimento do serviço é partilhada pela equipa de programadores.

2.3 Serviços adicionais

Para além dos componentes base já descritos e dos serviços que os suportam, existem outros serviços externos que têm um papel fundamental ao minimizarem o esforço de administração associado a outros componentes da arquitectura. Esses serviços são apresentados de seguida:

Assembla[8] Serviço que aloja um servidor de Git, o sistema de controlo de versões utilizado. O Git tem como vantagem a integração com o Heroku que permite a publicação automática de código para *stage* ou produção.

New Relic[9] Ferramenta de monitorização de aplicações web que integra com o Heroku através de um *plugin* e que disponibiliza os tempos de resposta e carga de esforço do servidor da aplicação e da base de dados.

[6] https//mongohq.com
[7] http://aws.amazon.com/ec2
[8] http://www.assembla.com
[9] http://newrelic.com

O *plugin* é associado à aplicação no Heroku e monitoriza o tráfego real dos utilizadores do serviço, proporcionando assim dados reais para análise.

Airbrake[10] Serviço que monitoriza, agrega e notifica os programadores quando acontecem erros na aplicação.

Postmark[11] Serviço que se encarrega de enviar os emails da aplicação pelos seus próprios servidores, que já foram validados e verificados pelos servidores de email mais conhecidos, aumentando assim a probabilidade de o email não ser marcado como *spam* e ignorado. O serviço inclui também estatísticas sobre o sucesso do envio dos *mails* e bibliotecas que facilitam a integração com Ruby on Rails.

HireFire[12] Ferramenta que executa tarefas em *background*, que foram colocadas em espera pela aplicação.

Sendgrid[13] Serviço de envio de *newsletters*.

Kiss Metrics[14] Serviço utilizado para receber *feedback* em tempo real de utilizadores do sistema.

Google Analytics[15] Serviço de monitorização e análise do tráfego e dos visitantes da aplicação.

Paylane[16] Serviço de processamento de pagamentos.

AddThis[17] Serviço que disponibiliza estatísticas das partilhas feitas pelos utilizadores nas redes sociais, através da partilha de um URL próprio que contabiliza a partilha e redirecciona para o URL que se pretendia partilhar originalmente. Facilita também a colocação de botões de partilha na *interface*.

A utilização de todos estes serviços não traz só vantagens de diminuição da carga administrativa. De facto, ao recorrer-se a todos estes servidores externos para construir as próprias páginas da aplicação, está-se a diminuir

[10] http://airbrakeapp.com
[11] http://postmarkapp.com
[12] http://hirefireapp.com
[13] http://sendgrid.com
[14] http://kissmetrics.com
[15] http://analytics.google.com
[16] http://paylane.com
[17] http://addthis.com

a sua performance, já que são mais *scripts* Javascript e pedidos HTTP a executar. Em particular, serviços como o AddThis, mesmo sendo carregados de forma assíncrona depois do HTML da página ter carregado, fazem com que alguns elementos da página por que estão responsáveis só sejam mostrados depois de algum tempo, o que pode parecer estranho a quem esteja precisamente à procura desses elementos.

Além disso, segundo as 14 regras para websites mais rápidos de Steve Souders[10], deve-se tentar diminuir o número de DNS *lookups* feito em cada pedido, já que estes demoram tipicamente de 20 a 120 milissegundos a encontrar o endereço IP de um determinado *hostname*. Um teste com a ferramenta de análise do Yahoo YSlow[18] revelou rapidamente que a versão mais recente da *homepage* do Bundlr (Junho de 2012) tem 27 domínios diferentes, sendo 3 deles dos serviços apresentados em cima e os restantes das imagens de capa dos bundles e utilizadores apresentados, que não são alojadas no serviço.

Para aumentar a performance destes pedidos, a maioria dos *browsers* modernos têm mecanismos de *caching* da informação de mapeamento *hostname* para endereço de IP.

[18] http://developer.yahoo.com/yslow

Capítulo 3

Metodologia

Como já foi dito anteriormente, o Bundlr é um projecto recente. Por essa razão, ainda se encontra numa fase caracterizada pela sua natureza imprevisível, já que o modelo de negócio ainda se encontra em validação.

Esta imprevisibilidade tornou necessária a adopção de uma metodologia ágil para o desenvolvimento do estágio. Optou-se por uma variação do Scrum[11], uma metodologia de desenvolvimento iterativo e incremental bastante popular.

Ao longo deste capítulo serão descritas sucintamente as principais diferenças entre a metodologia de desenvolvimento utilizada e a original, finalizando com a descrição das metodologias de teste utilizadas durante a implementação.

3.1 Módulos: blocos de trabalho

Para facilitar a divisão dos objectivos práticos do estágio em blocos de trabalho bem definidos, adoptou-se o conceito de módulo. Este estágio será composto por 6 módulos, que serão apresentados no Capítulo 4. Os módulos terão uma duração consistente com o seu esforço estimado, sendo depois inseridos no contexto dos sprints, definidos também neste capítulo.

A descrição dos módulos neste documento segue uma estrutura que ajuda a definir o trabalho executado na sua implementação. Por esta razão, serão de seguida descritos os vários componentes de um módulo, tal como se encontram apresentados no Capítulo 5.

Os módulos são constituídos pelas seguintes secções: Motivação, Estado da Técnica, Análise de Requisitos, Implementação e Testes.

O resultado da análise de requisitos é uma lista de requisitos de alto nível que são implementados ao longo do sprint. Neste estágio, a sua descrição é

feita através de *User Stories* e de *Acceptance Stories*, decisão justificada no Apêndice G deste relatório.

A secção de testes descreve os testes que foram feitos para garantir o cumprimento dos requisitos não-funcionais definidos na secção Análise de Requisitos.

3.2 A vertente Scrum

A principal diferença entre a metodologia utilizada e o Scrum é o facto de apenas haver um elemento na equipa de implementação, uma vez que este estágio incide sobre funcionalidades em que só o estagiário vai trabalhar. Esta mudança é bastante significativa, já que todos os artefactos da metodologia Scrum deixam de ter a função de ajudar a coordenar a equipa internamente para passar a servir apenas para transmitir ao *Scrum Master* qual o progresso do estagiário.

Os papeis do Scrum são então distribuídos entre o estagiário que representa a Equipa de Desenvolvimento e Sérgio Santos, orientador do estágio por parte da empresa, que assume os papeis de *Scrum Master* e Cliente.

Os artefactos utilizados são os *Backlogs* de Produto e de Sprint e os Gráficos de *Burndown*, que são criados e mantidos pelo estagiário. Estes documentos estão nos Anexos externos M, N.1 a N.7 e no Apêndice C, respectivamente. As tarefas tanto podem ser *User Stories* como mais genéricas, tal como escrever o estado da técnica do módulo em desenvolvimento.

Relativamente às actividades, mantêm-se as reuniões de planeamento e encerramento do sprint, tal como a reunião de perspectiva diária.

3.2.1 Duração dos Sprints e Definição de *done*

Os sprints têm a duração de 4 semanas e podem compreender mais do que um módulo de trabalho. Da mesma forma, um módulo de trabalho pode-se estender até dois sprints.

Uma *User Story* é considerada concluída quando os seus testes de validação não apresentam erros e é escrita uma primeira versão da secção correspondente no relatório.

Já um módulo é considerado completo quando todas as suas *User Stories* e tarefas estiverem concluídas.

3.3 Metodologias de Teste Funcionais

No início do estágio apenas cerca de 5% da aplicação estava coberta por testes, na sua maioria desactualizados e a falharem a execução. Embora o trabalho de criação de testes para garantir uma cobertura aceitável da aplicação seja importante, é algo bastante dispendioso em termos de tempo e fora do âmbito deste estágio.

No entanto, para que o software criado seja de melhor qualidade, a implementação dos módulos é acompanhada por testes unitários e de regressão, que serão descritos mais em detalhe de seguida.

3.3.1 Testes Unitários

Para a implementação dos testes unitários utilizou-se a biblioteca RSpec[1], que integra com Ruby on Rails e é mais simples de usar que a biblioteca de testes original (Test::Unit).

Esta foi usada para testar procedimentos atómicos e complexos definidos tanto nos modelos como nos controladores do modelo *Model-View-Controller*.

Foi também utilizada a biblioteca FactoryGirl[2], que permite criar representações de objectos em memória e agilizar a sua criação na base de dados de teste.

O Apêndice L lista o *spec* de testes unitários implementados.

3.3.2 Testes de Regressão

Os testes de regressão automatizados são essenciais por duas razões: facilitam imenso o processo de validação das *User Stories* e garantem que o sistema funciona como se pretende, isto é, como um todo e não só nos seus diversos componentes, que é a função dos testes unitários.

Como ferramenta para implementar esta automatização foi escolhida a biblioteca Cucumber[3]. Esta permite escrever testes de aceitação em linguagem natural, no formato que é descrito para as *Acceptance Stories* no Apêndice G. O Cucumber simula o comportamento de um *browser* web, navegando pelas páginas e verificando a presença de elementos no seu conteúdo[12], tal como descrito nas *Acceptance Stories* implementadas pelo programador.

O Apêndice B lista as *Acceptance Stories* que dão origem aos testes de regressão implementados.

[1] http://rspec.info
[2] http://github.com/thoughtbot/factory_girl
[3] http://cukes.info

Capítulo 4

Planeamento

Neste estágio os módulos fazem a ponte entre o trabalho desenvolvido e os seus objectivos práticos. Segue-se a sua apresentação.

4.1 Módulos de Trabalho

A tabela 4.1 descreve resumidamente os módulos de trabalho realizados tanto no primeiro semestre (até ao 3º) como no segundo (do 4º ao 6º). O trabalho desenvolvido nos módulos é descrito no Capítulo 5.

O primeiro sprint serviu para enquadrar o estagiário no projecto, dar início ao estudo do estado da arte e definir a metodologia a utilizar, não sendo considerado um módulo de trabalho.

4.2 Alterações ao plano inicial

Tal como se previa no Relatório Intermédio deste estágio, foram feitas alterações ao planeamento inicial, nomeadamente a substituição dos módulos API de leitura e API de escrita pelo módulo *Full-Text Search*. Esta alteração foi feita a pedido da empresa, que mostrou urgência em melhorar a funcionalidade de pesquisa do website.

4.3 Diagramas de Gantt

O planeamento do estágio foi também feito recorrendo a gráficos de Gantt. No Apêndice D apresenta-se o diagrama do planeamento inicial e o diagrama final.

# Sprint	Módulo	Descrição
Sprint 0	Iniciação (80h)	Integração no projecto, análise da competição, estudo da arquitectura e código-fonte do serviço e definição da metodologia de desenvolvimento.
Sprint 1	1 - Bundles privados (40h)	Permitir que certos bundles criados pelos utilizadores apenas sejam visíveis por estes e pelos seus colaboradores, se o autor subscrever o respectivo plano pago do serviço.
	2 - Limitar colaboração (40h)	Apenas permitir colaboração na criação de bundles aos utilizadores que subscreveram o respectivo plano pago.
Sprint 2	3 - Subscrição de bundles e utilizadores (160h)	Permitir que um utilizador subscreva as actualizações de um bundle ou as actividades de um utilizador. As actividades subscritas serão apresentadas em forma de *activity stream*.
Sprint 3		
Sprint 4	4 - Categorização e exploração de bundles (160h)	Facilitar a descoberta de bundles de qualidade criados por outros utilizadores. Um bundle poderá pertencer a uma categoria e os utilizadores poderão navegar pelos melhores bundles de cada uma.
Sprint 5	5 - Sugestão de conteúdo (320h)	Sugerir conteúdo relevante para cada utilizador. Construção do motor de sugestão que extraia informação de cada clip e compare utilizadores, bundles e clips.
Sprint 6		
Sprint 7	6 - *Full-Text Search* (160h)	Permitir a pesquisa dentro do conteúdo dos clips e não apenas pelos títulos dos bundles.

Tabela 4.1: Descrição dos Módulos de Trabalho realizados durante o estágio, associados aos respectivos sprints. A tabela está dividida em duas secções que representam os dois semestres do estágio.

Capítulo 5

Módulos de Trabalho

Neste capítulo é apresentado o trabalho desenvolvido ao longo do estágio, dividido pelos módulos apresentados na tabela 4.1 do Capítulo 4. A descrição de cada módulo segue o formato referido na secção 3.1 do Capítulo 3.

5.1 Módulo 1: Bundles Privados

5.1.1 Motivação

Um dos aspectos observados pela empresa foi que utilizadores que usam o Bundlr num contexto profissional sentem a necessidade de seleccionar conteúdo em bundles privados. O exemplo mais evidente é uma equipa de trabalho que use o serviço para reunir conteúdo para um projecto e que não queira que esse conteúdo seja público, correndo o risco de ser encontrado por competidores.

Ao ser privado, o bundle deixa de ser visível para utilizadores que não tenham sido explicitamente autorizados pelo autor.

5.1.2 Estado da Técnica

A agregação de conteúdos em colecções privadas não é um conceito novo. Para facilitar a contextualização e o levantamento de requisitos da funcionalidade implementada, foram analisadas algumas soluções alternativas ao Bundlr. A tabela 5.1 apresenta as soluções analisadas.

Como se pode ver, foram identificados dois tipos de opções no que toca a colecções privadas: não publicação (rascunho) e definição da privacidade. Estes dois tipos serão detalhados de seguida.

	Colecções Privadas	Escolha ao criar	Visibilidade alterável	Privado por defeito
Storify	Através de rascunho	Sim	Não	Sim
Scoop.it	Não	-	-	-
Evernote	Sim	Não	Sim	Sim
Delicious	Através de rascunho	Sim	Não	Sim
Zootool	Sim	Não	Não	Sim
Gimme Bar	Sim	Sim	Sim	Não

Tabela 5.1: Comparação de serviços com colecções privadas.

Não Publicação

Estas soluções implementam colecções privadas de forma implícita, através da introdução de rascunhos. A lógica subjacente é que um utilizador pode alterar e preparar a sua colecção antes de a publicar. Desta forma, um rascunho apenas é visível para o seu autor e nunca para o público. Estas soluções tornam-se pouco intuitivas porque não são feitas à medida para conteúdo privado e pressupõem sempre que o utilizador acabará por publicar. Geralmente, neste tipo de soluções, não há forma de inverter a publicação do conteúdo.

Definição de Privacidade

Estas soluções definem, na sua maioria, dois níveis de privacidade para as colecções - público e privado. Ao contrário da não publicação, a possibilidade de esconder o conteúdo do público em geral é introduzida explicitamente na interface, normalmente acompanhada com uma descrição que assegura ao utilizador que o conteúdo privado apenas é visível por ele próprio. Algumas soluções apresentam também um terceiro nível que permite que uma colecção seja pública mas não indexada na funcionalidade de pesquisa do sistema.

Além das categorias de colecções privadas já detalhadas foram identificadas algumas assimetrias no que toca à possibilidade de alterar a visibilidade de uma colecção após a sua criação. Verifica-se que na maioria dos serviços com definição de privacidade é possível fazer esta alteração. Outro pormenor interessante é que ter colecções privadas não é uma funcionalidade paga em nenhum dos serviços analisados.

5.1.3 Análise de Requisitos

Tal como no restante trabalho desenvolvido, o estudo e análise dos requisitos do módulo foi feito através de *User Stories*, antes do início do desenvolvimento. As *User Stories* são histórias de utilização simplificadas que descrevem uma funcionalidade ou passo em específico.

No apêndice B.1 apresenta-se a sua descrição completa, incluindo as respectivas *Acceptance Stories*.

Requisitos não-funcionais

Os requisitos não-funcionais definem restrições relacionadas com a performance, segurança e escalabilidade das soluções encontradas para as funcionalidades descritas através das *User Stories*. Durante a análise de requisitos foi feita uma selecção dos requisitos não-funcionais mais importantes, que será listada de seguida.

Durante esta selecção não foram encontrados requisitos de segurança e de escalabilidade específicos das alterações implementadas neste módulo.

Desempenho

D.1. A pesquisa não deverá ter uma perda de desempenho de mais de 200ms com a implementação do módulo.

5.1.4 Implementação

A implementação das *User Stories* deu origem a alguns problemas que exigiram uma análise mais cuidada. Esta secção tem o objectivo de apresentar resumidamente esses problemas e as soluções encontradas.

No estado da técnica (secção 5.1.2) analisaram-se duas formas de implementar colecções privadas. Para a implementação deste módulo foi escolhida a opção Definição de Privacidade, por ser mais adequada ao problema.

Um dos principais problemas encontrados durante a implementação está relacionado com a pesquisa de bundles: que bundles é que podem fazer parte dos resultados? Devemos simplesmente excluir os bundles privados de todos os resultados? Dependerá do utilizador que está a pesquisar? A necessidade de um utilizador conseguir encontrar facilmente um dos seus bundles (privado ou não) levou, então, à personalização dos resultados para cada utilizador. No entanto, esta decisão adicionou complexidade à *query* feita à base de dados.

A pesquisa original do sistema procurava sempre em todos os bundles existentes. Com este módulo, passou-se a fazer uma pesquisa em bundles

públicos e, se o utilizador tiver a sessão iniciada, nos seus bundles, públicos ou não, incluindo colaborativos. Para isso, implementou-se a *query* que se segue:

```
(bundles públicos OR bundles do utilizador) AND
(título do bundle ~= query OR descrição do bundle ~= query)
```

Nesta, começa-se por se seleccionar o conjunto de bundles pesquisáveis, através da primeira instrução *OR*, fazendo-se depois a pesquisa pelos campos título e descrição do bundle.

O aumento da complexidade da pesquisa levou a uma perda no seu desempenho, o que pode ser analisado mais pormenorizadamente na secção 5.1.5 deste módulo.

A restante implementação focou-se em fazer com que só utilizadores que subscrevem um plano pago tenham acesso à opção de tornar um bundle privado, dando a opção de subscrição para os restantes utilizadores. Além disso, foram também considerados casos em que o utilizador deixa de pagar pelo serviço e tenta aceder a um bundle privado. A figura 5.1 apresenta a mensagem que esse utilizador viria.

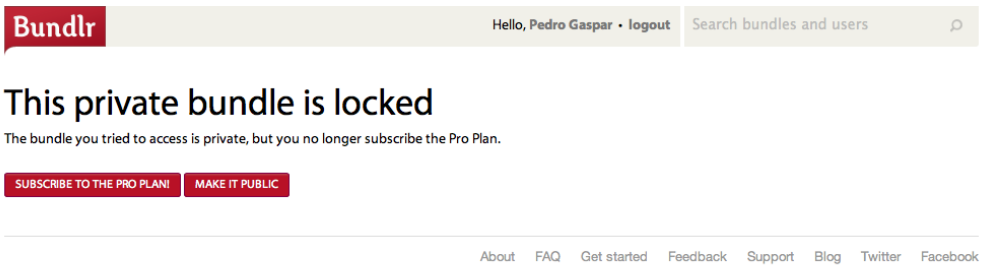

Figura 5.1: Mensagem de bloqueio de um bundle privado para um utilizador que deixou de pagar pelo serviço. Recolhida em Dezembro de 2011.

O Apêndice E.2 apresenta as alterações feitas neste módulo a nível dos modelos e controladores da arquitectura MVC da aplicação.

5.1.5 Testes

Testes Funcionais

Os testes de aceitação são a materialização das *Acceptance Stories* introduzidas na análise de requisitos.

Passaram **106** dos **106** cenários de teste derivados das *Acceptance Stories* descritas no Apêndice B.1.

Os testes unitários operam a um nível mais baixo que os de aceitação e testam algumas funções e operações mais complexas.

Passaram **17** dos **17** testes unitários criados para este módulo. O *spec* final de testes unitários é listado no Apêndice L.

Testes de Desempenho

Nesta secção avalia-se o efeito das alterações descritas anteriormente no desempenho da funcionalidade de pesquisa de bundles, tendo em conta o requisito não funcional de desempenho D.1.

A base de dados utilizada para os testes que se apresentam é composta por 7150 bundles. A pesquisa feita manteve-se constante, tendo sempre 410 resultados, um dos quais privado.

Os testes foram feitos usando um utilizador com sessão iniciada que tem 15 bundles (um deles privado) e 3 colaborações (duas delas privadas).

O primeiro teste pretende analisar as diferenças de desempenho de um pedido remoto à acção de pesquisa. Para isso, fez-se uma série de 300 pedidos à página, separados por 40±20 milissegundos, e registou-se o intervalo de tempo até à chegada do HTML do servidor. Para este teste foi utilizada a ferramenta Apache JMeter[1].

A tabela 5.2 apresenta os resultados do primeiro teste. Observa-se uma diminuição de desempenho de cerca de 75ms na pesquisa introduzida neste módulo em comparação com a original, mas também que o desvio padrão é bastante significativo e inviabiliza quaisquer conclusões em termos comparativos. A suspeita da origem deste desvio padrão estar associada aos tempos de latência de rede entre o computador cliente (Portugal) e o servidor (Estados Unidos da América) leva a novo conjunto de testes.

	Pesquisa Original	**Pesquisa Implementada**
Tempo médio	1.086 ± 0.226	1.159 ± 0.2846

Tabela 5.2: Desempenho da pesquisa remota para o código original e para o implementado no módulo. Resultados em segundos.

Desta forma, para tornar os resultados mais discriminantes optou-se por fazer testes no próprio servidor, eliminando tempos de latência de rede entre cliente e servidor. Estes testes dividiram-se em dois tipos: pedidos locais à acção de pesquisa e execução directa da *query* em questão (sem qualquer tipo

[1] http://jmeter.apache.org

de computação de HTML nem pedido HTTP). Os testes foram executados em série, 300 vezes, com 40±20 milissegundos de intervalo.

A tabela 5.3 apresenta os resultados destes testes. Analisando os resultados dos pedidos à acção de pesquisa (pedidos HTTP), verifica-se uma pequena diminuição do desvio padrão, tão baixa que sugere que a distância entre o cliente e o servidor não são a sua causa. Por outro lado, observando os resultados da execução directa da *query* vemos uma diminuição do desvio padrão mais acentuada, levando à conclusão que o factor diferenciador está, então, relacionado com o servidor web e com os acessos à base de dados.

	Pesquisa Original	**Pesquisa Implementada**
Pedido HTTP	0.8164 ± 0.1889	0.8522 ± 0.2779
Execução da *Query*	0.2931 ± 0.0284	0.3210 ± 0.0351

Tabela 5.3: Desempenho da pesquisa local ao servidor tanto a nível de um pedido HTTP completo como a nível da *query*. Resultados em segundos.

Quanto aos resultados em si, no teste ao pedido HTTP, não é possível tirar conclusões devido ao desvio padrão observado. No segundo teste, mesmo com a diminuição do desvio padrão que se verificou, os valores não indicam uma diferença significativa entre o desempenho das duas soluções. Segundo estes dados, existe 95.45% de probabilidade do valor da pesquisa original estar no intervalo [0.2363, 0.3499] ($[-2\sigma, 2\sigma]$) e 95.45% de probabilidade da nova pesquisa ficar no intervalo [0.2508, 0.3912] ($[-2\sigma, 2\sigma]$). Desta forma, pode-se concluir que a diferença entre as duas soluções será, com 91.11% de probabilidade, menor que $0.3912 - 0.2363 = 0.1549$, ou seja, menor que cerca de 155ms. Desta forma, conclui-se que o desempenho da pesquisa não deverá sofrer uma perda superior a 200ms, com este nível de probabilidade.

Analisando detalhadamente a *query* identificam-se dois pontos em que esta se pode tornar mais lenta que a original: omissão dos bundles privados de outros utilizadores e inclusão dos bundles privados do utilizador, caso tenha a sessão iniciada. A filtragem por bundles públicos representa um desempenho na mesma ordem de grandeza da *query* original, destacando assim a inclusão dos bundles privados do utilizador como o principal interveniente na possível degradação do desempenho.

Tendo isso em conta, o principal factor passa a ser a quantidade de bundles colaborativos privados que o utilizador em questão tem, já que o aumento do número de bundles na base de dados vai degradar as duas soluções de forma semelhante. Considerando, então, que se prevê a estabilização da média da quantidade de bundles colaborativos por utilizador, admite-se que a degradação correspondente para o utilizador normal não vai aumentar ao longo do tempo de forma significativa.

5.2 Módulo 2: Limitação da Colaboração

5.2.1 Motivação

A colaboração é uma funcionalidade presente no serviço desde muito cedo. De facto, tem sido um dos factores diferenciadores na comparação com os competidores mais próximos. Por estas razões, a decisão de limitar a colaboração a contas pagas pode parecer contraditória. No entanto, essas dúvidas são atenuadas pela descrição do público-alvo das contas pagas: equipas de trabalho e profissionais.

Ou seja, o Bundlr está a dar um especial enfoque à colaboração entre equipas de trabalho, em detrimento de outro tipo de colaboração mais aberta, como a que se vê na Wikipedia, por exemplo. Unindo os bundles privados com a colaboração temos então um conjunto de funcionalidades de grande valor para este público-alvo.

5.2.2 Estado da Técnica

Tendo em conta que os bundles colaborativos já faziam parte do serviço e que o trabalho desenvolvido neste módulo foi principalmente o de limitar o acesso a essa funcionalidade a utilizadores pagantes, a análise do estado da técnica será somente uma rápida comparação entre os serviços competidores. Os parâmetros de comparação são se o serviço oferece colaboração e, se sim, se é paga. A tabela 5.4 é o resultado dessa análise.

Serviço	**Colaboração?**	**Pago?**
Storify	Não	-
Scoop.it	Sim	Sim
Evernote	Sim	Sim
Delicious	Não	-
Zootool	Não	-
Gimme Bar	Sim	Não

Tabela 5.4: Comparação da colaboração com os serviços competidores.

5.2.3 Análise de Requisitos

As *User Stories* resultantes da análise de requisitos encontram-se no apêndice B.2, com as respectivas *Acceptance Stories*. Algumas delas foram inspiradas pelas do módulo anterior, por proporcionarem funcionalidades semelhantes.

Requisitos não-funcionais

Durante a análise de requisitos não foram encontrados requisitos de segurança, escalabilidade e desempenho específicos deste módulo. Isto deve-se principalmente ao facto de ser um módulo simples, comparativamente com os restantes.

5.2.4 Implementação

O controlo de permissões é algo fulcral para garantir a segurança da aplicação e dos dados dos utilizadores. Na solução implementada, foi criado um método no modelo Bundle que dita se um utilizador tem ou não permissões para editar o conteúdo do bundle. Este método tem o nome de *can_edit?* e recebe o utilizador que se pretende testar por parâmetro, devolvendo um booleano que indica se este tem ou não permissão para editar o bundle.

O *can_edit?* combina uma série de condições que dão origem ao comportamento que é definido em algumas das *User Stories* deste módulo e do que lhe antecede. As condições vão desde testar se o utilizador é colaborador do bundle até testar se o autor do bundle subscreve algum plano pago, acabando por definir uma mensagem de erro, no caso da permissão ser negada.

O facto do utilizador ser capaz de alterar o seu plano torna esta gestão de permissões um pouco mais complexa. É necessário um mecanismo que determine se o autor está numa situação de não renovação da subscrição do Plano Profissional e bloqueie o acesso aos seus bundles privados, por exemplo. A detecção destes casos é feita através das condições que compõem o *can_edit?*. A tabela 5.5 descreve estas possibilidades.

A figura 5.2 representa a mensagem vista pelos colaboradores no caso de tentarem criar um clip num bundle colaborativo cujo plano seja agora grátis.

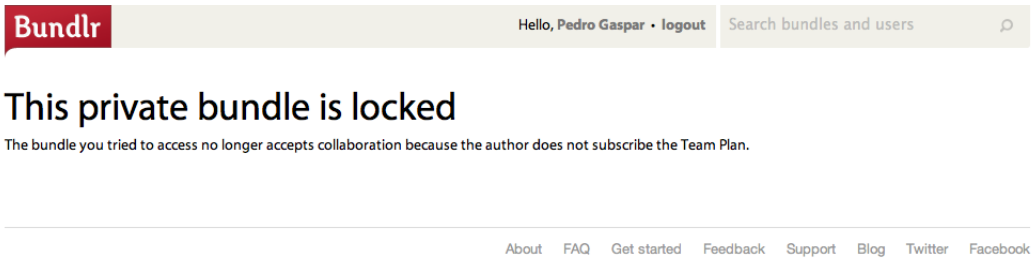

Figura 5.2: Mensagem de bloqueio de um bundle colaborativo quando o autor do bundle deixou de pagar pelo serviço. Recolhida em Dezembro de 2011.

O *can_edit?* é chamado várias vezes durante a utilização do serviço (cada vez que se mostra um bundle, por exemplo), pelo que é importante tentar

Privado?	Colaborativo?	Plano	Acção
Sim	Sim	Grátis	Impedir
		Equipa	
		Profissional	Permitir
Sim	Não	Grátis	Impedir
		Equipa	
		Profissional	Permitir
Não	Sim	Grátis	Impedir colaboradores
		Equipa	Permitir
		Profissional	
Não	Não	Grátis	Permitir
		Equipa	
		Profissional	

Tabela 5.5: Descrição das situações tratadas pela rotina *can_edit?*.

optimizá-lo. Para isso deu-se prioridade ao teste de algumas condições, como por exemplo, impedir imediatamente a edição se o utilizador não for nem autor nem colaborador, antes de testar as condições relacionadas com o plano a que o autor subscreve.

O Apêndice E.3 apresenta as alterações feitas neste módulo a nível dos modelos e controladores da arquitectura MVC da aplicação.

5.2.5 Testes

Testes de Funcionais

Passaram **11** dos **11** cenários de teste derivados das *Acceptance Stories* descritas no Apêndice B.2.

Passaram **13** dos **13** testes unitários criados para este módulo. O *spec* final de testes unitários é listado no Apêndice L.

5.3 Módulo 3: Subscrição de Bundles e de Utilizadores

5.3.1 Motivação

No cenário da Internet actual, o acto de subscrever conteúdo produzido por outra pessoa é um dos pilares que aumenta o valor da experiência de interacção social online. A sua importância é tanta que serviços criados só em torno desta interacção social, como o Facebook[2] e o Twitter[3], são dos websites mais utilizados pelo público em geral[13].

Do ponto de vista de negócio, a subscrição de conteúdo é uma forma de aumentar a taxa de retenção, ou seja, de manter utilizadores no serviço. Ao subscrever as actualizações de um conhecido, o utilizador está a permitir a emissão de notificações, que podem ser entregues, por exemplo, por email. Este tipo de notificação aumenta a probabilidade do utilizador voltar ao serviço, já que este lhe apresenta novidades relevantes.

Neste módulo pretende-se que os utilizadores do Bundlr possam subscrever as actualizações de bundles (por exemplo, a adição de um clip) e de outros utilizadores (por exemplo, a criação de um novo bundle).

5.3.2 Estado da Técnica

São vários os modelos de subscrição de conteúdo utilizados na Internet. Um dos mais complexos de configurar, que teve uma subida de popularidade em 2005[14], foi a subscrição de *feeds* RSS. Estas são ficheiros em XML actualizados pelo servidor com os últimos conteúdos e que se podem subscrever através de um leitor de *feeds*, como o Google Reader[4]. Os utilizadores podem então juntar os *feeds* das actualizações do conteúdo que querem seguir, que têm de ser disponibilizados pelo serviço, e depois consultar as novidades no seu leitor de eleição.

Esta abordagem tem a vantagem de juntar num único leitor todo o tipo de conteúdo de diferentes serviços e websites que o utilizador queira seguir. No entanto, ao exigir tantos passos de configuração e ao fazer com que a leitura seja feita fora do Bundlr, esta solução torna-se inadequada.

O modelo mais comum é o modelo adoptado pelas redes sociais, as *activity streams*[15]. Este modelo junta todas as actualizações que o utilizador subscreve numa lista ordenada cronologicamente da actividade mais recente para

[2] http://facebook.com
[3] http://twitter.com
[4] http://reader.google.com

a mais antiga. O utilizador passa então a poder subscrever conteúdo através da própria interface do serviço, tornando-se numa experiência bastante mais integrada e personalizada que a apresentada anteriormente.

Uma análise das ferramentas competidoras mostra que só o Evernote, a ferramenta de uso principalmente privado, não incorpora alguma forma de subscrição do conteúdo de outros utilizadores. Descrevem-se de seguida as formas de subscrição dos outros serviços.

Storify Subscrição de utilizadores. Apenas reconhece a actividade de criação de novas colecções.

Scoop.it Subscrição de colecções. Cria uma actividade cada vez que é adicionado conteúdo à colecção.

Delicious Subscrição de utilizadores e de colecções. Na primeira, são criadas actividades quando o utilizador guarda um link e publica uma colecção. Na última, é criada uma actividade cada vez que é adicionado conteúdo à colecção.

Zootool Subscrição de utilizadores. É criada uma actividade cada vez que o utilizador adiciona conteúdo ao serviço.

Gimme Bar Subscrição de utilizadores e de colecções. São criadas actividades quando é adicionado conteúdo pelo utilizador ou na colecção subscrita.

Um dos principais exemplos de sucesso na implementação do modelo *activity stream* é o Facebook. O serviço é construído à volta da *wall*, a lista de actividades recentes dos amigos do utilizador. A popularidade que o serviço tem exige que toda a arquitectura por trás do armazenamento de actividades e dos acessos a essa informação seja escalável e robusta.

Embora o número de utilizadores do Bundlr não se possa comparar com o do Facebook, não deixa de ser importante ter em consideração questões de performance e escalabilidade ao desenhar a arquitectura de uma solução que siga o modelo *activity stream*. Segue-se uma descrição dos principais desafios de engenharia deste módulo:

Armazenamento das actividades

A forma como as actividades são guardadas na base de dados vai influenciar directamente o tempo que demora a sua publicação, a sua recuperação para ser apresentada e o tamanho da base de dados. Descrevem-se de seguida as duas principais opções encontradas:

Tabela de Actividades Consiste em manter uma tabela com todas as actividades registadas pela aplicação. Cada actividade teria uma ligação aos utilizadores que a devem receber na sua *activity stream*.

Incorporação nos Utilizadores Mantêm-se as actividades de cada utilizador através de uma relação de incorporação na colecção Utilizadores.

A primeira opção tem a vantagem de não implicar a repetição de actividades, que é o principal defeito da segunda. Além disso, torna a publicação de uma actividade bastante mais rápida, por apenas ter de criar um pequeno registo na tabela, ao contrário da segunda opção que tem de incorporar esse registo em cada utilizador subscritor. Por outro lado, ao manter os dados numa tabela separada, torna a construção da *activity stream* bastante mais lenta, o que é minimizado pela segunda opção. Ou seja, estas duas opções exigem um compromisso: publicação rápida e recuperação lenta ou o contrário.

Este compromisso incentiva a procura de soluções híbridas que aproximam os tempos de publicação dos tempos de recuperação, embora com uma esperada perda de performance em relação às soluções base apresentadas. Um exemplo deste tipo de soluções híbridas é a gestão tanto de uma tabela separada de actividades como a incorporação dos identificadores das actividades na colecção Utilizadores. Note-se que mesmo esta solução pode levar a um crescimento bastante acentuado da base de dados, se todos os utilizadores tiverem muitos seguidores.

Por esta razão, toma-se a opção de tabela separada como favorita, embora se continue com o problema da lentidão da construção da *activity stream*, que pode ser atenuado, por exemplo, com a utilização de índices na tabela de actividades.

Outra optimização que será explorada na secção 5.3.4 envolve a colocação de operações mais lentas em espera para serem processadas em *background*. Desta forma conseguem-se reduzir ainda mais os tempos de publicação das actividades.

Existem também outras soluções mais complexas a nível de arquitectura e de manutenção futura que envolvem a configuração e utilização de um servidor dedicado para a gestão das actividades. Estas soluções fazem sentido quando o foco da aplicação é a rede social e quando se encontra um sistema de base de dados ideal para guardar as actividades. Como o MongoDB já é eficiente para guardar este tipo de dados[16] e como se dá prioridade a opções que não envolvam a gestão de mais bases de dados, descartou-se esta opção.

Representação de uma actividade

A forma de representação de uma actividade também é uma questão importante. A diversidade de actividades que podem ser registadas implica alguma flexibilidade no que toca à sua representação na base de dados e à sua apresentação na interface.

A especificação Activity Streams 1.0[17] é um esforço no sentido da normalização dessa representação entre os diversos serviços com componente social. Esta especificação define uma actividade como um conjunto de 3 elementos: o actor, o verbo e o objecto. O verbo é a acção executada, o actor é quem a executa e o objecto o resultante da acção. Há também a opção de incluir o elemento alvo, que indica onde aconteceu a acção.

Tipo de relação social

Existem dois tipos principais de relações sociais na Internet, no que toca à subscrição de conteúdo, que serão descritos de seguida.

Amizade Este é o modelo seguido pelo Facebook. Para um utilizador poder ter acesso ao conteúdo de outro tem de haver uma confirmação de ambas as partes. Ou seja, o utilizador cujo conteúdo foi subscrito tem de aceitar a amizade do primeiro utilizador, passando ambos a ter acesso à informação do outro.

Seguidor/Seguido Este é o modelo implementado pelo Twitter. Os utilizadores podem seguir o conteúdo de outros sem a sua autorização explicita.

Tendo em conta que o interesse pelo conteúdo de um utilizador não implica que esse utilizador queira seguir o nosso conteúdo, conclui-se que o modelo Seguidor/Seguido é o mais adequado para o problema em mãos.

5.3.3 Análise de Requisitos

A análise de requisitos deste módulo esteve muito relacionada com as questões colocadas no estado da técnica. O estudo necessário para a escolha das melhores soluções para resolver os problemas encontrados, assim como a sua implementação, ocupou o primeiro dos dois sprints dos quais este módulo fazia parte.

As *User Stories* foram implementadas no segundo sprint, sobre a estrutura idealizada e implementada no primeiro sprint. No apêndice B.3 apresenta-se a descrição completa dessas *User Stories*, incluindo as respectivas *Acceptance Stories*.

A tabela 5.6 contém as actividades que serão tidas em conta na implementação das subscrições.

Elemento subscrito	Actividade
Bundle	Novo clip
	Novo colaborador
	Seleccionado para *homepage*
Utilizador	Criou bundle
	Começou a colaborar num bundle
	Seguiu utilizador
	Seguiu bundle
	Subscreveu o plano Equipa
	Subscreveu o plano Profissional

Tabela 5.6: Tipos de actividade.

Requisitos não-funcionais

Durante a análise de requisitos foi feita uma selecção dos requisitos não-funcionais mais importantes, que será listada de seguida. Não foram encontrados requisitos de segurança ou escalabilidade.

Note-se que apenas são apresentados os requisitos específicos das alterações implementadas neste módulo.

Desempenho

D.1. A publicação de actividades deve ser feita em menos de 200ms, embora só possa ser mostrada na *activity stream* até 2 minutos depois[5];

D.2. O processamento da *activity stream* de um utilizador no servidor não deverá demorar mais do que 500ms.

5.3.4 Implementação

Antes de mais, a figura 5.3 apresenta o aspecto final da *activity stream*. Esta secção detalha os desafios de engenharia que surgiram neste módulo.

[5]Ver requisitos não-funcionais de escalabilidade.

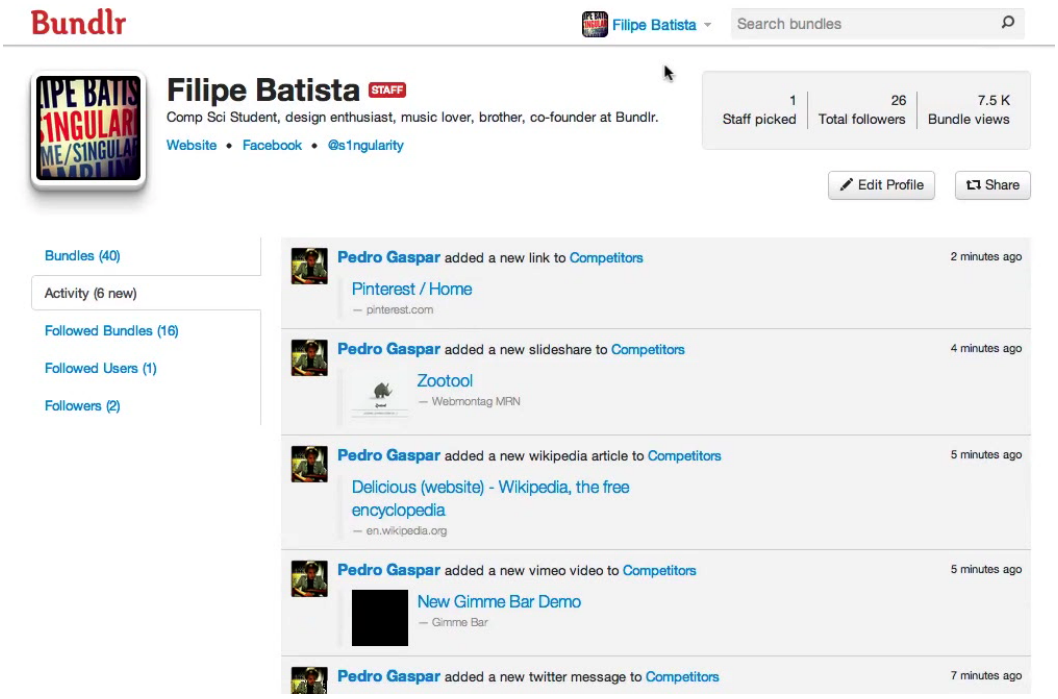

Figura 5.3: *Activity Stream* do utilizador Filipe Batista com actividades do utilizador Pedro Gaspar. À esquerda encontram-se os links para as listagens de bundles e utilizadores seguidos pelo Filipe Batista e para a lista dos seus seguidores. Recolhida em Janeiro de 2012.

Escolha da solução adequada

Durante a análise do estado da técnica encontraram-se algumas opções no que toca ao armazenamento das actividades na base de dados (página 30) e à sua forma de representação (página 32). Depois dessa análise foi feita uma pesquisa por bibliotecas que pudessem já implementar algumas das funcionalidades que se procuram desenvolver. Foram encontradas algumas opções, mas a que destacou foi o Streama[6] por integrar com o Ruby on Rails versão 3.0, por ser especialmente desenvolvida para Mongoid (o ODM utilizado) e por seguir a especificação Activity Streams 1.0. Quanto ao armazenamento das actividades, o Streama segue a abordagem de manter uma tabela de actividade separada, cujas actividades contêm uma lista dos utilizadores a quem devem ser apresentadas.

O Streama inclui já algumas funcionalidades interessantes, como *caching* de atributos dos actores, objectos e alvos ou a inclusão de índices para me-

[6] https://github.com/christospappas/streama

lhorar o desempenho da construção da *activity stream* de cada utilizador. Ao contrário de outras, esta biblioteca não tem quaisquer funcionalidades específicas para Ruby on Rails no que toca à apresentação da lista de actividades, deixando isso ao critério do programador. De forma a permitir a alteração do código da biblioteca, esta foi instalada no próprio repositório da aplicação.

Para testar e comparar o desempenho de uma solução em que as actividades passam a não conter a lista de utilizadores subscritores, procedeu-se à alteração do código da biblioteca, criando uma segunda solução a que se dará o nome de "Streama v2" durante esta secção. Um dos componentes que necessitou de alteração foi a *query* de construção da *activity stream*, que se tornou mais complexa por ser necessário filtrar a tabela de actividades excluindo as que não foram feitas pelos utilizadores subscritos, em vez de excluir apenas as que não contêm o utilizador na lista de receptores.

Para comparar as duas soluções, fez-se uma análise das acções de publicação de actividade e de construção da *activity stream*, que levou à selecção das 3 principais variáveis a testar:

Tamanho da *activity stream* Determinar a influência do número de actividades que compõe a *activity stream* no desempenho da sua construção.

Popularidade do conteúdo subscrito Determinar a influência do número de utilizadores que receberam a actividade no tempo que demora a sua recuperação da base de dados.

Tamanho da audiência Determinar a influência do número de utilizadores que irão receber a actividade no desempenho da sua publicação.

Os resultados comparativos relativamente ao primeiro critério são apresentados na tabela 5.7. Analisando-os podemos concluir que as soluções são semelhantes, mas também que os resultados são piores para 10000 actividades na *activity stream*. O problema é rapidamente solucionado se se tiver em conta que nunca seria apresentada a totalidade das actividades na *interface* da aplicação. Desta forma, admitindo que se limitava o número de actividades a recuperar na *query* de construção da *activity stream* para 30, os resultados seriam igualmente baixos para ambas as soluções.

Os resultados referentes ao segundo critério (Popularidade do conteúdo subscrito) são dispostos na tabela 5.8. Estes resultados mostram que a performance do Streama v2 se degrada consideravelmente com o aumento da popularidade da actividade. Isto significa que, quanto mais subscritores tiver um utilizador, mais lenta será a apresentação da *activity stream* de cada subscritor, o que é preocupante. Esta degradação é de cerca de 22 vezes de

Nº de actividades	Streama	Streama v2
100	0.0496 ± 0.0020	0.0595 ± 0.0234
1000	0.5001 ± 0.0055	0.4935 ± 0.0176
5000	4.1146 ± 0.2692	4.2488 ± 0.2544
10000	12.8390 ± 1.1414	12.2801 ± 1.6212

Tabela 5.7: Influência do tamanho da *activity stream* na sua construção. Resultados em segundos.

100 subscritores para 10000, sendo bastante menor no Streama original, cujos resultados baixam para cerca de 4 vezes no mesmo intervalo de subscritores.

Nº de utilizadores	Streama	Streama v2
100	0.0094 ± 0.0009	0.0157 ± 0.0055
1000	0.0114 ± 0.0008	0.0358 ± 0.0083
5000	0.0228 ± 0.0044	0.145 ± 0.0110
10000	0.0425 ± 0.0044	0.351 ± 0.0289

Tabela 5.8: Influência do número de utilizadores a quem a actividade diz respeito na sua recuperação da base de dados. Resultados em segundos.

Os resultados referentes ao último critério (Tamanho da audiência) são apresentados na tabela 5.9. Os resultados mostram as desvantagens de manter a lista de utilizadores que irão receber a actividade, dentro da própria representação da actividade. Ao ser necessário guardar essa informação extra, a escrita torna-se mais lenta e a tabela maior. A solução Streama v2 mantém sempre um alto desempenho por guardar sempre a mesma informação, independentemente da sua audiência.

Nº de utilizadores	Streama	Streama v2
100	0.0314 ± 0.0664	0.0063 ± 0.0027
1000	0.1303 ± 0.1319	0.0055 ± 0.0016
5000	1.0768 ± 0.2123	0.0043 ± 0.0004
10000	5.0337 ± 0.7784	0.0045 ± 0.0006

Tabela 5.9: Influência do número de utilizadores que irão receber a actividade no desempenho da sua publicação. Resultados em segundos.

Esta análise revelou um compromisso semelhante ao do estado da técnica, quando se olha para os casos extremos: ou se torna a publicação rápida e a recuperação lenta com o Streama v2, ou o contrário com o Streama original.

Uma tentativa de melhorar o desempenho das soluções nos casos extremos revelou que é possível delegar a publicação de uma actividade para ser executada de forma assíncrona em *background*, evitando que um utilizador com muitos subscritores tenha de esperar, em média, 5 segundos no pior caso analisado (10000 subscritores). Sendo isto possível, o Streama original passa a ser a melhor opção, já que se torna melhor ou equivalente ao Streama v2 em todos os critérios.

A tabela 5.10 apresenta os resultados da comparação desta nova solução com o Streama original, para o critério tamanho da audiência. Para implementar este envio de operações para *background* utilizou-se o Delayed Job[7], uma biblioteca já utilizada pelo Bundlr e que integra com o HireFire.

Nº de utilizadores	**Streama**	**Delayed Streama**
100	0.0314 ± 0.0664	0.0173 ± 0.0404
1000	0.1303 ± 0.1319	0.0047 ± 0.0027
5000	1.0768 ± 0.2123	0.0657 ± 0.1890
10000	5.0337 ± 0.7784	0.0426 ± 0.1213

Tabela 5.10: Efeito da publicação de actividades de forma assíncrona, em *background*, através do uso da biblioteca Delayed Job. Resultados em segundos.

Tendo em conta a análise aqui descrita, a solução escolhida foi a junção da biblioteca Streama com a publicação assíncrona de actividades através da biblioteca Delayed Job.

Outros detalhes da implementação

Durante a implementação surgiram desafios relacionados com a escolha feita acima. O primeiro está relacionado com a capacidade de *caching* do Streama.

Além de guardar uma referência para os autores, objectos e alvos de cada actividade, permite também que sejam guardados os valores de quaisquer outros atributos do objecto, removendo assim a necessidade de os ir buscar à base de dados. Por exemplo, ao associar um utilizador ao campo actor de uma actividade é importante guardar também o seu *username*, nome completo e o link para a sua foto, facilitando assim a apresentação da actividade e aumentando bastante a sua performance.

O problema deste comportamento é que se o utilizador altera a sua foto ou mesmo o seu nome, as suas actividades não são actualizadas, sendo apresentada a foto e o nome antigos. Uma solução é a configuração de rotinas

[7] https://github.com/tobi/delayed_job

que corram diariamente e que actualizem os dados das actividades mais recentes de cada utilizador. Esta solução é bastante lenta por percorrer todos os utilizadores, mesmo aqueles que não tenham sido alterados, tornando-se impraticável a sua utilização. Outra solução bastante melhor é a utilização de um *Observer*[18], implementado pelo Mongoid, que actua quando o utilizador é alterado. Esta foi a solução implementada, garantindo assim a actualização das 100 actividades mais recentes de um utilizador cujo atributo nome ou foto tenha sido alterado.

Outro problema encontrado está relacionado com a publicação assíncrona. Ao atrasar a publicação da actividade para ser executada em *background* podem-se rapidamente encontrar dois problemas: o atributo data de criação passa a ter a data de publicação assíncrona e não a data real; se um elemento for eliminado antes da publicação da actividade podem acontecer erros. O primeiro resolve-se enviando a data real como argumento da rotina que corre em *background* e que coloca a data correcta ao publicar a actividade. A solução do segundo problema passa por testar a existência de cada elemento antes da publicação. Se o elemento não existir não se publica a actividade.

Deste segundo problema surge outra questão que vale a pena considerar: é necessário definir o que acontece quando um dos elementos da actividade é eliminado depois da sua publicação. Existem várias formas de resolver a questão, desde a utilização de um *Observer* para eliminar as actividades daquele elemento até mesmo a deixar as actividades intactas, sabendo que os seus links irão gerar um erro HTTP 404, quando o utilizador os segue.

A solução escolhida é uma mistura destas duas: utiliza-se um *Observer* para eliminar actividades de clips eliminados e mantêm-se as restantes intactas. A razão por trás da eliminação das actividades de clips é que este será o único tipo de actividade em que não serão apenas mostrados os atributos guardados em *cache* pelo Streama. De facto, a actividade publicada com a criação de um novo clip será apresentada com a visualização do próprio clip adicionado, para que o utilizador possa consumir esse conteúdo directamente na *activity stream*, o que não seria possível se o clip tivesse sido eliminado e daí a necessidade de eliminar as actividades correspondentes.

O Apêndice E.4 apresenta as alterações feitas neste módulo a nível dos modelos e controladores da arquitectura MVC da aplicação.

5.3.5 Testes

Testes Funcionais

Passaram **33** dos **33** cenários de teste derivados das *Acceptance Stories* descritas no Apêndice B.3.

Passaram **38** dos **38** testes unitários criados para este módulo. O *spec* final de testes unitários é listado no Apêndice L.

Testes de Desempenho

Segundo o requisito D.1, a publicação de actividades deve ser feita em menos de 200ms. Os resultados obtidos anteriormente para a solução escolhida (Delayed Streama), presentes na tabela 5.11 mostram que o número de utilizadores receptores não tem influência significativa na performance da solução. De facto, a perda de desempenho é de cerca de 2 vezes e meia no intervalo de 100 a 10000 subscritores, o que se torna desprezível. Já para 100 o tempo de execução é em média 17ms.

Nº de utilizadores	**Delayed Streama**
100	0.0173 ± 0.0404
1000	0.0047 ± 0.0027
5000	0.0657 ± 0.1890
10000	0.0426 ± 0.1213

Tabela 5.11: Resultados da publicação de actividades com a solução escolhida para um número crescente de subscritores. Resultados em segundos.

Tendo em conta que as médias ficaram todas abaixo dos 70ms, pode-se admitir que o servidor não demorará, em média, mais que 200ms a publicar uma actividade com até 10000 receptores.

O requisito D.2 define um limite máximo de 500ms para o processamento da *activity stream* de um utilizador. Como se discutiu anteriormente, é preciso ter em conta que a *activity stream* nunca será apresentada na sua plenitude, sendo sempre limitada por páginas de 30 actividades.

Assim, tendo em conta os resultados da solução implementada para a variação do número de actividades apresentados na tabela 5.12, admite-se que uma *query* limitada a 30 elementos demorará, em média, menos que a *query* com 100 actividades, que demora, em média, 50ms - dez vezes menos que o limite imposto pelo requisito.

Nº de actividades	**Delayed Streama**
100	0.0496 ± 0.0020
1000	0.5001 ± 0.0055

Tabela 5.12: Influência do tamanho da *activity stream* na sua construção para a solução implementada. Resultados em segundos.

5.4 Módulo 4: Categorização e Exploração de bundles

5.4.1 Motivação

O Bundlr tem-se apresentado, desde o início, como uma ferramenta para organização e partilha de conteúdo web. No entanto, a empresa sempre ambicionou ser mais que isso - tornar-se uma plataforma de conteúdo de qualidade seleccionado pelos seus utilizadores.

Com esse objectivo em mente, surge a necessidade de divulgar o conteúdo presente no serviço. Neste módulo pretende-se responder a essa necessidade.

5.4.2 Estado da Técnica

Foi feita uma análise da competição no que toca a funcionalidades de descoberta de conteúdo para identificar quais as mais comuns. Com o objectivo de determinar a abertura desses serviços à publicação e divulgação do seu conteúdo, registou-se de que forma tanto utilizadores não autenticados como autenticados podem descobrir esse conteúdo. Os resultados encontram-se na tabela 5.13.

A análise revelou a existência de 4 estratégias principais:

Pesquisa Facilita a recuperação de conteúdo através de termos específicos. Ideal para quando o utilizador sabe o que quer encontrar.

Conteúdo Popular Mostra conteúdo relevante no momento em que o utilizador está a explorar.

Categorias Ao dividir o conteúdo por categorias facilita-se a descoberta dentro das áreas de interesse do utilizador.

Conteúdo destacado Este conteúdo é seleccionado pela sua qualidade. A selecção pode ser manual (*staff-picked*) ou automática, baseada em regras estabelecidas pelos programadores do serviço.

Destas quatro estratégias, o Bundlr já suporta a pesquisa, bundles populares e bundles *staff-picked*. Estas duas últimas, no entanto, só estão disponíveis na *homepage* para utilizadores não autenticados. Ou seja, para aceder à lista de bundles populares e *staff-picked*, o utilizador teria de terminar a sua sessão no Bundlr. A figura 5.4 apresenta a secção da *homepage* que contém esses bundles populares e *staff-picked*.

Serviço	**Não autenticado**	**Autenticado**
Storify	*Story* em destaque *Stories* populares Tópicos em destaque Users em destaque	Populares de subscrições
Scoop.it	-	Pesquisa: tópicos e *posts* Ver mais tópicos: - maior *audience engagement* - mais vistos nos últimos 7 dias - escolhidos pela equipa - mais vistos de sempre
Evernote	-	-
Delicious	Pesquisa: *stacks*, links e tags Users em destaque *Staff-picked Stacks* Navegação por Categorias Ver todos os *Stacks*	Pesquisa (*stacks*, links e tags) Users em destaque *Staff-picked Stacks* Navegação por Categorias Ver todos os *Stacks*
Zootool	Links Populares	Ver todos os itens Items populares *Packs* populares Users em destaque Tags populares
Gimme Bar	-	Colecções em destaque Users em destaque Secção *Discovery*: - conteúdo subscrito

Tabela 5.13: Funcionalidades de descoberta de conteúdos nos serviços competidores

Alguns dos serviços analisados fazem o contrário: só expõem o conteúdo a utilizadores autenticados. No contexto do Bundlr, é relevante divulgar o conteúdo para esses dois tipos de utilizadores por várias razões: torna-se mais fácil perceber a mensagem do serviço ao ver algumas colecções criadas por outros utilizadores; incentiva a comunidade a criar conteúdo para ganhar mais visibilidade na *homepage*; permite a exploração de conteúdo também a utilizadores autenticados.

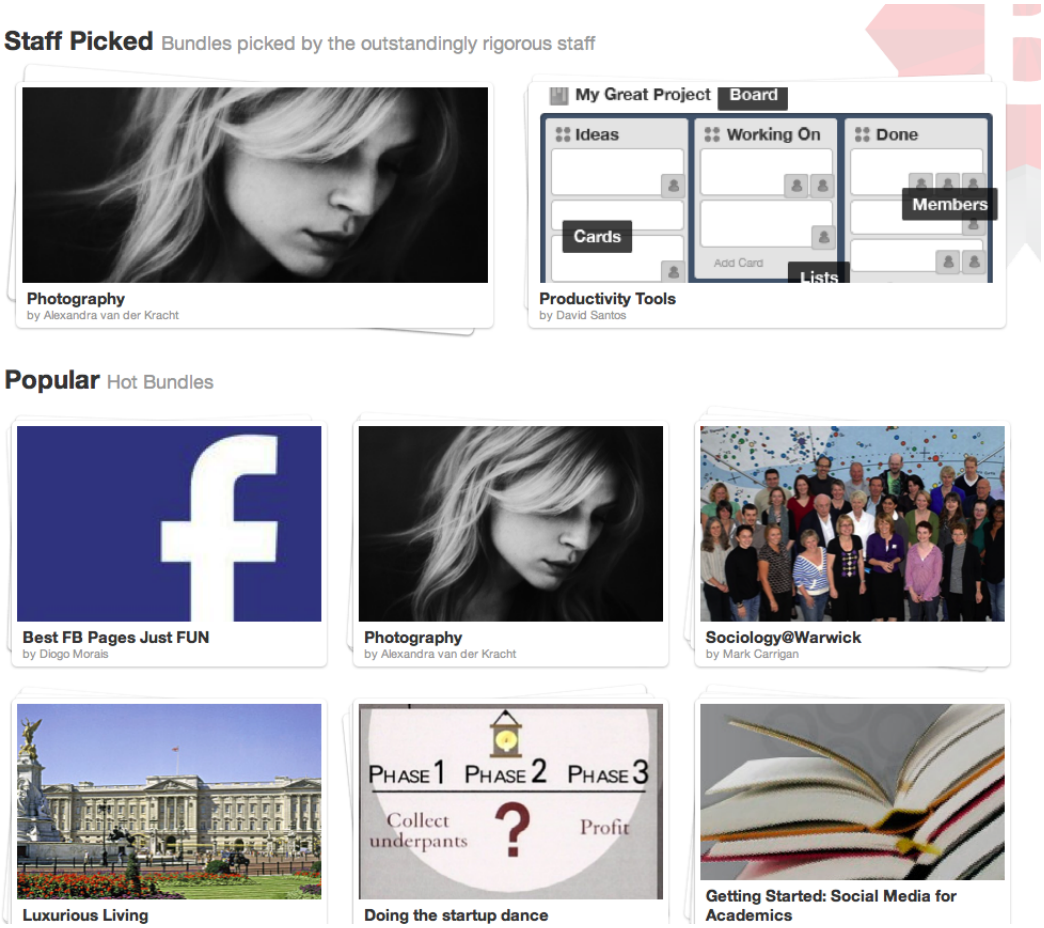

Figura 5.4: Secção de exploração de conteúdo na *homepage* original do Bundlr. Recolhida em Março de 2012.

Armazenamento das categorias

Neste módulo pretende-se, entre outras coisas, possibilitar a categorização manual de bundles. Esta informação pode ser guardada na base de dados de duas formas:

***String* "categoria" na colecção Bundle** O tempo de acesso e de alteração da categoria de um dado bundle é menor por não ter de ser feita uma *query* separada. No entanto, esta solução não é tão boa se se quiserem alterar dinamicamente as categorias que existem, já que seria necessária a alteração do código ou de um ficheiro de configuração.

Criar colecção "Categorias" separada Facilita a adição de novas categorias dinamicamente. É preferível se for necessário ter estatísticas e

dados individuais de cada categoria.

Como não se prevê que as categorias sejam mudadas frequentemente, optou-se pela primeira opção (guardar a categoria como um atributo na colecção Bundle), evitando também acessos extra à base de dados para recuperar a categoria de um bundle.

Categorias escolhidas

Foi feita uma recolha das categorias mais usadas por alguns serviços competidores de forma facilitar a escolha das categorias a incluir na implementação deste módulo. Apresenta-se de seguida a lista de categorias final:

- *Arts & Design*
- *Business*
- *Education & Science*
- *Entertainment*
- *Lifestyle*
- *News*
- *Places*
- *Technology*

A escolha das categorias pode ser importante, já que pode implicar que o utilizador não atribua nenhuma categoria ao seu bundle por haverem demasiadas escolhas ou por não encontrar a que pretendia. Para chegar a estas escolha foram listadas e comparadas as categorias de outros serviços e foi pedido aos membros da equipa que categorizassem alguns bundles para se analisar se a distribuição estava a ser equilibrada por todas as opções. Em Julho de 2012 já foram categorizados cerca de 5000 bundles, sendo a categoria mais utilizada a *Lifestyle*.

5.4.3 Análise de Requisitos

Depois da análise do estado da técnica decidiu-se implementar a categorização de bundles, como já foi referido, e uma secção do website à qual chamaremos página de exploração. Esta é a página onde o utilizador deve ir quando quer descobrir conteúdo novo e interessante. Além dos bundles *staff-picked* a

página contém também os bundles mais populares nos últimos dias em cada categoria e uma selecção de utilizadores destacados.

Além desta página implementaram-se durante este módulo as páginas de cada categoria, as funcionalidades de selecção e edição da categoria do bundle e notificações por email quando um bundle se torna popular.

Outro dos requisitos deste módulo é uma *interface* para que os administradores possam categorizar bundles antes do lançamento da página de exploração, evitando categorias vazias.

As *User Stories* resultantes desta análise encontram-se no apêndice B.4, com as respectivas *Acceptance Stories*.

Requisitos não-funcionais

Durante a análise de requisitos não foram encontrados requisitos de segurança, escalabilidade e desempenho específicos a este módulo.

5.4.4 Implementação

Atribuição de Categorias

A categorização de um bundle é uma acção manual, feita pelo autor do bundle ou por um dos administradores. O autor pode fazê-lo em dois momentos: quando cria o bundle e quando edita as suas propriedades através de um elemento da *interface* no qual escolhe uma das 8 opções.

Para evitar que a página de exploração de conteúdo tenha categorias vazias aquando do seu lançamento, é necessário fazer um trabalho de categorização prévia. Esse é um esforço conjunto dos administradores que, tal como descrito nas *User Stories* do apêndice B.4, pretendem categorizar cerca de 1000 dos melhores bundles do serviço, para garantir a presença de bom conteúdo durante o lançamento da página de exploração.

Para responder a esta necessidade, implementou-se o Categorizador (figura 5.5) - uma página de acesso protegido que escolhe aleatoriamente (obedecendo a alguns critérios) um bundle não categorizado e o apresenta ao administrador, dando-lhe a opção de o categorizar com uma das 8 categorias ou de passar ao bundle seguinte. Idealmente o administrador seria capaz de escolher a categoria só através do título, descrição e capa do bundle, podendo opcionalmente abri-lo para confirmar a sua escolha. Os critérios para a selecção dos bundles propostos são estes terem mais de 2 clips, mais de 10 visualizações e não serem privados. Esta é uma forma de garantir a qualidade mínima dos bundles categorizados previamente pelos administradores.

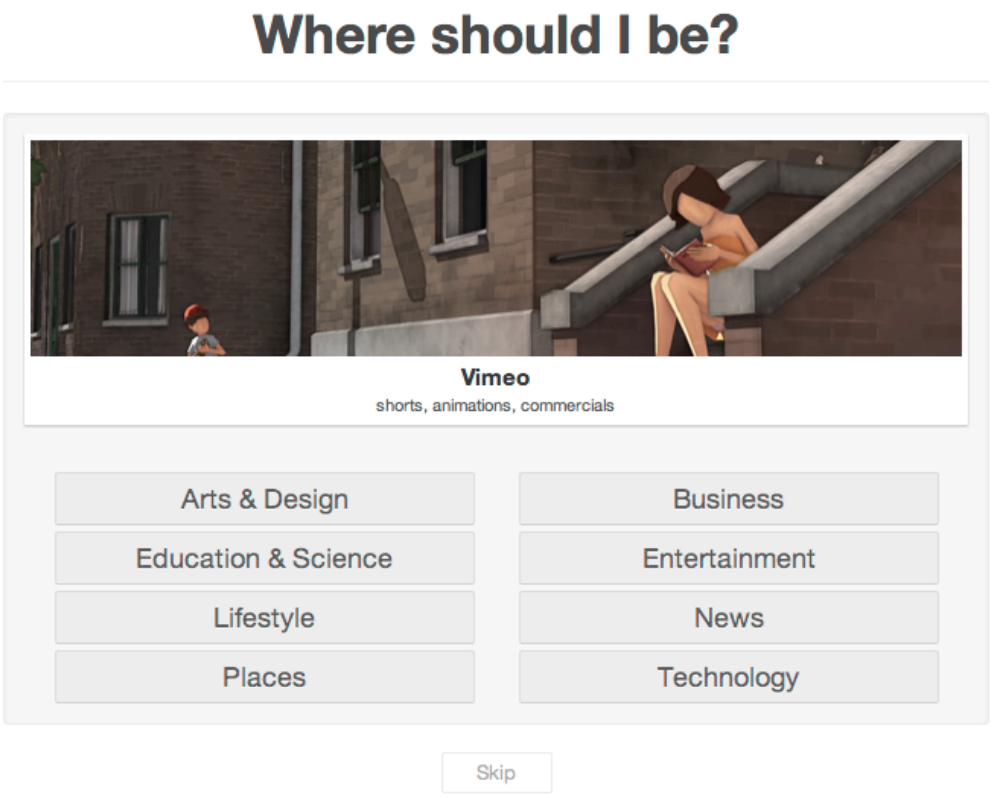

Figura 5.5: Categorizador - página onde os administradores podem categorizar bundles escolhidos aleatoriamente. Recolhida em Março de 2012.

Em termos técnicos, o Categorizador utiliza Ajax para categorizar os bundles e recuperar o próximo a analisar. Esta opção agiliza bastante a interacção com a página, tornando as acções mais rápidas já que a página não tem de ser recarregada totalmente a cada categorização.

A página de exploração

O propósito desta página é apresentar conteúdo interessante ao utilizador. Apresentam-se de seguida os conteúdos que a compõem:

***Staff-picked* bundles** Bundles de qualidade seleccionados pela equipa. Esta é a única secção desta página que já estava implementada no serviço antes do início deste módulo.

User em destaque A divulgação do perfil de um utilizador é uma das vantagens do plano pago. Desta forma, a escolha do utilizador em destaque

obedece à seguinte regra: se já existirem 15 utilizadores pagos, escolhe-se um aleatório entre eles; caso contrário, escolhe-se um entre os pagos e os utilizadores mais activos no último mês.

Listagem das Categorias Secção que apresenta os 4 bundles mais populares do momento em cada categoria. Faz-se também a ligação para as páginas das categorias, que efectuam a listagem de todos os bundles de cada uma. O mecanismo de selecção dos bundles populares é descrito de seguida.

A figura 5.6 apresenta a página de exploração, onde são visíveis os três elementos listados acima.

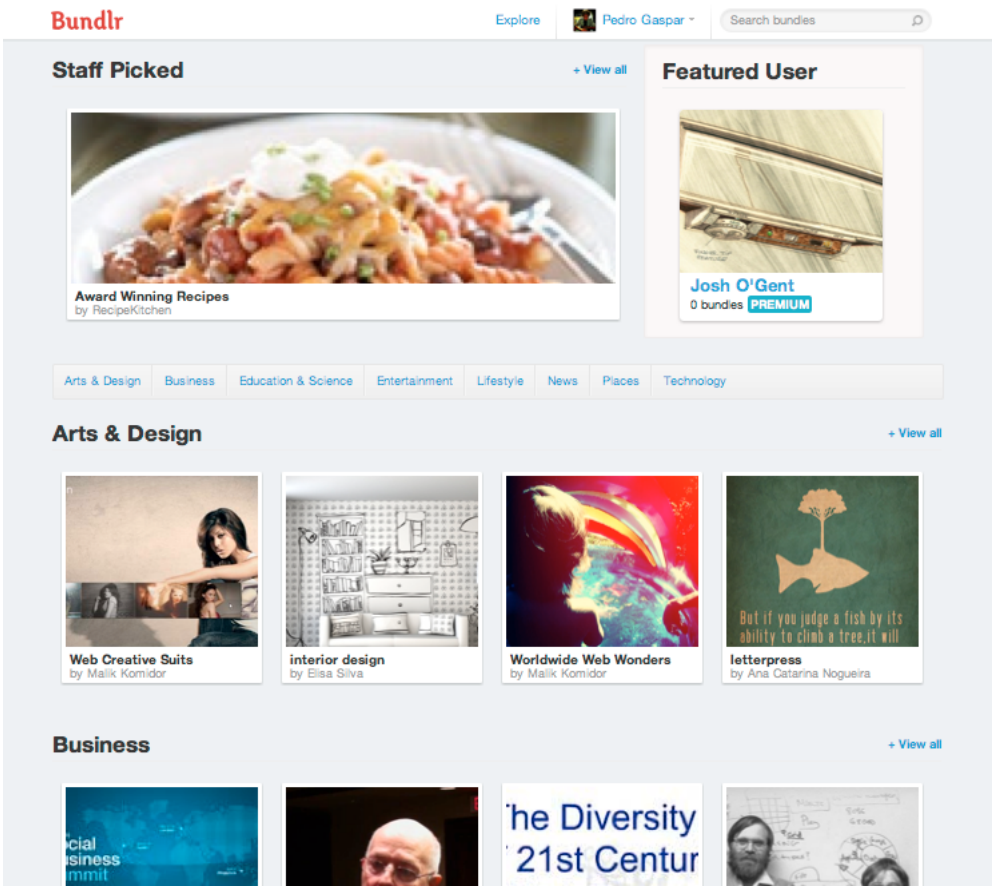

Figura 5.6: Página de Exploração. Recolhida em Março de 2012.

Selecção de bundles populares

A selecção dos bundles actualmente populares em cada categoria é uma tarefa importante, já que dita a qualidade do conteúdo que o utilizador vê na página de exploração. O cálculo da popularidade de um bundle já era feito antes deste módulo, tendo-se agora adicionado o número de subscrições do bundle ao cálculo. A popularidade é, então, calculada através da evolução do número de visualizações únicas e do número de subscritores do bundle nos últimos 4 dias. A fórmula que se segue representa uma simplificação do cálculo feito para obter a popularidade. O número de *visualizações* e *subscrições* considerados são apenas os dos últimos 4 dias.

$$popularidade = (visualizações + (subscrições \times 10)) \times bónus$$

A primeira parte da fórmula estabelece o valor de popularidade base do bundle. Note-se que uma subscrição tem o mesmo peso que 10 visualizações. A segunda parte adiciona um bónus caso haja uma diferença positiva entre a soma das visualizações e subscrições nos últimos dois dias e a mesma soma para o terceiro e quarto dias. O que se pretende é favorecer os bundles que estão agora a ser descobertos pela comunidade em detrimento dos que mantêm um fluxo constante de novas visualizações e subscrições. O bónus é calculado com base na diferença anterior e na soma dos valores para os últimos dois dias, tal como descrito na fórmula seguinte. Note-se que o bónus só é aplicado se tiver valor maior que 0.

$$bónus = (visualizações\ últimos\ 2\ dias + (subscrições\ últimos\ 2\ dias) \times 10)) - (visualizações\ dias\ 3\ e\ 4 + (subscrições\ dias\ 3\ e\ 4) \times 10))$$

O cálculo da popularidade é feito diariamente, para actualizar os valores associados a cada bundle na base de dados.

Definida a medida de popularidade, torna-se mais fácil seleccionar os bundles mais populares em cada categoria. No entanto, para garantir a sua qualidade, faz-se também uma filtragem dos bundles candidatos, excluindo os que:

- Já foram seleccionados anteriormente (um bundle só pode ser seleccionado uma vez);
- Não têm foto de capa;
- Têm um número de clips menor ou igual a 5;

- Têm valor de popularidade abaixo de 30;
- Foram previamente marcados pela equipa como tendo conteúdo não seguro.

O limite de popularidade 30 partiu de uma análise dos valores de popularidade dos bundles mais populares durante a implementação do módulo, necessitando de ser ajustado à medida que o serviço ganha mais visibilidade e utilizadores.

Filtrados os bundles de menor qualidade, falta só recuperar da base de dados o bundle com maior valor de popularidade para cada categoria. No caso de nenhum bundle corresponder aos critérios acima não é seleccionado nenhum bundle para essa categoria.

A marcação de um bundle para que seja mostrado na página de exploração é feita através da gravação da hora actual num dos seus atributos. Assim é possível saber se um bundle já foi alguma vez seleccionado e ordenar facilmente pela data em que foi seleccionado.

Esta selecção tem de ser feita diariamente para que a página de exploração tenha conteúdo novo frequentemente. Para resolver este problema foi implementado um *script* que é executado todos os dias, seleccionando os bundles populares de cada categoria da forma descrita acima.

Além disso, o *script* envia um email para os autores dos bundles seleccionados, avisando-os que estão a ser destacados na página de exploração.

O Apêndice E.5 apresenta as alterações feitas neste módulo a nível dos modelos e controladores da arquitectura MVC da aplicação.

5.4.5 Testes

Testes de Funcionais

Passaram **82** dos **82** cenários de teste derivados das *Acceptance Stories* descritas no Apêndice B.4.

Passaram **3** dos **3** testes unitários criados para este módulo. O *spec* final de testes unitários é listado no Apêndice L.

Note-se que este módulo teve um maior foco em funcionalidades de mais alto nível e daí a existência de muitos testes de aceitação comparando com os testes unitários.

5.5 Módulo 5: Sugestão de conteúdo

5.5.1 Motivação

Uma das formas de aumentar a subscrição de conteúdo é apresentar material relevante para o utilizador. A noção de conteúdo relevante é bastante subjectiva, já que pode ir desde conteúdo relacionado com pessoas na rede social do utilizador, a conteúdo popular no sistema.

Neste módulo pretende-se implementar um motor de sugestões personalizadas, que se baseie no conteúdo recolhido pelo utilizador para lhe sugerir bundles e pessoas para subscrever.

Embora possam surgir uma série de aplicações interessantes com um motor deste tipo, o foco deste módulo será principalmente a sua implementação, e não a sua aplicação prática. A demonstração do motor será feita através da sugestão de bundles e utilizadores para subscrever. Outros exemplos de aplicações práticas poderiam ser enviar um email semanal com novas sugestões, sugerir colaboradores para bundles, mostrar bundles semelhantes, entre outros que serão apresentados na análise de requisitos (secção 5.5.3).

5.5.2 Estado da Técnica

São vários os mecanismos de sugestão de conteúdo que se podem encontrar pelas aplicações web mais conhecidas, desde a sugestão de amigos do Facebook à sugestão de vídeos relacionados do YouTube e do Vimeo.

Uma breve análise dos serviços concorrentes mostra que apenas o Scoop.it e o Delicious têm algum tipo de sugestões. No primeiro caso, os próprios utilizadores podem sugerir conteúdo para as páginas temáticas de outras pessoas. Além disso, o próprio serviço apresenta os melhores resultados que encontra para o título da colecção em serviços como o Google, Twitter, YouTube, entre outros. Já o Delicious lista *Stacks* relacionadas na página de uma colecção.

Abordagens comuns ao problema de recomendação

O estudo de sistemas de recomendação é relativamente recente, comparando com outro tipo de técnicas e ferramentas, como motores de pesquisa ou bases de dados. O seu estudo surge de forma mais significativa em meados dos anos 90, tendo o interesse nesta área aumentado nos últimos anos, com o crescimento da Internet[19].

O principal desafio de um motor de recomendação é a tarefa de prever se um item se adequa aos gostos do utilizador e perceber se vale a pena

recomendá-lo. Burke[20] descreve 6 abordagens diferentes para este problema, que serão apresentadas resumidamente nesta secção do estado da técnica.

Collaborative Filtering

Esta abordagem baseia-se no princípio que se pessoa A tem a mesma opinião que a pessoa B em relação a um item, então é mais provável a pessoa A tenha opiniões semelhantes à pessoa B relativamente a outros itens, do que ter opiniões semelhantes a um outro utilizador escolhido aleatoriamente.

As recomendações são, então, calculadas com base no histórico de acções de todos os utilizadores, sugerindo conteúdos que foram aprovados (implícita ou explicitamente) por utilizadores semelhantes. Faz-se um trabalho de selecção de utilizadores semelhantes e usa-se o histórico de cada um para sugerir conteúdo aos outros.

Um exemplo prático desta abordagem é o serviço Last.fm[8] que gera listas de música através da comparação das bandas e músicas que o utilizador costuma ouvir com o comportamento de outros utilizadores. As listas contêm músicas que não foram ouvidas anteriormente pelo utilizador mas que são ouvidas frequentemente pelos seus semelhantes.

Este é um exemplo de *collaborative filtering* centrado no utilizador. Existem outros métodos, como o centrado em itens, visto pela primeira vez no website da Amazon.com[21], que proporciona recomendações do tipo "Utilizadores que compraram A também compraram B". Este método define uma matriz item-item que determina a relação entre pares de itens, que é usada em conjunto com os histórico do utilizador para inferir as suas preferências.

As acções do utilizador podem ser recolhidas de forma explícita ou implícita. A recolha é explícita quando o sistema pede ao utilizador para classificar um item, escolher o melhor de dois, organizar por preferência ou outras acções semelhantes. A recolha implícita, por sua vez, não pede uma classificação directa do utilizador mas infere os seus gostos através de acções como ouvir uma música, ver um item de uma loja (tendo em conta a duração dessa visita), comprar um produto, entre outros.

Existem vários mecanismos para a implementação deste tipo de sistema, sendo os dois principais grupos os *Memory-based* e os *Model-based*. O primeiro grupo de algoritmos usa as classificações de itens para inferir a semelhança entre utilizadores. Exemplos são a aplicação de algoritmos de análise da vizinhança, como o *K-Nearest-Neighbours*, ou algoritmos de selecção dos N utilizadores ou itens mais semelhantes, como nos exemplos acima.

[8] http://last.fm

O segundo grupo de algoritmos usa modelos gerados através de técnicas de *data mining* e *machine learning* para observar os padrões nas acções dos utilizadores e fazer previsões. Alguns exemplos dos algoritmos usados são redes bayesianas e algoritmos de *clustering*.

Um problema interessante que surge com o *collaborative filtering* tem o nome de *cold start problem*[22]. Como esta abordagem utiliza o histórico de um utilizador para fazer recomendações, um utilizador sem histórico não pode ter recomendações personalizadas. Isto acontece com utilizadores novos no sistema ou com itens adicionados recentemente. Algumas das soluções para este problema passam pela recomendação de itens não personalizados, como itens populares.

Abordagens *Content-based*

Estes sistemas consideram uma série de características de um item para sugerir itens com características semelhantes. Contrariamente ao *collaborative filtering*, não usa as opiniões de todos os utilizadores do sistema para recomendar um item, usando apenas o histórico do utilizador e os próprios itens que ele aprovou para sugerir itens semelhantes.

Um exemplo deste tipo de sistemas é o serviço Pandora Radio[9], que gera uma lista de música a partir dos atributos de uma música inicial. O serviço extrai esses atributos iniciais e selecciona uma série de músicas com atributos parecidos. Note-se que neste exemplo o utilizador escolhe a música inicial, ditando assim o tipo de música que quer ouvir. Seria também possível criar uma lista de música recomendada para aquele utilizador se se procurassem músicas com atributos semelhantes às que o utilizador marcou como favoritas.

Implementações desta abordagem modelam cada item com base num conjunto de atributos que o caracterizam, criando depois modelos do utilizador com base na importância que cada um dos atributos tem para si ou simplesmente na presença desses atributos nos itens do seu histórico. A importância dos atributos pode ser inferida através de *feedback* directo do utilizador, que diz, por exemplo, se gosta ou não da sugestão.

Uma desvantagem desta abordagem é o facto de ser dependente da complexidade dos itens recomendados. A extracção de atributos nem sempre é fácil, tornando-se complexa quando se pretende recomendar filmes, imagens ou mesmo música. As restantes abordagens são agnósticas ao item que se recomenda, utilizando outros factores para fazer a sugestão.

Por outro lado, ao contrário do *collaborative filtering*, esta abordagem não costuma ter o problema de *cold start*, já que para conseguir recomendar itens

[9] http://pandora.com

só precisa de um item inicial que indique o que o utilizador quer ver. No entanto, este problema pode existir se a recomendação for feita não com base nos itens favoritos do utilizador mas em itens que o próprio utilizador adicione ao sistema. Por exemplo, no contexto do Bundlr, se um utilizador não criar nenhum bundle nem fizer nenhuma acção não há forma de inferir os seus gostos, independentemente da abordagem ser *content-based* ou *collaborative filtering*.

Abordagens Demográficas

Esta abordagem assume que utilizadores com o mesmo perfil demográfico têm interesses semelhantes. Exemplos desta abordagem são a apresentação de sugestões tendo em conta a idade e o sexo do utilizador.

Estes sistemas são relativamente mais simples conceptualmente que os apresentados anteriormente, já que apenas se têm de mapear os itens a sugerir nos atributos demográficos considerados.

Abordagens *Knowledge-based*

Esta abordagem é principalmente usada em sistemas que gerem conhecimento, como por exemplo, um website de suporte técnico. Os itens são recomendados com base em conhecimento da área que indica quais são os mais indicados às necessidades do utilizador. Por exemplo, se o utilizador necessitasse de ajuda com um componente do seu computador, um item que podia ser recomendado era o manual de utilização desse componente.

Estes sistemas são divididos em dois tipos principais. Os primeiros são baseados em casos (*case-based*), enquanto que os segundos se baseiam em restrições (*constraint-based*).

Abordagens *Community-based*

Este tipo de sistemas faz recomendações com base nas preferências dos amigos do utilizador. É baseado na premissa que as pessoas tendem a confiar mais nas opiniões dos amigos do que nas opiniões de estranhos, mesmo que tenham gostos semelhantes.

Esta abordagem adquire e modela informação sobre a rede social do utilizador e sobre as preferências e classificações dos seus amigos. É uma abordagem que ainda se encontra numa fase inicial, tendo ganho interesse pela proliferação das redes sociais[23].

Tem, à partida, algumas limitações, como a eventualidade de não ser possível aceder à rede social do utilizador ou de o utilizador não ter registo de nenhum amigo que possa iniciar o processo de recomendação.

Abordagens Híbridas

Finalmente, estas abordagens propõem a combinação das abordagens anteriores de forma a anular os defeitos de cada uma.

Por exemplo, sabendo que os sistemas de *collaborative filtering* sofrem do problema de não conseguirem recomendar um novo item até ele ter classificações suficientes, pode-se implementar de uma abordagem *content-based* para agir neste caso, já que esta se baseia nos atributos do item, que normalmente estão presentes assim que este é adicionado.

Antes de aplicar estes conceitos ao Bundlr importa determinar o tipo de recomendação que se pretende conseguir. Para conseguir tirar o máximo partido do sistema de recomendações a implementar, decidiu-se que seria interessante poder criar recomendações dos pares de combinações existentes entre as três entidades principais no Bundlr: Utilizadores, Bundles e Clips.

Desta forma, seria possível obter sugestões de utilizadores, bundles ou clips para um utilizador, para um bundle e para um clip. Embora algumas destas sugestões não tenham uma aplicação interessante, existem outras como a sugestão de um clip a partir de outro que podem ter aplicações práticas, como por exemplo sugerir clips semelhantes quando um utilizador faz um clip para um bundle seu.

Analisando agora as abordagens apresentadas, e comparando especificamente o *collaborative filtering* com as *content-based*, observa-se que a comunidade do Bundlr ainda não é activa o suficiente para o *collaborative filtering* ter bons resultados. De facto, existem muitos utilizadores do serviço que estão desligados do resto da comunidade, não subscrevendo conteúdo de outras pessoas e usando o serviço de forma exclusivamente pessoal. Numa abordagem de *collaborative filtering* estes utilizadores e bundles nunca seriam recomendados, independentemente da sua qualidade, já que nenhum dos outros utilizadores interage com eles. Além disso, se um utilizador A faz um bundle sobre o tema "Apple" e outro sobre "Carros", não é seguro concluir que um utilizador que também goste de "Apple" vá ter interesse em "Carros", já que uma pessoa pode ter gostos bastante dispersos.

Uma abordagem *content-based* já se adequa melhor ao pretendido, tendo em conta que se baseia nas características do bundle, clip ou utilizador para sugerir outros semelhantes. Esta abordagem tem a vantagem de sugerir qualquer bundle ou utilizador da plataforma, independentemente de serem conhecidos pelo resto da comunidade ou não, desde que estes tenham atributos suficientes para poderem ser comparados.

Surge, no entanto, o problema de que atributos extrair de cada um dos

tipos de objectos a recomendar. Para começar a resolver este problema utilizou-se a seguinte simplificação: um utilizador é o conjunto dos seus bundles e o bundle é o conjunto dos seus clips. Este modelo simplifica a tarefa de extracção de atributos, tornando apenas necessária a extracção de atributos dos clips. Estes atributos, por sua vez, serão um conjunto de tags, termos de uma ou mais palavras que resumem o conteúdo dos clips. Na secção da implementação (5.5.4) abordam-se as opções de extracção destas tags dos clips.

No entanto, a abordagem ideal seria uma abordagem híbrida, que considerasse outros factores como a rede social do utilizador para lhe sugerir conteúdo, como é feito nos sistemas *community-based*. Esta opção foi posta de lado no contexto deste módulo, principalmente por causa das restrições de tempo a que este foi sujeito. É, no entanto, uma adição interessante como trabalho futuro.

Medidas de semelhança entre conjuntos de tags

Existem várias formas de medir a semelhança e distância entre itens diferentes. As mais comuns são a distância Euclideana, a distância de Manhattan e a distância de Mahalanobis[19]. Outro método comum é o cálculo da semelhança através do coseno do ângulo de dois vectores que representam os documentos a comparar, um método chamado *cosine similarity*.

O coeficiente correlação de Pearson mede o grau e direcção da correlação entre dois vectores, calculando um valor entre -1 (nenhuma correlação) e 1 (correlação total). Este coeficiente tem normalmente melhores resultados que métodos como a distância Euclideana, respondendo melhor a situações em que os dados não estão bem normalizados.

Para este módulo interessam especialmente as medidas de semelhança entre conjuntos de atributos binários, já que um clip ou tem ou não tem uma tag. Podiam ser aplicados outros métodos mais complexos como calcular a distância de Levenshtein entre pares de tags, mas além de aumentar o número de erros ("job" e "bob" teriam uma semelhança alta, por exemplo), torna a comparação mais lenta.

Os métodos específicos para calcular a semelhança entre vectores binários baseiam-se nas quantidades *M01*, *M10*, *M11* e *M00*, em que *M01* é o número de atributos em que o vector x teve valor 0 e o vector y teve valor 1, *M10* é o número de atributos em que o vector x teve valor 1 e o vector y valor 0, e por aí adiante.

Dadas essas quantidades, podem-se calcular alguns coeficientes de semelhança, sendo os mais utilizados apresentados de seguida:

***Simple Matching Coefficient*:** $SMC = \frac{M11+M00}{M01+M10+M00+M11}$. Este coeficiente pode ser descrito como o número de valores comuns sobre o número total de valores e é equivalente a uma percentagem dos valores comuns dentro do conjunto total de valores.

***Jaccard Coefficient*:** $JC = \frac{M11}{M01+M10+M11}$. Este coeficiente responde ao facto de nem sempre fazer sentido o cálculo da quantidade *M00*. Por exemplo, no caso da comparação de tags, não faz sentido calcular o número de tags que não pertencem a nenhum dos objectos, já que não acrescenta informação relevante ao cálculo que se pretende fazer.

***Dice Coefficient*:** $DC = \frac{2|X \cap Y|}{|X|+|Y|} = \frac{2 \times M11}{2 \times M11+M10+M01}$. O coeficiente de Dice pode ser convertido no coeficiente de Jaccard através da fórmula $JC = \frac{DC}{(2-DC)}$.

***Overlap Coefficient*:** $OC = \frac{|X \cap Y|}{min(|X|,|Y|)} = \frac{M11}{min(|X|,|Y|)}$. Este coeficiente é 1 se um dos conjuntos for um subconjunto do outro.

Algumas destas medidas não são as mais indicadas para o problema de comparação de conjuntos de tags. O *Jaccard Coefficient* é preferido ao *Simple Matching Coefficient*, pelas razões apresentadas na sua descrição. O *Overlap Coefficient* não é adequado já que não se considera que dois bundles, um que tem 10 tags e outro que tem 100 mas que inclui as 10 do primeiro, devam ter semelhança total.

Na secção da implementação (5.5.4) apresenta-se um estudo comparativo entre algumas medidas de semelhança consideradas.

5.5.3 Análise de Requisitos

Foi feita uma análise de quais seriam as aplicações mais interessantes do sistema de sugestões para avaliar a sua relevância para o serviço. Os resultados apresentam-se de seguida:

Conteúdo a seguir Sugestões de bundles e utilizadores a subscrever. Estes resultados podem ser apresentados numa página do serviço ou, por exemplo, enviadas num email semanal.

Conteúdo a adicionar a um bundle Sugerir clips a adicionar a um bundle com base no conteúdo de ambos. Não é fácil ter boas sugestões inicialmente, já que o bundle não tem conteúdo suficiente para ser comparado a outros clips (problema do *cold start*).

Bundles relacionados Sugerir bundles semelhantes a partir de um dado um bundle. Passa também por uma análise e comparação do conteúdo dos dois bundles.

Auto-categorização Atribuir automaticamente uma categoria ao bundle assim que tenha clips suficientes para se conseguir extrair a sua temática. Esta aplicação acaba por ser um pouco arriscada já que actua realmente sobre os atributos do bundle, alterando-os sem consentimento prévio do utilizador.

Sugerir Categoria Não atribuir a categoria automaticamente mas dar apenas uma sugestão da categoria que se pensa ser mais indicada. Tanto nesta como na opção anterior seria necessário modelar o item "Categoria" de forma a conseguir ligar cada uma com as tags que lhe pertencem.

Sugerir colaboradores A sugestão de colaboradores é uma aplicação bastante interessante, já que um dos interesses da equipa do Bundlr é aumentar a criação de bundles colaborativos.

Como já foi referido, estas aplicações poderiam ter recomendações ainda mais interessantes se fosse tido em conta o contexto da rede social do utilizador. Desta forma, nas sugestões de utilizadores a seguir poderiam ser recomendados, também, utilizadores que são seguidos por alguns dos amigos, por exemplo. Embora não seja obrigatório que as pessoas seguidas pelos amigos do utilizador sejam relevantes para ele, haverá vários casos em que são. Mais uma vez, a integração com uma abordagem *community-based* não foi incluída na implementação deste módulo.

Destas aplicações práticas, a única que faz parte dos requisitos será, então, a sugestão de bundles e utilizadores a subscrever, libertando tempo para a implementação do sistema de recomendação em si. As *User Stories* resultantes desta análise encontram-se no apêndice B.5, com as respectivas *Acceptance Stories*.

Requisitos não-funcionais

Durante a análise de requisitos não foram encontrados requisitos de segurança, escalabilidade e desempenho específicos para este módulo.

5.5.4 Implementação

A implementação deste sistema foi, sem dúvida, um dos maiores desafios do estágio. O grande número de opções para um grande número de problemas

tornou o sistema complexo. A especificidade da arquitectura do Bundlr e as tecnologias que utiliza foram também limitativas, tanto a nível de bibliografia específica como a nível de soluções estabelecidas no mercado. Nesta secção será apresentado o sistema e as diversas decisões tomadas durante a implementação de cada uma das suas partes.

Extracção de significado dos clips

A primeira das duas fases deste sistema está centrada no problema de extrair os atributos (neste caso as tags) que melhor representam o conteúdo de um dado clip. Este processamento é feito após a criação do clip, em *background* através do Delayed Job.

A principal dificuldade deste desafio é a diversidade do conteúdo dos clips. Como já foi dito anteriormente, os clips podem ser vídeos do YouTube ou do Vimeo, fotos do Flickr, apresentações do Slideshare ou até links para qualquer website da Internet. Como é possível construir um sistema genérico que consiga extrair tags de tantos tipos de conteúdo?

Existem algumas soluções como as *Meta Keywords* que foram pensadas para resolver este problema. Estas são um campo no cabeçalho de um ficheiro HTML que permite especificar palavras-chave relacionadas com a página. Mesmo parecendo uma boa ideia, menos de 40% da base de dados actual do Bundlr utiliza este campo (inferido de uma amostra de 1000 clips aleatórios). Além disso, não há qualquer garantia que as palavras-chave lá colocadas estão relacionadas com o conteúdo da página. Existem também websites que mantêm as mesmas *Meta Keywords* em todas as suas páginas, tornando-as irrelevantes para definir o conteúdo específico de cada página.

Soluções genéricas como as *Meta Keywords* não conseguem ser ideais em todos os casos, pelo que é necessário fazer compromissos e tentar melhorar as soluções para os tipos de conteúdo mais populares. Seguindo este raciocínio fez-se um estudo aos conteúdos da base de dados com o objectivo de descobrir quais os tipos de conteúdo mais populares no Bundlr. Os resultados são apresentados na tabela 5.14.

Esta tabela indica que a maior parte dos clips não são feitos nos websites explicitamente suportados (YouTube, Vimeo, etc.) mas sim em websites genéricos (indicado pelos links, imagens e selecções de texto). Esta divisão dá origem a duas formas de actuação, já que é diferente extrair tags de conteúdo genérico mas principalmente textual, do que de conteúdo maioritariamente visual (vídeos, imagens, apresentações, entre outros).

Outro aspecto que distingue os serviços suportados explicitamente dos links genéricos é que em serviços suportados como o YouTube, Vimeo e Flickr, o conteúdo é submetido por um utilizador desse serviço. Uma pesquisa rá-

Tipo de Clip	Quantidade	Percentagem
Link	34680	38.53%
Imagem	26929	29.92%
Vídeo do YouTube	9680	10.75%
Selecção de texto	9138	10.15%
Mensagem no Twitter	5001	5.56%
Foto do Flickr	2373	2.64%
Vídeo do Vimeo	1325	1.47%
Artigo na Wikipedia	496	0.55%
Apresentação do Slideshare	306	0.34%
Documento do Scribd	41	0.05%
Podcast Audioboo	32	0.04%
Vídeo do Flickr	10	0.01%

Tabela 5.14: Tipos de clip mais populares no Bundlr a 20 de Abril de 2012.

pida revela também que todos os serviços, excepto a Wikipedia, promovem a utilização de tags para descrever o conteúdo. Ou seja, a tarefa passa a ser encontrar essas tags criadas pelos utilizadores em vez de gerar tags que representem o conteúdo, assumindo que a classificação feita por eles será melhor que a inferência automática. Neste estágio, à tarefa de recuperar tags criadas pelos utilizadores nos serviços suportados explicitamente pelo Bundlr, dá-se o nome de *Tag Retrieval*.

Tag Retrieval

Recuperar as tags criadas pelos utilizadores no YouTube, Vimeo, e outros exige algum trabalho: cada um destes serviços tem a sua própria API, e utilizar o método de pesquisa das tags no HTML (*scrapping*) é descartado pela imprevisibilidade de resultados, já que basta o serviço alterar uma secção chave do HTML para este método deixar de funcionar.

O Embedly[10] é um serviço que disponibiliza, entre outras, a funcionalidade de recolher as tags de alguns serviços (incluindo os suportados pelo Bundlr), mas apenas no plano mais caro. O preço de 199 US$ por mês leva a que seja mais interessante financeiramente optar por outra solução.

Resta-nos apenas a opção de interagir directamente com as APIs dos 8 serviços suportados. Através da tabela 5.14 é possível identificar os serviços para os quais não vale a pena interagir com a API, por serem pouco utilizados. Os serviços suportados pelo *Tag Retrieval* são: YouTube, Flickr,

[10] http://embed.ly

Vimeo, Slideshare, Twitter e Wikipedia. A interacção com as APIs dos 4 primeiros é feita pelas seguintes bibliotecas, respectivamente: youtube_it[11], flickraw[12], vimeo[13] e slideshare[14]. As tags das mensagens do Twitter são extraídas através de uma expressão regular que procura termos com o caracter "#" no início, indicativo de uma *hashtag*[24], a forma de colocar tags numa mensagem. Os artigos da Wikipedia são deixados para serem processados da mesma forma que os links genéricos por terem tags pouco específicas.

Resolvido o problema de *Tag Retrieval*, falta encontrar forma de lidar com os websites genéricos. Considerem-se as seguintes opções:

Meta Keywords Já descritas anteriormente, têm o problema de não serem adoptadas correctamente pela maioria dos websites.

Tag Extraction Extracção de palavras-chave através da análise do conteúdo textual do website. Obriga à extracção do texto, uma tarefa chamada *Boilerplate Removal*, que pode introduzir erros no processo.

Usar outro serviço como intermédio Aproveitar o facto dos utilizadores de serviços como o Delicious atribuírem tags manualmente aos links que guardam e tentar obter essas tags para cada link do Bundlr. Tem as desvantagens de estar dependente da base de dados de outro serviço e de não ser uma utilização "correcta" da API de um competidor.

A opção escolhida foi a *Tag Extraction*, por utilizar o conteúdo do website na sua análise. Existem várias formas de fazer esta análise que serão discutidas de seguida.

Tag Extraction

A análise feita aos websites genéricos será centrada em websites com texto, como artigos ou blogs. Existem várias formas de extrair palavras-chave de secções de texto:

Análise de frequência Algoritmos que escolhem as palavras-chave do texto com base no número de vezes que são repetidas. Podem ou não aplicar *stemming* e remover *stopwords*.

[11] https://github.com/kylejginavan/youtube_it
[12] http://hanklords.github.com/flickraw/
[13] https://github.com/matthooks/vimeo
[14] https://github.com/icebreaker/Super-Mega-Slideshare

APIs de extracção de entidades Estas APIs comerciais fazem um análise bastante completa do texto na óptica do *information retrieval* e da análise linguística. Existem várias, como a AlchemyAPI[15]. No entanto, todas elas são ou demasiado caras ou limitativas para uso comercial. A título de exemplo, a AlchemyAPI apenas disponibiliza 1000 chamadas por dia grátis, sendo o próximo nível 10000 chamadas por dia com um preço de 400 US$/mês.

Uso de motores de processamento de linguagem Utilização de motores como o Apache OpenNLP[16], Carrot2[17] ou NLTK[18] para analisar e extrair entidades e informação do texto. A maioria destes motores não são implementados em Ruby, o que dificulta qualquer tentativa de integração.

A decisão tendeu sobre os algoritmos de análise de frequência, pela sua simplicidade e relativa facilidade de implementação. Foram analisadas as seguintes bibliotecas de análise de frequência:

Phrasie[19] Biblioteca em Ruby que ordena os termos pelo número de repetições no texto e com base na sua análise linguística. Tem também suporte para termos com duas ou mais palavras. Das três é a que tem maior dificuldade em lidar com *encoding*.

Pismo[20] Biblioteca em Ruby que permite *stemming* e a remoção de *stopwords*. Só considera palavras individuais. Contém também um sistema de *Boilerplate Removal*, que será discutido mais à frente.

Existem mais bibliotecas deste género em Python e outras linguagens, como é o caso da Topia.TermExtract[21], no qual o Phrasie é baseado. No entanto, por existirem também opções em Ruby que facilitam a integração no projecto, tomou-se a decisão de apenas considerar bibliotecas em Ruby.

O Apêndice I faz uma análise mais aprofundada das diferentes formas de extracção de tags a partir de texto.

No Apêndice K apresenta-se um estudo comparativo entre as soluções Phrasie, Pismo e as *Meta Keywords*, incluindo algumas soluções híbridas.

[15] http://alchemyapi.com
[16] http://opennlp.apache.org
[17] http://project.carrot2.org
[18] http://nltk.org
[19] https://github.com/ashleyw/Phrasie
[20] https://github.com/peterc/pismo
[21] http://pypi.python.org/pypi/topia.termextract

Segundo o resultado deste estudo, a solução escolhida para a extracção de tags a partir do texto foi uma junção da biblioteca Pismo com as tags multi-palavra extraídas pela biblioteca Phrasie.

Boilerplate Removal

Como foi referido anteriormente, este tipo de algoritmos apresenta formas de extrair o texto de artigos de páginas HTML, removendo todos os elementos que não fazem parte do texto do artigo em si, como os menus de navegação e a secção de comentários[25].

Existem também várias soluções para o problema do *Boilerplate Removal*. O Apêndice J apresenta alguns dos métodos mais utilizados e descreve sucintamente o funcionamento de 3 soluções: Boilerpipe[22], Readability[23] e o Pismo.

A dificuldade de integração de sistemas como o Boilerpipe (Java) na arquitectura Ruby do Bundlr volta a ser um factor importante, inviabilizando muitas das soluções mais utilizadas.

O Readability é uma das soluções mais conhecidas por ter o seu código livre e transcrito para uma série de linguagens do *script* original em Javascript. Tem também uma versão Ruby sob a biblioteca ruby_readability[24].

Uma funcionalidade recentemente adicionada ao Bundlr teve também de resolver o problema de *Boilerplate Removal*, pelo que já existe uma solução implementada no sistema. Esta solução é uma versão modificada do Readability e é também considerada no estudo do Apêndice K, que para além de comparar algumas soluções de *Tag Extraction*, compara também alguns algoritmos de *Boilerplate Removal*. Esta versão está optimizada para funcionar melhor nos websites mais guardados no Bundlr.

Os resultados desse estudo apontaram, a já referida solução híbrida Pismo/Phrasie e a utilização do algoritmo Readability (tanto a versão original como a alterada pela equipa do Bundlr) como as mais apropriadas para o problema de *Tag Extraction*.

Limpeza e armazenamento das tags

Resolvido o problema da extracção de tags de um clip por *Tag Retrieval* ou *Tag Extraction*, restam alguns problemas para resolver antes de ser possível sugerir conteúdos.

[22] http://code.google.com/p/boilerpipe
[23] http://code.google.com/p/arc90labs-readability
[24] https://github.com/iterationlabs/ruby-readability

Antes de mais, é feita uma operação de limpeza das tags que remove espaços a mais, passa as palavras para letra minúscula e remove termos com menos de 3 caracteres.

Como foi referido no estado da técnica, assume-se que o utilizador é a soma dos seus bundles e que os bundles são a soma dos seus clips. Aplicando este conceito, decidiu-se passar todas as tags de um clip para o seu bundle e para o seu autor. Desta forma, garante-se uma maior quantidade de informação nos utilizadores e bundles, melhorando o processo de cálculo de semelhança entre objectos.

Depois da operação de limpeza é necessário guardar as tags na base de dados. Apresentam-se de seguida as duas formas principais de o fazer e as suas vantagens e desvantagens:

Guardar as tags nas colecções Clip, Bundle e Utilizador Pode aumentar consideravelmente o tamanho dos documentos destas colecções, já que, se cada clip tiver 10 tags e cada bundle 10 clips, bastava um bundle para o autor ter mais 100 tags no seu registo. Por outro lado, esta opção torna o acesso às tags mais rápido, assim que se tiver o documento em questão.

Guardar as tags numa colecção à parte Evita carregar as tags quando não são necessárias, não prejudicando o desempenho global do serviço.

Foi decidido guardar as tags numa colecção à parte, evitando uma perda considerável de desempenho no acesso ao resto do sistema.

Finalmente, foi criado um *script* para obter as tags dos cerca de 80000 clips existentes na base de dados, com o objectivo de ser corrido antes do lançamento do motor de sugestão.

O motor de Sugestão

O resultado da fase anterior é uma série de itens com os seus atributos associados, as tags. Estes objectos podem ser Clips, Bundles ou Utilizadores.

Neste contexto, o problema da sugestão pode ser descrito como a procura dos objectos cujas tags se aproximam mais do objecto para o qual se estão a gerar sugestões. Ou seja, se quiser encontrar os bundles mais parecidos com o utilizador A, preciso de comparar as tags desse utilizador com as tags de todos os bundles, para os poder ordenar por semelhança.

No estado da técnica foram apresentadas algumas medidas de semelhança entre vectores binários. Dessas, implementou-se o *Jaccard Coefficient*, que é semelhante a uma percentagem das tags em comum entre os dois itens. Foi

também testada uma variação deste coeficiente que dá importância a tags repetidas e uma soma ponderada para tags multi-palavra.

Percentagem de tags em comum Esta abordagem é baseada no *Jaccard Coefficient* e dá vantagem a conjuntos com muitas tags em comum. O cálculo de semelhança é feito pela fórmula apresentada no estado da técnica: $JC = \frac{M11}{M01+M10+M11}$. Por exemplo, aplicando esta fórmula, os conjuntos ["a", "a", "a", "d"] e ["a", "a", "b", "c"] teriam *semelhança*= $\frac{1}{4} = 25\%$. Note-se que este coeficiente utiliza conjuntos sem repetições de tags nos seus cálculos.

Percentagem de tags em comum incluindo repetições Abordagem semelhante à anterior mas tendo em conta tags repetidas. Espera-se com isto aumentar o resultado de semelhança entre conjuntos de tags com várias tags repetidas em comum, admitindo que tags repetidas significa uma maior quantidade de conteúdo sobre essa temática. O cálculo é feito da seguinte forma: *S = número de tags comuns / (número de tags total - número de tags comuns)*. Aplicando a fórmula aos conjuntos ["a", "a", "a", "d"] e ["a", "a", "b", "c"] teriam *semelhança*= $\frac{2}{(8-2)} = \frac{1}{3} = 33.3(3)\%$.

Soma ponderada para tags multi-palavra Abordagem baseada na primeira mas que não se limita a comparar as tags uma a uma, comparando antes as palavras correspondentes a cada tag. Tenta-se que as tags "bin laden" e "osama bin laden" não contem como tags diferentes só por não serem exactamente iguais. Esta abordagem só faz sentido quando as tags possam ter mais do que uma palavra, o que não é o caso da biblioteca Pismo.

O estudo do Apêndice K apresenta também uma comparação de cinco destas métricas, incluindo algumas referidas no estado da técnica. Embora os resultados não sejam muito diferentes, a escolha final foi a opção "Percentagem de tags em comum incluindo repetições", por ser um método relativamente simples e por ter em conta repetições de tags.

O cálculo das sugestões

Para obtermos sugestões de bundles para um utilizador A precisamos, então, de comparar as tags desse utilizador com todos os bundles existentes. Uma tarefa deste nível torna-se impossível de concretizar em tempo real durante o pedido ao servidor, pelo que é necessário fazer estes cálculos previamente, em *background*. O cenário ainda se torna mais grave se se quiser calcular

sugestões para todos os utilizadores do sistema, já que teríamos de comparar cada um dos cerca de 14.000 utilizadores com 15.000 bundles. Se não se tiverem alguns cuidados de optimização, esta operação pode demorar mais de um dia a completar.

Foram então criados dois *scripts*, um que corre diariamente e outro semanalmente. O *script* diário tem a particularidade de apenas calcular sugestões para os utilizadores que foram alterados desde o último cálculo. Existir uma alteração no utilizador sugere que este é um utilizador activo, aumentando a importância de apresentar sugestões actualizadas. O *script* semanal encarrega-se de actualizar as sugestões de todos os utilizadores.

A nível de optimizações, são necessários alguns cuidados como guardar os utilizadores e bundles a considerar em memória (dentro dos limites do sistema), para que os *scripts* não estejam constantemente a aceder à base de dados para ir buscar os mesmos dados. Além disso, limita-se também o número de sugestões em cada objecto - neste momento estão-se a calcular 30 sugestões de utilizadores e de bundles para cada utilizador. Evitam-se também comparações com objectos sem tags, já que o cálculo de semelhança com esses objectos dará sempre 0.

Para ser possível atribuir sugestões aos modelos Clip, Bundle e Utilizador de forma genérica, implementou-se essa funcionalidade num módulo comum, o *Suggestion::HasSuggestions*. Este módulo permite o cálculo de sugestões entre estes 3 modelos, utilizando as propriedades não relacionais do MongoDB a seu favor para criar atributos dinamicamente. Através do método *save_suggestions*, que recebe um *array* de objectos sugestão e uma *string* que identifica a classe desses objectos, pode-se então guardar sugestões de qualquer tipo de modelo que contenha tags. No caso de serem sugestões de clips, por exemplo, seria criado um campo *clip_suggestions* para as guardar (incluindo também as suas pontuações de semelhança) e um campo *last_clip_suggestions* que guarda um registo temporal de quando foram calculadas as últimas sugestões deste tipo.

Finalmente, o módulo inclui também alguns métodos de suporte para calcular, por exemplo, se um objecto foi alterado desde o último cálculo de sugestões de clips, de bundles ou de utilizadores.

A figura 5.7 apresenta o resultado da implementação das *User Stories*, que descreviam apenas as funcionalidades de sugestão de utilizadores e bundles a subscrever.

O Apêndice E.6 apresenta as alterações feitas neste módulo a nível dos modelos e controladores da arquitectura MVC da aplicação.

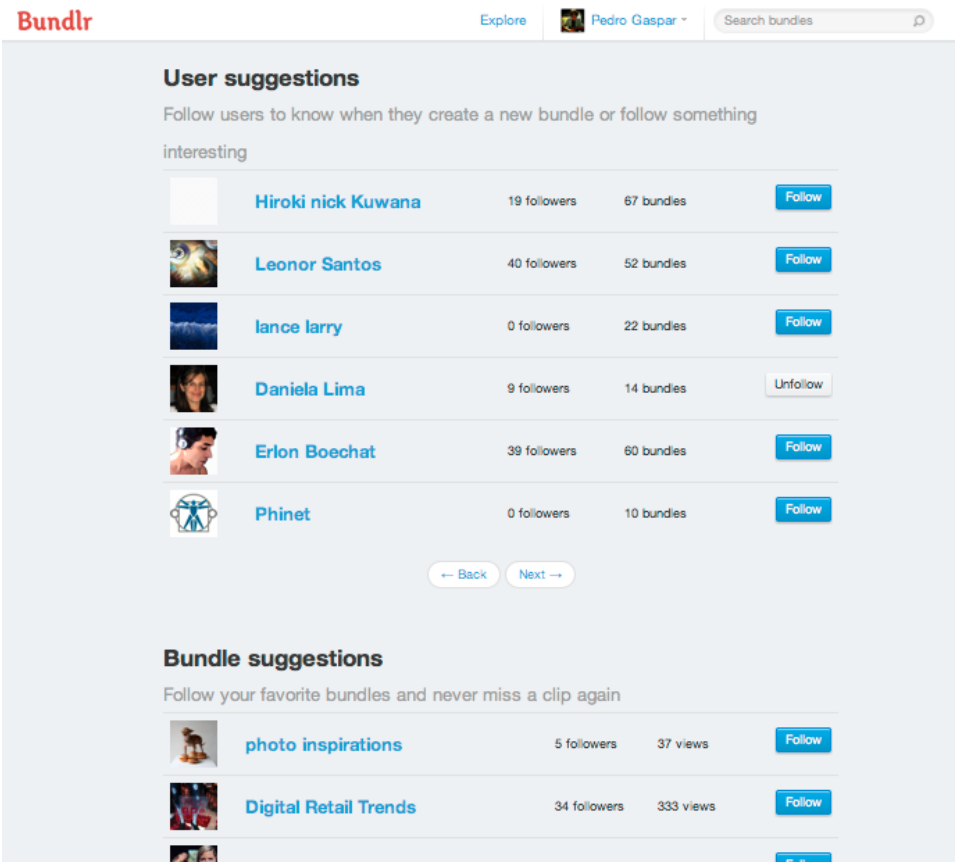

Figura 5.7: Sugestão personalizada de utilizadores e bundles a subscrever. Recolhida em Maio de 2012.

5.5.5 Testes

Testes de Funcionais

Passaram **8** dos **8** cenários de teste derivados das *Acceptance Stories* descritas no Apêndice B.5.

Passaram **68** dos **68** testes unitários criados para este módulo. O *spec* final de testes unitários é listado no Apêndice L.

Mais uma vez, observa-se o foco deste módulo no desenvolvimento do sistema de sugestões, acabando por serem criadas poucas *User Stories* e, consequentemente, poucos testes de aceitação.

5.6 Módulo 6: *Full-Text Search*

5.6.1 Motivação

A pesquisa é fundamental num serviço com muitos conteúdos. Além de ser uma forma de descobrir novo conteúdo, tal como discutido no módulo 4 (5.4.2), é uma forma de aceder rapidamente a algo existente no serviço. À medida que aumenta a quantidade de conteúdo aumentam também as necessidades de desempenho do sistema de pesquisa.

Originalmente, a pesquisa implementada no Bundlr era uma simples pesquisa sequencial pelos campos título e descrição da colecção Bundle, implementada com recurso a uma operação de MongoDB que compara estes campos com uma expressão regular que representa os termos da pesquisa. Além de ser uma pesquisa muito pouco versátil (não tinha em conta os conteúdos dos clips), é pouco eficiente. Se considerassemos, por exemplo, uma pesquisa por "Apple", o MongoDB iria percorrer os títulos e descrições de todos os bundles para tentar encontrar uma ocorrência. A *Full-Text Search* é uma abordagem diferente, pensada para resolver o problema da pesquisa em grandes extensões de texto.

O principal objectivo deste módulo é possibilitar a pesquisa não só pelo título e descrição do bundle mas também pelo próprio conteúdo dos clips. Essa pesquisa é possível porque o conteúdo extraído passou recentemente a ser guardado na base de dados para ser apresentado aos utilizadores, de forma a evitar uma ida ao *website* original. Inicialmente, os clips apenas mantinham uma pequena descrição do conteúdo. Esta mudança não fez parte do contexto do estágio e foi implementada pela restante equipa.

A implementação de *Full-Text Search* é uma forma de responder ao aumento do universo de pesquisa e às necessidades de desempenho do serviço. A desvantagem desta abordagem é ser tecnologicamente mais complexa, exigindo a manutenção de índices e eventualmente de servidores extra, como se discute de seguida.

5.6.2 Estado da Técnica

A abordagem da *Full-Text Search* passa pela indexação invertida das palavras presentes em cada documento. Cada uma das palavras do documento torna-se chave do índice, tendo como valor o ID do documento a que pertencem e, eventualmente, outros dados auxiliares, como a sua posição no documento. À medida que estas palavras vão sendo identificadas noutros documentos, adicionam-se esses novos IDs à lista de valores.

Isto leva a que hajam duas tarefas fundamentais para qualquer motor *Full-Text Search*:

Construção do Índice Analisa e extrai os termos de cada documento e indexa-os como explicado anteriormente. Este processo pode fazer algumas operações como a remoção de *stopwords*, *stemming* ou normalização da capitalização dos termos, entre outras. É um processo demasiado demorado para ser feito em cada pesquisa, portanto tem de ser feito previamente.

Pesquisa O processo desencadeado pelo utilizador. É feita a navegação pelo índice procurando os termos da pesquisa e devolvendo os IDs dos documentos em que se encontram. Nesta fase devem ser feitas operações de normalização dos termos de pesquisa semelhantes às da indexação, para que estes possam ser comparáveis com os termos indexados.

O aumento de desempenho desta abordagem na fase de pesquisa deve-se, fundamentalmente, à preparação prévia do índice, o que reduz um problema de pesquisa sequencial a um problema de pesquisa numa estrutura semelhante a uma Tabela de Dispersão.

Full-Text Search em MongoDB

Neste módulo pretende-se encontrar uma solução que possibilite este tipo de pesquisa, dando-se preferência a soluções que sejam compatíveis com a *framework* de desenvolvimento, com o sistema de base de dados e com a plataforma onde está alojado o serviço. Assim, começou-se por verificar se o MongoDB inclui algum tipo de suporte de origem para *Full-Text Search*.

A funcionalidade que mais se aproxima ao pretendido são os campos *Multikey*, que permitem a indexação de valores de um *array*. Embora seja possível utilizá-los para implementar uma versão muito simples de *Full-Text Search* em MongoDB[26], a ausência de funcionalidades de processamento dos termos de pesquisa e de ordenação de resultados inviabiliza a sua utilização.

Excluída a possibilidade de manter o índice na base de dados actual, voltou-se o foco da análise para sistemas dedicados, procurando um que obedeça às condições referidas anteriormente e que não penalize o sistema por falta de suporte a funcionalidades base de *Full-Text Search*, como foi o caso do MongoDB.

Outras opções de *Full-Text Search*

Fora do contexto do Ruby on Rails e do MongoDB existem várias soluções já estabelecidas no mercado como o Apache Lucene[25] e o Sphinx[26], motores implementados em Java e C++, respectivamente. O Apache Solr[27] é outra solução que implementa um servidor HTTP sobre o Apache Lucene, disponibilizando uma API *REST-like* para aceder aos seus métodos. No caso do Sphinx, o próprio sistema já tem à partida um servidor HTTP para facilitar acesso aos seus métodos. O acesso por HTTP é fundamental para facilitar a comunicação entre o sistema principal e o sistema de pesquisa, já que é agnóstico às linguagens de programação utilizadas em cada um dos lados.

Além destes sistemas, existem bases de dados não específicas ao problema de *Full-Text Search* com bom suporte de origem para este tipo de pesquisa, como o PostgreSQL[27] e o MySQL[28].

Integração com o Ruby

Existem bibliotecas de integração com a linguagem Ruby tanto para o Apache Solr como para o Sphinx, sendo as mais conhecidas o Sunspot[28] e o Thinking-Sphinx[29], respectivamente. Estas bibliotecas tornam trivial a configuração e comunicação com os servidores de pesquisa, removendo a necessidade de comunicar com as APIs HTTP directamente.

Para garantir a compatibilidade com a arquitectura do Bundlr, é necessário que estas bibliotecas suportem MongoDB e, mais especificamente, Mongoid. Isto é fundamental porque estas têm de ir buscar o texto a indexar à base de dados MongoDB através do Object-Document-Mapper utilizado - Mongoid. A indexação é feita pelo servidor de pesquisa, portanto é necessário enviar o texto a indexar através das APIs HTTP.

O suporte a Mongoid pode ser adicionado à biblioteca Sunspot através da biblioteca sunspot_mongoid[30]. No que toca ao Sphinx e ao Thinking-Sphinx, não é tão fácil encontrar uma biblioteca que adicione o suporte pretendido, sendo necessária a substituição do Thinking-Sphinx pelo Mongoid-Sphinx[31], que não tem tanto suporte pela comunidade.

[25] http://lucene.apache.org
[26] http://sphinxsearch.com
[27] http://lucene.apache.org/solr
[28] https://github.com/sunspot/sunspot
[29] http://freelancing-god.github.com/ts/en
[30] https://github.com/Empact/sunspot_mongoid
[31] https://github.com/redbeard-tech/mongoid-sphinx

Integração com o Heroku

Tendo já algumas opções que suportam Ruby on Rails e MongoDB, falta analisar a compatibilidade com o Heroku - a plataforma que aloja o serviço. Na prática, o que se pretende é conseguir configurar facilmente a solução final, evitando se possível a instalação manual de um servidor de pesquisa dedicado, o que obrigaria a equipa do Bundlr a manter e ajustar o servidor para o ambiente de utilização.

O Heroku suporta 3 opções principais de integração[29]:

Flying Sphinx[32] Um serviço de alojamento e manutenção de servidores Sphinx.

WebSolr[33] O equivalente ao Flying Sphinx para servidores Apache Solr.

Servidor próprio Instalar um dos sistemas de pesquisa num servidor próprio e configurar tanto o servidor como a ligação ao Heroku manualmente.

Descartando por agora a última opção pelas razões descritas acima, restam-nos os dois primeiros serviços. Como já foi referido, tanto o Apache Solr como o Sphinx são ferramentas já estabelecidas nesta área. Ambos são suficientemente avançados para suportar as funcionalidades base exigidas de um motor de *Full-Text Search*. Desta forma, a decisão acabou por ser feita com base nos preços de cada uma das soluções. O Apêndice H resume essa análise, que determinou que o WebSolr é a solução que mais compensa.

Desta forma, as tecnologias utilizadas durante este módulo são: WebSolr (Apache Solr), Sunspot e Sunspot Mongoid.

5.6.3 Análise de Requisitos

Como foi referido na introdução (secção 5.6.1), pretende-se implementar uma pesquisa de conteúdo. Ou seja, uma pesquisa que não se limite a procurar no título e descrição dos bundles mas também no conteúdo dos clips. É importante, então, definir o tipo de pesquisa mais vantajoso para os utilizadores e a forma como os resultados serão apresentados, para ser possível depois adaptar a implementação às necessidades identificadas.

A figura 5.8 apresenta o resultado de uma pesquisa por "riots"no sistema de pesquisa original. Uma comparação com os resultados da *Full-Text Search*, apresentados na figura 5.9 mostram a vantagem deste tipo de pesquisa.

[32] http://flying-sphinx.com
[33] http://websolr.com

Enquanto a original apenas foi capaz de encontrar 7 bundles com o termo da pesquisa no título ou na descrição, a *Full-Text Search* encontrou 302 clips em 120 bundles.

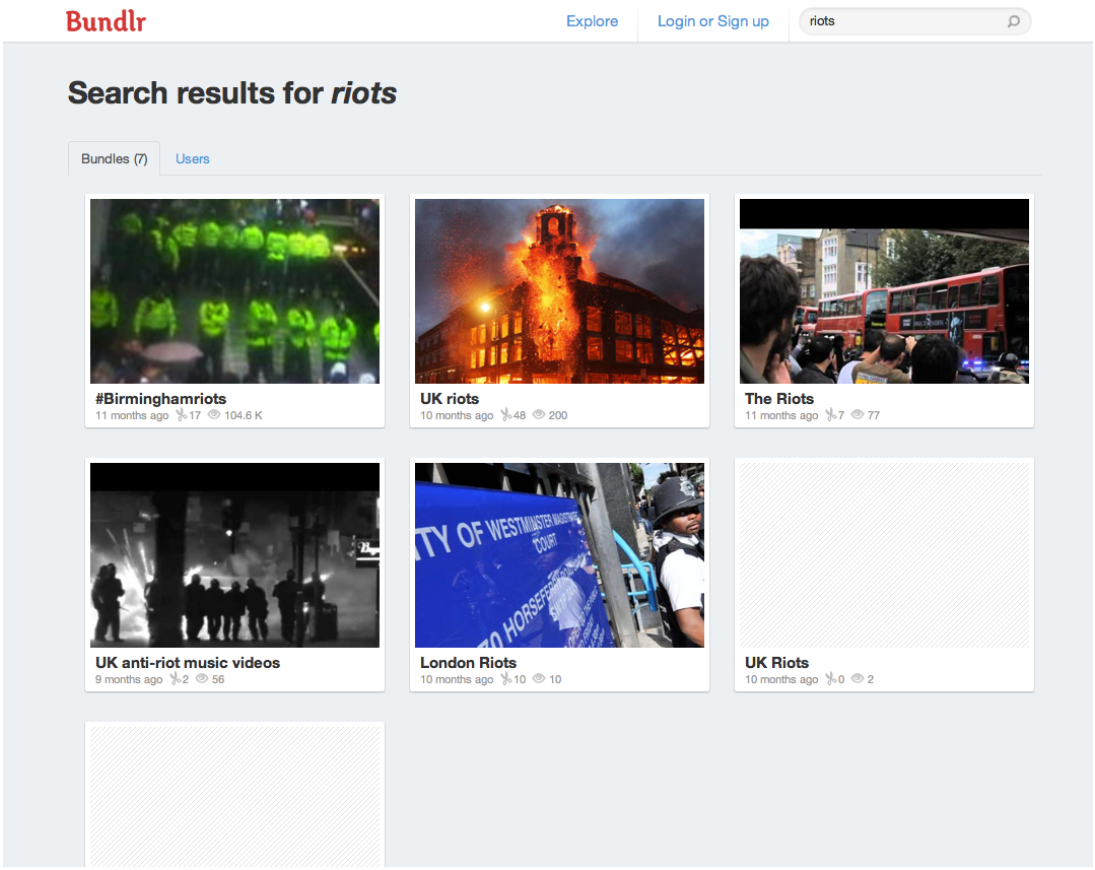

Figura 5.8: Pesquisa original para o termo *riots*. Note-se que apenas são encontrados 7 bundles. Recolhida em Maio de 2012.

Tipos de pesquisa pretendidos

Antes de mais, pretende-se que seja possível pesquisar utilizadores e conteúdo. Aqui define-se conteúdo como uma junção entre os bundles e os seus clips e não apenas bundles como na pesquisa original. Interessa também permitir ao utilizador pesquisar apenas no conteúdo dos seus bundles.

Apresentação dos resultados

Embora seja interessante ter clips no resultado das pesquisas, não se pode ignorar o facto de vários clips resultado poderem ser provenientes do mesmo

bundle. De facto, esse é um dos aspectos mais importantes e diferenciadores do Bundlr - o conteúdo é organizado em colecções temáticas. Assim, ao pesquisar "iPhone" temos todo o interesse em saber que os três primeiros clips resultado fazem todos parte de um bundle chamado "Apple".

Torna-se então importante agrupar os resultados (que são clips) pelos bundles a que pertencem.

A figura 5.9 apresenta o resultado desta agregação.

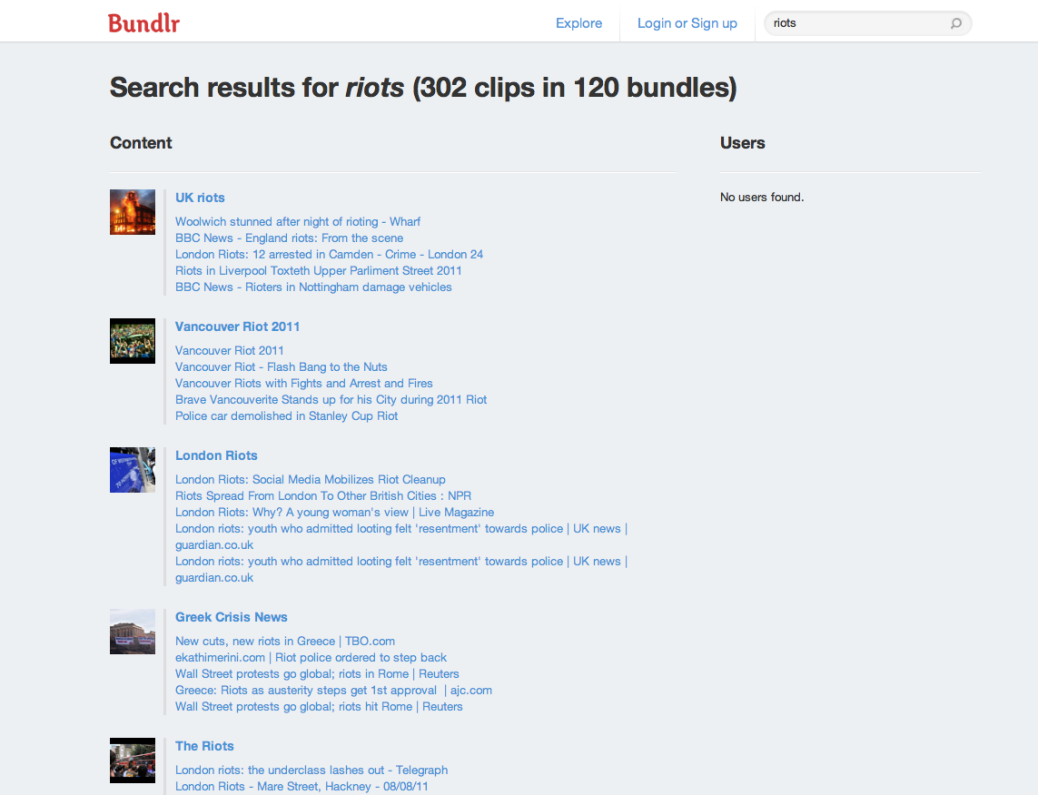

Figura 5.9: Pesquisa *Full-Text* para o termo *riots*. Os clips resultado encontram-se agrupados pelo bundle a que pertencem. O aspecto dos resultados não é o que será colocado em produção. Recolhida em Junho de 2012.

As *User Stories* resultantes desta análise de requisitos encontram-se no apêndice B.6, com as respectivas *Acceptance Stories*.

Requisitos não-funcionais

Durante a análise de requisitos foi feita uma selecção dos requisitos não-funcionais mais importantes, que será listada de seguida.

Note-se que apenas são apresentados os requisitos específicos das alterações implementadas neste módulo.

Desempenho

D.1. O desempenho da pesquisa implementada neste módulo deve ter um acréscimo de menos de 50% no tempo médio de resposta do serviço, que é 1.5 segundos.

5.6.4 Implementação

No estado da técnica (secção 5.6.2) chegou-se à conclusão que as tecnologias a utilizar neste módulo são: WebSolr (Apache Solr), Sunspot e Sunspot Mongoid. Nesta secção são abordados alguns problemas e decisões provenientes da utilização destas tecnologias.

Configurações do Apache Solr

O Apache Solr permite configurar o seu processo de indexação através de filtros de configuração em três momentos[30]:

Filtros de Caracteres Fazem o pré-processamento dos caracteres do texto a indexar e podem adicionar, alterar e remover caracteres.

Tokenizer Divide o texto em tokens (podem ser palavras). Um exemplo seria um *tokenizer* que separasse o texto por espaços.

Filtros de Tokens Depois de divididos, os tokens são passados por uma série de filtros que os adiciona, altera ou remove do grupo de tokens a indexar. Um exemplo seria um filtro de remoção de *stopwords*. Depois disso, o grupo de tokens resultantes é finalmente indexado.

Estes filtros podem também (e devem) ser aplicados durante a pesquisa, para que haja correspondência entre o que foi indexado e os termos procurados.

A tabela 5.15 apresenta e descreve os filtros utilizados no processamento do texto a indexar e dos termos de pesquisa.

Estes filtros influenciam a qualidade dos resultados, afectando directamente a capacidade de retorno (*recall*) do motor de pesquisa e de devolver resultados relevantes (*precision*). Filtros como o *StopFilterFactory* aumentam normalmente a precisão dos resultados por melhorarem a sua qualidade, enquanto que filtros como o *KStemFilterFactory* aumentam a capacidade de retorno, aumentando o número de resultados.

Tipo de filtro	Filtro	Descrição
Filtro de Caracteres	*HTMLStripCharFilterFactory*	Remove tags HTML.
Tokenizer	*StandardTokenizerFactory*	Separação por espaços e pontuação.
Filtro de Tokens	*StandardFilterFactory*	Remove pontuação dos acrónimos e o caracter "s" do final dos tokens.
	LowerCaseFilterFactory	Converte os tokens para minúsculas.
	StopFilterFactory	Remove *stopwords* definidas numa lista configurável.
	LengthFilterFactory	Exclui tokens de tamanho menor que 3 e maior que 512.
	ISOLatin1AccentFilterFactory	Converte caracteres acentuados em versões sem acento.
	KStemFilterFactory	*Stemming* de palavras específico para Inglês.
	RemoveDuplicatesTokenFilterFactory	Remove tokens duplicados que surjam com os filtros anteriores.

Tabela 5.15: Filtros utilizados para configurar o processamento dos termos para a indexação e pesquisa.

Os filtros utilizados aparentam representar um bom compromisso destes dois factores.

Atributos pesquisados e filtragem dos resultados

A selecção dos atributos a indexar é feita em Ruby on Rails através da definição em cada modelo que se pretende pesquisar do bloco *searchable*, proveniente do Sunspot. Os campos indexados no clip são o título, a descrição, o conteúdo (contém uma transcrição do conteúdo do link, tal como extraído pela solução de Readability apresentada no módulo anterior), a selecção (no caso do utilizador seleccionar algum texto ao fazer clip), o título do bundle do qual faz parte e o texto da nota (se esta existir). No caso do *User*, os campos indexados são o nome, o username e a descrição.

O Sunspot permite também reforçar o peso de um campo nos cálculos de relevância, através de uma propriedade *boost*. No clip, ambos os campos título e título do bundle foram reforçados, enquanto que no utilizador se fez um reforço do campo nome, já que estes são os campos mais importantes. No caso do título do bundle, pretende-se que resultados que pertençam a um bundle cujo título inclua os termos de pesquisa tenham mais relevância que os restantes.

Durante a pesquisa é também feita a filtragem de resultados privados e de conteúdo não seguro da seguinte forma:

Utilizador não autenticado Excluído conteúdo privado e não seguro.

Utilizador autenticado Excluído conteúdo privado que não seja do utilizador.

Agrupamento de resultados por bundles

Como foi referido anteriormente, a apresentação dos resultados passa pelo agrupar dos clips resultado nos bundles de que fazem parte. O Apache Solr suporta este tipo de agrupamento desde a versão 3.3, através da funcionalidade *Result Grouping*[31], que utiliza um dos campos indexados para agrupar os resultados. No nosso caso o campo utilizado é *bundle_id* de cada clip.

Esta funcionalidade é também suportada pela versão mais recente do Sunspot (v2.0) e pode ser configurada para alterar o número de resultados a manter de cada grupo (por defeito apenas é devolvido o melhor resultado de cada grupo) e a ordem dos resultados dentro de cada um.

Mais importante que a ordem dos clips é a ordem dos bundles que os agrupam. O Solr tem algumas limitações no controlo desta ordenação, apenas considerando atributos do resultado melhor classificado de cada bundle. Isto é limitativo porque inviabiliza a ordenação por uma média das classificações dos resultados de cada bundle. Ou seja, imaginando dois bundles em que o primeiro tinha apenas um resultado com boa classificação e 9 resultados com má classificação e o segundo tinha 10 resultados com classificações boas mas um pouco mais baixas que a melhor do primeiro bundle, o primeiro estaria no topo da lista de resultados. Este tipo de situações deteriora bastante a relevância dos resultados, pelo que se tentaram encontrar soluções.

Em 2010 foi criado um *bug report*[32] no website de suporte do Apache Solr para que fosse possível ordenar os grupos tendo em conta todos os seus resultados. Em Janeiro de 2012 foi proposto um *patch* para resolver a situação que, até à data de escrita deste documento, ainda não foi incluída em nenhuma versão estável do sistema. Uma solução para o problema seria instalar uma versão alterada do Apache Solr num servidor próprio que incluísse

este *patch*. No entanto, além dos problemas de manter um servidor que já foram referidos anteriormente, o código alterado não foi revisto ou aceite pelos programadores do Solr, pelo que seria arriscado colocá-lo em produção - principalmente pertencendo a um sistema que nenhum dos membros da equipa conhece bem internamente.

Outra solução seria não agrupar os resultados por bundles e apresentar simplesmente uma lista de clips. Embora esta solução fosse a mais fácil, perder-se-ia o valor acrescentado da relação conteúdo / bundle, que é um importante factor diferenciador do serviço.

Finalmente, propôs-se uma solução intermédia, baseada no princípio que os melhores resultados de uma pesquisa estarão nos primeiros 100 grupos. Este princípio assume que os melhores bundles para a pesquisa em questão têm pelo menos um clip com boa classificação, embora também possa seleccionar alguns bundles menos bons, como é o caso do primeiro bundle do exemplo dado anteriormente. A solução propõe então que, assim que o utilizador faz a pesquisa, sejam encontrados os melhores grupos através da ordenação suportada pelo Apache Solr e seja depois feita uma ordenação secundária (já depois de obtidos os resultados, na aplicação Ruby on Rails) por critérios que englobam a totalidade de resultados para cada bundle devolvido.

Se for tida em conta a paginação de resultados para o utilizador final, as consequências são ter de recuperar sempre os 100 melhores grupos durante as 10 primeiras páginas (se forem apresentados 10 resultados por página).

A ordenação dos resultados das primeiras 10 páginas é feita através da seguinte regra:

> *classificação do bundle = média da classificação dos seus 5 melhores clips resultado* $\times$ *número de clips resultado desse bundle*

Esta regra favorece bundles com muitos resultados bons. A partir da décima página, os resultados passam a ser só ordenados pelo critério original do Solr.

A solução escolhida foi a última, por apresentar um compromisso aceitável, dada a importância do agrupamento de resultados. No futuro, assim que o *patch* for incluído numa versão do Apache Solr, pode-se considerar adaptar a solução para utilizar as próprias funções do sistema, tornando a ordenação mais rápida.

Pesquisa no conteúdo do utilizador

Para além da pesquisa no conteúdo de todos os bundles públicos, foi também implementada uma procura limitada aos conteúdos do utilizador. O objectivo é facilitar o acesso a informação recolhida anteriormente pelo utilizador.

Esta funcionalidade é implementada limitando o espaço de procura aos clips cujo *bundle_id* faz parte dos IDs dos bundles do utilizador.

Actualização do Índice de procura

Já se discutiram anteriormente os dois momentos principais de um motor de *Full-Text Search*: a construção do índice e a pesquisa. Uma questão importante desta separação é a actualização do índice ao serem criados, alterados e removidos registos.

O Sunspot lida com este problema através de ligações aos *callbacks* dos modelos indexados. Cada vez que detecta uma mudança numa instância do modelo (seja a sua criação, alteração ou destruição), encaminha a correspondente actualização para o servidor Solr.

Seguindo o princípio da não-surpresa, é feita uma actualização do índice cada vez que qualquer atributo do modelo indexado seja alterado. Isto pode significar um grande fluxo de actualizações irrelevantes no índice. Por exemplo, cada vez que o utilizador se autentica no sistema é actualizado o campo da data do seu último acesso. Esta pequena alteração - irrelevante para os conteúdos indexados do utilizador - iria causar uma actualização no índice do Solr.

Para apenas corrigir o índice quando os campos indexados são alterados é necessário fazer algumas alterações. A biblioteca permite definir quais os atributos a ignorar através da opção *without_attribute_changes_of*, mas não indicar apenas aqueles que são relevantes. Para colmatar esta falta acrescentou-se a opção *only_attribute_changes_of* ao Sunspot, especificando-se assim apenas os atributos indexados. Durante a reacção a uma mudança no modelo verifica-se se algum dos atributos indicados foi alterado, através do método *changed* do Mongoid e, se tiverem sido, corrige-se o índice.

A operação de correcção do índice pode demorar algum tempo a executar, já que tem de ser feito um pedido HTTP ao servidor Solr. A solução é executar essas operações em *background*, com recurso ao Delayed Job, tal como indicado no suporte do Heroku[33].

O Apêndice E.7 apresenta as alterações feitas neste módulo a nível dos modelos e controladores da arquitectura MVC da aplicação.

5.6.5 Testes

Testes de Funcionais

Passaram **24** dos **24** cenários de teste derivados das *Acceptance Stories* descritas no Apêndice B.6.

Passaram **14** dos **14** testes unitários criados para este módulo. O *spec* final de testes unitários é listado no Apêndice L.

Testes de Desempenho

Nesta secção compara-se o desempenho da solução de pesquisa *Full-Text Search* implementada com a solução original, já com as alterações do Módulo 1. Pretende-se determinar se, para um utilizador sem sessão iniciada, existe uma acréscimo de menos de 50% no tempo médio do serviço (1.5 segundos).

A base de dados usada pelos testes seguintes é composta por 106618 clips (dos quais 25503 têm o campo *content*), 14895 bundles e 13967 utilizadores. O índice de procura gerado pelo Solr é composto por 120585 documentos, que são a totalidade dos clips e dos utilizadores.

O utilizador considerado durante os testes não tem sessão iniciada, já que o objectivo dos testes que se seguem não é analisar nenhuma particularidade da pesquisa para utilizadores com sessão iniciada.

Contrariamente ao que foi feito nos testes de desempenho do Módulo 1, desta vez o primeiro teste foi realizado no próprio servidor da aplicação. Este teste divide-se também em dois tipos: pedidos locais à acção de pesquisa e a execução directa da *query* em questão (sem qualquer tipo de computação de HTML nem pedido HTTP). Os testes foram executados em série, 300 vezes, com 40±20 milissegundos de separação. A pesquisa feita é constante entre as várias execuções, tendo 2539 clips resultado agregados em 1162 bundles no caso da *Full-Text Search* e 646 bundles resultado na pesquisa original.

A tabela 5.16 apresenta os resultados destes testes. Tanto nos pedidos à acção de pesquisa (pedidos HTTP) como na execução directa da *query* se nota um acréscimo de cerca de 200ms na pesquisa *Full-Text Search*. Uma comparação dos resultados da pesquisa original com os resultados obtidos nos testes do Módulo 1 mostra uma diminuição acentuada do tempo de execução da *query*. Este facto pode dever-se a optimizações feitas pela equipa do Bundlr fora do contexto do estágio.

	Pesquisa Original	***Full-Text Search***
Pedido HTTP	0.4460 ± 0.1531	0.6581 ± 0.1887
Execução da *Query*	0.0102 ± 0.0212	0.2244 ± 0.0645

Tabela 5.16: Desempenho da pesquisa local ao servidor tanto a nível de um pedido HTTP completo como a nível da *query*. Resultados em segundos.

Como se pretende avaliar o desempenho da funcionalidade de pesquisa como um todo, fez-se um segundo teste, desta vez a partir de um computador cliente situado em Portugal, para analisar as diferenças de desempenho de um

pedido remoto à acção de pesquisa. Neste teste foram feitas 50 pesquisas dos 12 termos mais pesquisados no Bundlr e de um termo com poucos resultados (13 termos no total). As pesquisas são feitas com ordem aleatória e são espaçadas por 40±20 milissegundos. Pretende-se obter o tempo médio das pesquisas mais populares no serviço para poder então avaliar as duas soluções. O termo com poucos resultados dá uma percepção da influência do número de resultados no tempo de pesquisa. A ferramenta utilizada para este teste foi o Apache JMeter.

	Pesquisa Original	***Full-Text Search***
Tempo médio	0.848 ± 0.4831	1.561 ± 0.5903

Tabela 5.17: Desempenho da pesquisa remota para o código original e para o implementado no módulo. Resultados em segundos.

A tabela 5.17 apresenta a média e desvio padrão dos resultados. Tal como aconteceu nos testes do Módulo 1, houve um aumento bastante significativo do desvio padrão neste teste.

Analisando os resultados individuais de cada termo de pesquisa, disponíveis nos Anexo externo O.2, percebe-se que independentemente da quantidade de resultados da pesquisa, há sempre o limite dos 100 bundles necessários à ordenação das primeiras 10 páginas.

Os valores conseguidos mostram que com 68.3% de segurança ($[-\sigma, \sigma]$) que a média real da nova pesquisa se encontra no intervalo [0.9707, 2.1513], estando abaixo dos 2.25 segundos objectivo do requisito D.1.

Capítulo 6

Conclusões

Este documento apresentou o trabalho desenvolvido e as decisões tomadas durante ambos os semestres de estágio. No seu decorrer, desenvolvi um conjunto de módulos para o serviço web Bundlr.

Usei uma metodologia de desenvolvimento de software baseada em Scrum, com sprints, *Product Backlogs*, *Burndown Charts* e estimativas de esforço diárias.

Escrevi software, criei e executei testes unitários e de regressão em todos os módulos. Além desses testes funcionais, avaliei também alguns atributos não funcionais, tais como segurança, desempenho e escalabilidade.

Fiz um conjunto de estudos e comparações de desempenho e obtive vários resultados para análise. Com base nesses estudos, pude concluir sobre as melhores soluções a aplicar em determinados algoritmos fundamentais de alguns dos módulos.

No final de cada módulo, o trabalho desenvolvido passou por uma fase de *Quality Assurance* com o resto da equipa, para ser feita a sua integração plena no site real.

Muito em breve, todas as funcionalidades desenvolvidas estarão em produção, e sinto que participei em pleno num processo de Engenharia de Software.

Referências

[1] R. Good, "The newsmaster toolkit: Content curation tools to aggregate, filter, edit, curate and distribute any type of content." http://www.mindmeister.com/maps/show/55395228, 2010. Acedido a 04-01-2012.

[2] E. Ries, "Minimum viable product: a guide." http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html, 2009. Acedido a 04-01-2012.

[3] D. H. Hansson, "Rails 0.5.0: The end of vaporware!." http://david.heinemeierhansson.com/arc/000254.html, 2004. Acedido a 06-01-2012.

[4] B. Venners, "Twitter on scala." http://www.artima.com/scalazine/articles/twitter_on_scala.html, 2009. Acedido a 06-01-2012.

[5] L. Carlson and L. Richardson, *Ruby Cookbook*. O'Reilly Media, first ed., 2006.

[6] Heroku, "Dynos." http://devcenter.heroku.com/articles/dynos, Dezembro 2011. Acedido a 07-01-2012.

[7] K. Banker, "The state of mongodb and ruby." http://blog.mongodb.org/post/2844804263/the-state-of-mongodb-and-ruby, 2011. Acedido a 07-01-2012.

[8] MongoHQ, "General questions - where is my mongohq database located?." http://docs.mongohq.com/faq-general#where-is-my-mongohq-database-located, 2011. Acedido a 10-01-2012.

[9] Heroku, "Can i connect to services outside of heroku?." http://devcenter.heroku.com/articles/external-services, 2011. Acedido a 10-01-2012.

[10] S. Souders, *High Performance Web Sites: Essential Knowledge for Front-End Engineers*. O'Reilly Media, first ed., 2007.

[11] J. Sutherland, R. van Solingen, and E. Rustenberg, *The Power of Scrum*. CreateSpace, first ed., 2011.

[12] N. Rappin, *Rails Test Prescriptions: Keeping Your Application Healthy*. Pragmatic Bookshelf, first ed., 2011.

[13] R. Goad, "Social networks now more popular than search engines in the uk." http://weblogs.hitwise.com/robin-goad/2010/06/social_networks_overtake_search_engines.html, 2010. Acedido a 04-01-2012.

[14] "Google trends: Rss." http://www.google.com/trends/?q=rss. Acedido a 09-01-2012.

[15] A. Toxboe, "Activity stream design pattern." http://ui-patterns.com/patterns/ActivityStream, 2010. Acedido a 09-01-2012.

[16] MongoDB, "Mongodb is fantastic for logging." http://blog.mongodb.org/post/172254834/mongodb-is-fantastic-for-logging, 2009. Acedido a 10-01-2012.

[17] A. S. W. Group, "Json activity streams 1.0." http://activitystrea.ms/specs/json/1.0/, 2011. Acedido a 09-01-2012.

[18] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, first ed., 1994.

[19] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, eds., *Recommender Systems Handbook*. Springer Berlin Heidelberg, first ed., 2010.

[20] R. Burke, "Hybrid web recommender systems," in *The Adaptive Web* (P. Brusilovsky, A. Kobsa, and W. Nejdl, eds.), pp. 377–408, Springer Berlin Heidelberg, 2007.

[21] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, pp. 76–80, 2003.

[22] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pp. 253–260, ACM, 2002.

[23] J. Golbeck, "Generating predictive movie recommendations from trust in social networks," in *iTrust*, vol. 3986 of *Lecture Notes in Computer Science*, pp. 93–104, Springer, 2006.

[24] Twitter, "What are hashtags ("#" symbols)?." http://support.twitter.com/articles/49309-what-are-hashtags-symbols. Acedido a 10-06-2012.

[25] J. Pomikálek, *Removing Boilerplate and Duplicate Content from Web Corpora*. PhD thesis, Masaryk University, Faculty of Informatics, 2011.

[26] MongoDB, "Full text search in mongo." http://www.mongodb.org/display/DOCS/Full+Text+Search+in+Mongo, 2011. Acedido a 18-05-2012.

[27] PostgreSQL, "Postgresql: Documentation: 8.3: Full text search." http://www.postgresql.org/docs/8.3/static/textsearch.html. Acedido a 21-05-2012.

[28] MySQL, "Full-text search functions." http://dev.mysql.com/doc/refman/5.6/en/fulltext-search.html. Acedido a 21-05-2012.

[29] Heroku, "Full text search options on heroku." https://devcenter.heroku.com/articles/full-text-search, 2012. Acedido a 21-05-2012.

[30] Solr, "Analyzers, tokenizers, and token filters." http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters/, 2012. Acedido a 23-05-2012.

[31] Solr, "Result grouping / field collapsing." http://wiki.apache.org/solr/FieldCollapsing, 2012. Acedido a 25-05-2012.

[32] "[#solr-2072] search grouping: expand group sort options." https://issues.apache.org/jira/browse/SOLR-2072, 2010. Acedido a 25-05-2012.

[33] Heroku, "Websolr - updating asynchronously with heroku workers." https://devcenter.heroku.com/articles/websolr#updating_asynchronously_with_heroku_workers, 2012. Acedido a 8-06-2012.

[34] G. Little, "Why evernote is winning with the soft stuff." http://www.forbes.com/sites/nicoleperlroth/2011/07/13/why-evernote-is-winning-with-the-soft-stuff/, 2011. Acedido a 29-12-2011.

[35] B. Johnson, "Oh, delicious - where did it all go so wrong?." http://gigaom.com/2011/09/28/oh-delicious-where-did-it-all-go-so-wrong/, 2011. Acedido a 29-12-2011.

[36] M. Cohn, *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional, first ed., 2004.

[37] L. Keogh, "Acceptance criteria vs. scenarios." http://lizkeogh.com/2011/06/20/acceptance-criteria-vs-scenarios/, 2011. Acedido a 05-11-2011.

[38] M. Pool, "The easy way to writing good user stories." http://www.codesqueeze.com/the-easy-way-to-writing-good-user-stories/, 2008. Acedido a 06-11-2011.

[39] E. Hovy and C.-Y. Lin, "Automated text summarization and the summarist system," in *Proceedings of a workshop on held at Baltimore, Maryland: October 13-15, 1998*, TIPSTER '98, pp. 197–214, Association for Computational Linguistics, 1998.

[40] S. Rose, D. Engel, N. Cramer, and W. Cowley, *Automatic Keyword Extraction from Individual Documents*, pp. 1–20. John Wiley & Sons, Ltd, 2010.

[41] Y. Matsuo and M. Ishizuka, "Keyword extraction from a single document using word co-occurrence statistical information," *International Journal on Artificial Intelligence Tools*, vol. 13, 2004.

[42] I. Mani, E. Bloedorn, and B. Gates, "Using cohesion and coherence models for text summarization," in *AAAI Symposium Technical Report SS-989-06*, pp. 69–76, AAAI Press, 1998.

[43] AlchemyAPI, "Concept tagging." http://www.alchemyapi.com/api/concept/. Acedido a 2-04-2012.

[44] C. Kohlschütter, P. Fankhauser, and W. Nejdl, "Boilerplate detection using shallow text features," in *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pp. 441–450, ACM, 2010.

[45] J. Pasternack and D. Roth, "Extracting article text from the web with maximum subsequence segmentation," in *Proceedings of the 18th international conference on World wide web*, WWW '09, pp. 971–980, ACM, 2009.

Apêndice A

Glossário

Acceptance Story Pequena frase que descreve um teste de aceitação de uma determinada *User Story*. Estes testes permitem ao cliente avaliar a implementação da funcionalidade descrita na *User Story* e ajudam a aumentar o seu nível de detalhe. A funcionalidade só é aceite se todos os testes passarem.

Activity Stream Listagem de actividades dos utilizadores que o utilizador subscreve, ordenada cronologicamente mantendo as mais recentes no topo. No contexto do Bundlr, esta listagem também inclui as actualizações relacionadas com os bundles subscritos pelo utilizador.

Artefactos Documentos desenvolvidos como suporte à prática de Scrum. Na metodologia original são identificados 3: *Backlog* de Sprint, *Backlog* de Produto e Gráfico *Burndown*. Ver Scrum.

Backlog de Produto Lista de todas as potenciais funcionalidades a serem desenvolvidas ao longo do projecto. Na metodologia Scrum original, pode ser editada a qualquer momento e é ordenada por importância.

Backlog de Sprint Lista do trabalho que a equipa tem de desenvolver durante o próximo sprint. É composta por alguns itens mais importantes do *Backlog* de Produto, tantos quanto a equipa achar que consegue implementar no sprint. Na metodologia original esta lista não pode ser editada durante o sprint, restrição que foi relaxada na metodologia implementada devido à necessidade de adicionar itens à lista depois da análise de requisitos, que é feita no seu decorrer.

Bookmarklet Código Javascript dentro de um favorito do *browser* que o executa sobre a página HTML que se encontra a ser visualizada assim que o utilizador o activar.

Bundle Colecção de conteúdo recolhido de várias fontes da Internet. No contexto deste estágio, bundles são colecções tal como implementadas pelo serviço Bundlr. Um bundle contém um conjunto de clips.

Caching Utilização de um sistema de acesso rápido que contém dados que se prevêem muito requisitados, de forma a evitar a sua recuperação de uma base de dados ou a sua geração através da aplicação em cada acesso.

Clip Unidade de conteúdo de um bundle. Representa o conteúdo em si e pode ser um link, uma selecção de texto, uma imagem, um vídeo, uma apresentação (*slideshow*), um artigo, uma mensagem de estado ou um documento.

Curadoria Processo de selecção do melhor conteúdo online sobre uma determinada área e a sua posterior divulgação.

DNS lookup Pedido feito por uma máquina que suporta o protocolo IP a um servidor DNS para obter o endereço IP de uma máquina com determinado nome de rede (*hostname*). Por exemplo, o *DNS lookup* do nome de rede *bundlr.com* tem como resultado o endereço 75.101.163.44.

Dyno É um processo isolado que executa num dos servidores do Heroku. Os *dynos*[6] recebem um pedido HTTP de cada vez e acedem a recursos partilhados como a base de dados.

Embed Secção de código ou aplicação que pode ser inserida num website e que irá carregar informação vinda de outra fonte ou website. Poderá ser uma aplicação em Adobe Flash ou código Javascript. A sua utilização mais popular é na incorporação de vídeos do YouTube. No contexto do MongoDB, o *embed* ou incorporação é uma relação entre dois documentos em que o incorporado é inserido dentro do que incorpora.

Feed Documento normalmente em XML que contém uma lista de itens que representam conteúdo, incluindo links para a sua origem. Os leitores de *feeds* agregam vários destes documentos subscritos pelo utilizador num único website para consumo. Os dois formatos mais conhecidos são o RSS e o Atom.

Frameworks de desenvolvimento web Conjunto de bibliotecas de código numa dada linguagem de programação que evitam o desenvolvimento de raiz de funcionalidades recorrentes na construção de aplicações web. Exemplos destas funcionalidades são o envio e recepção de pedidos HTTP, autenticação de utilizadores, ORM ou o processamento de URLs.

Gráfico Burndown Representação gráfica do trabalho que falta fazer relativamente ao tempo restante. Acompanha cada sprint do Scrum, sendo actualizado diariamente. É uma ferramenta que permite uma análise do progresso do sprint, facilitando a detecção de eventuais problemas.

Hostname Nome de rede de determinada máquina ou conjunto de máquinas, usado para a sua identificação. Exemplo: google.com.

Metodologia Ágil Conjunto de práticas de gestão de projecto baseadas no desenvolvimento iterativo e incremental de software, encorajando respostas rápidas e flexíveis à mudança.

Minimum Viable Product Versão do produto ou serviço que contém somente as funcionalidades que permitem que este seja colocado em produção. Este tipo de versões são normalmente lançadas para um conjunto pequeno de utilizadores que se espera que dê *feedback* sobre esta versão preliminar.

Model-View-Controller (MVC) Padrão de desenho de software (*Software Design Pattern*) que ajuda a isolar a lógica de negócio da apresentação do conteúdo, dividindo a aplicação em três componentes principais. Os *Models* abstraem o acesso à base de dados. As *Views* permitem definir a apresentação dos dados ao utilizador. Os *Controllers* fazem a ligação entre os pedidos e acções do utilizador, os respectivos procedimentos nos *Models* e a *View* indicada.

Modelo Freemium Modelo de negócio que disponibiliza um serviço de forma gratuita, cobrando por funcionalidades avançadas*premium*.

Object-Relational mapping (ORM) Biblioteca que abstrai o acesso a um conjunto de sistemas de gestão de base de dados relacionais da estrutura desses dados e das suas relações. Pode ser usada para possibilitar o suporte de várias tecnologias de base de dados de forma transparente, ou simplesmente para servir de interface de alto nível para um sistema de gestão de base de dados.

Object-Document mapping (ODM) Equivalentes aos ORM, mas específicos para sistemas de bases de dados não-relacionais orientados a documentos.

Observer Padrão de desenvolvimento de software (*Software Design Pattern*) que define uma dependência 1-para-N entre o sujeito e N observadores, de forma que quando o sujeito muda o seu estado, todos os

observadores são notificados e actualizados automaticamente. O Mongoid implementa este padrão internamente através de uma classe Mongoid::Observer, que foi utilizada no desenvolvimento do terceiro módulo deste estágio. Nesta implementação, todos os modelos assumem o papel de sujeito, podendo ter ou não uma classe que os observe.

Profissional de informação Jornalista ou qualquer outro profissional cuja actividade se foque na geração e transmissão de informação. Na Internet, o conceito estende-se para os *bloggers* profissionais.

Redes Sociais No contexto da Internet são serviços que reflectem as relações interpessoais online. Normalmente, os utilizadores mantêm páginas de perfil, subscrevem ao conteúdo de outros utilizadores e comunicam através do próprio serviço. As redes sociais mais populares da actualidade são o Facebook e o Twitter.

Scrum Metodologia ágil de desenvolvimento de software baseada em *sprints*, iterações de 2 a 4 semanas, cujos objectivos são definidos inicialmente numa reunião e não são alteráveis até ao final da iteração. A metodologia define também um conjunto de actividades que decorrem ao longo do *sprint* e uma série de artefactos, documentos que servem o propósito de acompanhar os progressos da equipa de desenvolvimento.

Scrum Master No Scrum original, o *Scrum Master* é o responsável por maximizar os benefícios da metodologia, por garantir que está a ser bem utilizada e por todo o processo Scrum em geral. Na metodologia implementada, o *Scrum Master* assume um papel de facilitador, esclarecendo dúvidas relativas à arquitectura do sistema que possam surgir no decorrer dos sprints.

Sprint Iterações da metodologia Scrum que tipicamente têm a duração de 2 a 4 semanas. Ver Scrum.

User Story Pequena frase que resume um cenário de utilização, descrevendo uma funcionalidade ou passo em específico do sistema. Existem vários formatos para a sua representação, dando-se especial importância à sua simplicidade, já que deve ser entendida por todas as entidades envolvidas no projecto. São principalmente utilizadas em metodologias ágeis, por exigirem pouca manutenção e serem ideais, pela sua simplicidade, para responder à mudança.

Apêndice B

User Stories implementadas

No presente apêndice são apresentadas as *User Stories* de cada um dos módulos realizados. Cada uma é acompanhada por um identificador, uma estimativa das horas que levará a implementar e pelas respectivas *Acceptance Stories*.

B.1 Bundles Privados

US1 - Criar bundles privados

> Como **utilizador que subscreve o Plano Profissional**, quero poder **criar bundles privados**, de forma a gerir o seu conteúdo de forma privada.

Horas Estimadas: 3.

Acceptance Stories:

Escolha na criação: Dado um utilizador com Plano Profissional, quando tenta criar um bundle novo no website, deverá conseguir escolher se o bundle é privado ou não.

Escolha na criação no *bookmarklet*: Dado um utilizador com Plano Profissional, quando tenta criar um bundle novo através do *bookmarklet*, deverá conseguir escolher se o bundle é privado ou não.

Criação sem Plano: Dado um utilizador sem Plano Profissional, quando tenta criar um bundle novo no website ou através do *bookmarklet*, não deverá ter possibilidade de escolher o grau de privacidade do bundle, devendo este ser automaticamente público.

US2 - Conhecer a visibilidade

Como **autor de bundles**, quero conseguir **saber qual a visibilidade** dos meus bundles, de forma a distinguir os públicos dos privados.

Horas Estimadas: 2.

Acceptance Stories:

Indicação de visibilidade na página de perfil: Dado um utilizador, quando acede à sua página de perfil, deverá conseguir distinguir um bundle privado de um bundle público.

Indicação de visibilidade na página do bundle: Dado um utilizador, quando acede à página de qualquer bundle, deverá conseguir perceber se o bundle é privado ou público.

US3 - Alterar a visibilidade

Como **utilizador que subscreve o Plano Profissional**, quero poder **alterar a visibilidade** dos meus bundles, de forma a conseguir gerir quais são públicos e quais são privados.

Horas Estimadas: 4.

Acceptance Stories:

Alterar visibilidade: Dado um utilizador com Plano Profissional, quando acede à página de um dos seus bundles, deverá poder activar um elemento gráfico da página (link ou botão) que altere a sua visibilidade de público para privado e vice-versa.

Impedir alteração de visibilidade: Dado um utilizador sem Plano Profissional, quando acede à página de um dos seus bundles (ou dos bundles de qualquer outro utilizador), não deverá poder activar qualquer elemento gráfico da página (link ou botão) que altere a visibilidade do bundle de público para privado e vice-versa.

Ocultar controlos de partilha e *embed* Dado um utilizador com permissões para aceder a determinado bundle privado, quando lhe acede, não deverá ter acesso às funcionalidades de partilha e *embedding* do conteúdo do bundle.

Ocultar função de marcar como favorito Dado um utilizador com permissões para aceder a determinado bundle privado, quando lhe acede, não deverá poder marcar o bundle como favorito.

Ocultar contadores Dado um utilizador com permissões para aceder a determinado bundle privado, quando lhe acede, não deverá ver os contadores do número de partilhas e favoritos.

US4 - Limitar a visibilidade a outrem

Como **autor de um bundle privado**, quero que **apenas seja permitida a visualização** e interacção com o meu bundle a mim e aos seus colaboradores, de forma a garantir a segurança do seu conteúdo.

Horas Estimadas: 10.

Acceptance Stories:

Visibilidade Limitada: Dado um utilizador que não seja autor nem colaborador de um determinado bundle privado, quando tenta aceder à página desse bundle, deverá ser impedido de ver o conteúdo do bundle, de perceber que o bundle existe e de editar esse conteúdo.

Visível para colaboradores: Dado um colaborador de um determinado bundle privado, quando tenta aceder à página desse bundle, deverá ter acesso ao seu conteúdo e ser capaz de o editar.

Ocultar da pesquisa logged out: Dado um utilizador sem sessão iniciada, quando faz uma pesquisa por um bundle, apenas bundles públicos podem fazer parte dos resultados.

Adaptação da pesquisa: Dado um utilizador com sessão iniciada, quando faz uma pesquisa por um bundle, deverá poder encontrar, além dos bundles públicos, todos aqueles que tem permissão para ver (os seus e todos aqueles em que colabora), independentemente de serem públicos ou privados.

Protecção do RSS: Dado um utilizador que não seja autor nem colaborador de um determinado bundle privado, quando tenta aceder à sua *feed RSS*, deverá ser impedido de ver a listagem do conteúdo do bundle.

Protecção do *Embed*: Dado um qualquer utilizador, quando tenta visualizar o *embed* de um bundle que passou a ser privado, deverá ver uma mensagem de erro a explicar que o bundle já não é público em vez da listagem do conteúdo do bundle.

US5 - Privado não pago

Como **administrador**, quero que um participante de um bundle privado, cujo autor **já não subscreva** o Plano Profissional, seja **impedido de ver e alterar** o seu conteúdo, de forma a restringir as funcionalidades *premium* a pagantes.

Horas Estimadas: 12.

Acceptance Stories:

Impedir acesso ao autor: Dado um utilizador que já não subscreva o Plano Profissional, quando tenta aceder a um dos seus bundles privados, deverá ver uma mensagem de erro que lhe explique que tem de subscrever ao plano para ter acesso a bundles privados e ser impedido de ver ou alterar o conteúdo.

Impedir acesso ao colaborador: Dado um utilizador que colabore num bundle privado cujo autor já não subscreva o Plano Profissional, quando tenta aceder a esse bundle, deverá ver uma mensagem de erro que lhe explique que o autor já não subscreve ao plano e que enquanto assim for não é possível ver ou alterar o seu conteúdo e ser impedido de fazer essas mesmas acções.

Impedir adição de conteúdo pelo autor: Dado um utilizador que já não subscreva o Plano Profissional, quando tenta adicionar conteúdo através do *bookmarklet* a um dos seus bundles privados, deverá receber uma mensagem de erro que explique que tem de subscrever ao plano para poder alterar bundles privados e ser impedido de o fazer.

Impedir adição de conteúdo pelo colaborador: Dado um utilizador que colabore num bundle privado cujo autor já não subscreva o Plano Profissional, quando tenta adicionar conteúdo através do *bookmarklet* a esse bundle, deverá receber uma mensagem de erro que explique que o autor já não subscreve ao plano e que enquanto assim for não é possível ver ou alterar o seu conteúdo e ser impedido de fazer essas mesmas acções.

US6 - Publicação de um bundle não pago

> Como **administrador**, quero que o **autor de um bundle privado** que **já não subscreva** o Plano Profissional, o possa **tornar público**, de forma a possibilitar a recuperação do conteúdo a quem não renovar a subscrição.

Horas Estimadas: 3.

Acceptance Stories:

Opções em caso de bloqueio de bundle privado: Dado um utilizador que já não subscreva o Plano Profissional, quando tenta aceder à página de um dos seus bundles privados e vir o seu conteúdo bloqueado, deverá ter a opção de tornar o bundle público ou de subscrever o Plano Profissional.

US7 - Sensibilizar para subscrição

> Como **administrador**, quero que um utilizador que **ainda não subscreva** o Plano Profissional, veja uma **mensagem de sensibilização para a subscrição do plano** ao tentar tornar alterar a visibilidade de um bundle para privado, de forma a maximizar o número de subscritores do plano.

Horas Estimadas: 3.

Acceptance Stories:

Mensagem de sensibilização: Dado um utilizador que não subscreva o Plano Profissional, quando tenta alterar a visibilidade de um bundle de público para privado através do elemento gráfico na página do bundle, deverá ver uma mensagem que o sensibilize para subscrever o Plano Profissional.

B.2 Limitação da Colaboração

US8 - Limitar Colaboração

> Como **administrador**, quero que apenas os utilizadores **que subscrevam** um plano pago **possam adicionar colaboradores** aos seus bundles, de forma a restringir as funcionalidades *premium* a subscritores de planos pagos.

Horas Estimadas: 4.

Acceptance Stories:

Possibilitar Colaboração: Dado um utilizador que subscreva um dos planos pagos, quando acede à página de dos seus bundles, deverá ver e poder activar um elemento gráfico da página (link ou botão) que permita a adição de colaboradores ao bundle.

Impedir Colaboração: Dado um utilizador que não subscreva um dos planos pagos, quando acede à página de dos seus bundles, não poderá activar um elemento gráfico da página (link ou botão) que possibilite a adição de colaboradores ao bundle.

US9 - Colaborativo não pago

> Como **administrador**, quero que os **colaboradores** de um bundle, cujo autor **já não subscreva** um dos planos pagos, **sejam impedidos** de o alterar, de forma a restringir as funcionalidades *premium* a subscritores dos planos pagos.

Horas Estimadas: 4.

Acceptance Stories:

Alertar o autor: Dado um utilizador que já não subscreva um dos planos pagos, quando tenta aceder a um dos seus bundles colaborativos do qual é autor, deverá ver uma mensagem de aviso que lhe explique que tem de subscrever ao plano para poder adicionar e gerir colaboradores.

Alertar o colaborador ao aceder: Dado um utilizador que colabore num bundle cujo autor já não subscreva um dos planos pagos, quando tenta aceder a esse bundle, deverá ver uma mensagem de aviso que lhe explique que o autor já não subscreve ao plano e que enquanto assim for os colaboradores são impedidos de alterar o seu conteúdo e deverá ser impedido de fazer qualquer edição.

Impedir adição de conteúdo pelo colaborador: Dado um utilizador que colabore num bundle cujo autor já não subscreva um dos planos pagos, quando tenta adicionar conteúdo através do *bookmarklet* a esse bundle, deverá receber uma mensagem de erro que explique que o autor já não subscreve ao plano e que enquanto assim for os colaboradores são impedidos de alterar o conteúdo do bundle e deverá ser impedido de fazer qualquer adição ou edição.

US10 - Sensibilizar para subscrição

Como **administrador**, quero que um utilizador que **ainda não subscreva** um dos planos pagos, veja uma **mensagem de sensibilização para a subscrição do Plano de Equipa** ao tentar adicionar ou gerir os colaboradores de um bundle, de forma a maximizar o número de subscritores do plano.

Horas Estimadas: 3.

Acceptance Stories:

Mensagem de sensibilização: Dado um utilizador que não subscreva um dos planos pagos, quando tenta adicionar ou gerir os colaboradores de um bundle através do elemento gráfico na página do bundle, deverá ver uma mensagem que o sensibilize para subscrever o Plano de Equipa.

B.3 Subscrição de Bundles e de Utilizadores

US11 - Seguir utilizador

> Como **utilizador autenticado**, quero poder **subscrever as actividades de qualquer utilizador**.

Horas Estimadas: 8.

Acceptance Stories:

Apresentar elemento gráfico adequado: Dado um utilizador autenticado, quando acede à página de outro utilizador, deve conseguir ver um elemento gráfico que possibilite a subscrição.

Subscrever actividades de um utilizador: Dado um utilizador autenticado, quando acede à página de outro utilizador e tenta começar a seguir as suas actividades através de um elemento gráfico para esse efeito, deve ver uma mensagem que confirme a subscrição.

Impedir a utilizadores não autenticados: Dado um utilizador não autenticado, quando acede à página de um qualquer utilizador e tenta activar o elemento gráfico que possibilita a subscrição, deve ser impedido de subscrever e ser redireccionado para a página de início de sessão.

US12 - Seguir bundle

> Como **utilizador autenticado**, quero poder **subscrever as actividades de qualquer bundle** a que tenha acesso.

Horas Estimadas: 10.

Acceptance Stories:

Apresentar elemento gráfico adequado: Dado um utilizador autenticado, quando acede à página de um bundle a que tenha acesso, deve conseguir ver um elemento gráfico que possibilite a subscrição.

Subscrever actividades de um bundle: Dado um utilizador autenticado, quando acede à página de um bundle a que tenha acesso e tenta começar a seguir as suas actividades através de um elemento gráfico para esse efeito, deve ver uma mensagem que confirme a subscrição.

Impedir a utilizadores não autenticados: Dado um utilizador não autenticado, quando acede à página de um qualquer bundle público e tenta activar o elemento gráfico que possibilita a subscrição, deve ser impedido de subscrever e ser redireccionado para a página de início de sessão.

Seguir os meus bundles: Dado um utilizador autenticado, quando cria um bundle, passa a segui-lo automaticamente, mesmo não estando na lista de seguidores do bundle.

Seguir os bundles em que colaboro: Dado um utilizador autenticado, quando aceita um convite de colaboração para um bundle, passa a segui-lo automaticamente.

US13 - Parar de seguir utilizador

Como **utilizador autenticado**, quero poder **deixar de subscrever as actividades** de um utilizador que siga.

Horas Estimadas: 3.

Acceptance Stories:

Apresentar elemento gráfico adequado: Dado um utilizador autenticado, quando acede à página de um utilizador que siga, deve conseguir ver um elemento gráfico que possibilite cancelar a subscrição das suas actividades.

Parar de subscrever actividades de um utilizador: Dado um utilizador autenticado, quando acede à página de um utilizador que siga e tenta deixar de subscrever as suas actividades através de um elemento gráfico para esse efeito, deve ver uma mensagem que confirme o cancelamento da subscrição.

US14 - Parar de seguir bundle

Como **utilizador autenticado**, quero poder **deixar de subscrever as actividades** de um bundle que siga.

Horas Estimadas: 4.

Acceptance Stories:

Apresentar elemento gráfico adequado: Dado um utilizador autenticado, quando acede à página de um bundle que siga, deve conseguir ver um elemento gráfico que possibilite cancelar a subscrição das suas actividades.

Parar de subscrever actividades de um bundle: Dado um utilizador autenticado, quando acede à página de um bundle que siga e tenta deixar de subscrever as suas actividades através de um elemento gráfico para esse efeito, deve ver uma mensagem que confirme o cancelamento da subscrição.

Não poder parar de seguir os meus bundles: Dado um utilizador autenticado, quando acede a um dos seus bundles, não deve ver o elemento gráfico que permite parar de seguir o bundle nem ser capaz de activar o seu efeito.

US15 - Saber quem segue o quê

Como **utilizador**, quero **saber quem segue que bundles e utilizadores**.

Horas Estimadas: 14.

Acceptance Stories:

Seguidores de um utilizador: Dado um utilizador qualquer, quando acede à página de outro qualquer utilizador, deve conseguir ter acesso à listagem de utilizadores que seguem esse utilizador.

Seguidores de um bundle: Dado um utilizador qualquer, quando acede à página de um qualquer bundle público, deve conseguir ter acesso à listagem de utilizadores que seguem esse bundle.

Seguidos por um utilizador: Dado um utilizador qualquer, quando acede à página de outro qualquer utilizador, deve conseguir ter acesso à listagem de utilizadores e bundles que esse utilizador segue.

Os meus seguidores: Dado um utilizador autenticado, quando acede à sua página de perfil, deve conseguir ter acesso à listagem de utilizadores que o seguem.

Seguidos por mim: Dado um utilizador autenticado, quando acede à sua página de perfil, deve conseguir ter acesso à listagem de utilizadores e bundles que segue.

Ocultar privados: Dado um utilizador qualquer, quando acede à página de outro qualquer utilizador, não deve conseguir ter acesso aos bundles privados que esse utilizador segue.

US16 - Registo de actividades

Como **administrador**, quero que sejam **registadas actividades** assim que as seguintes acções sejam realizadas: novo clip, novo bundle, novo colaborador, bundle seleccionado para *homepage*, seguir utilizador, seguir bundle, subscrever um plano pago.

Horas Estimadas: 8.

Acceptance Stories:

Novo Clip: Dado um utilizador autenticado, quando adiciona um novo clip a um dos bundles que pode editar, deve ser criada uma actividade específica que o indique e à qual os seguidores do bundle tenham acesso.

Novo Bundle: Dado um utilizador autenticado, quando cria um novo bundle público, deve ser criada uma actividade específica que o indique e à qual os seus seguidores tenham acesso.

Novo Colaborador: Dado um utilizador autenticado, quando adiciona um novo colaborador a um dos seus bundles e este aceita o convite, deve ser criada uma actividade específica que o indique e à qual os seguidores do bundle e do novo colaborador tenham acesso.

***Staff-pick*:** Dado um administrador, quando selecciona um bundle público para a *homepage* (*staff-pick*), deve ser criada uma actividade específica que o indique e à qual os seguidores do bundle tenham acesso.

Seguir Utilizador: Dado um utilizador autenticado, quando subscreve as actividades de outro utilizador, deve ser criada uma actividade específica que o indique e à qual os seus seguidores tenham acesso.

Seguir Utilizador: Dado um utilizador autenticado, quando subscreve as actividades de um bundle a que tenha acesso, deve ser criada uma actividade específica que o indique e à qual os seus seguidores tenham acesso.

Subscrever o Plano Profissional: Dado um utilizador autenticado, quando subscreve o Plano Profissional, deve ser criada uma actividade específica que o indique e à qual os seus seguidores tenham acesso.

Subscrever o Plano Equipa: Dado um utilizador autenticado, quando subscreve o Plano Equipa, deve ser criada uma actividade específica que o indique e à qual os seus seguidores tenham acesso.

US17 - Consumir as actividades

Como **utilizador**, quero poder **ler e consumir** as actividades dos utilizadores e bundles que sigo, de forma a ter acesso ao conteúdo que mais me interessa.

Horas Estimadas: 16.

Acceptance Stories:

Página de Activity Stream: Dado um utilizador autenticado, quando acede sua página de perfil, deve conseguir aceder à página da sua *activity stream*.

Exibição das actividades: Dado um utilizador autenticado, quando acede à página da sua *activity stream*, deve ver uma listagem das actividades dos utilizadores e bundles que segue, ordenada cronologicamente do mais recente para o mais antigo e devidamente paginada.

Distinção de actividades lidas: Dado um utilizador autenticado, quando acede à página da sua *activity stream*, deve conseguir distinguir visualmente as actividades novas das já lidas.

Clips feitos por mim: Dado um utilizador autenticado, quando acede à página da sua *activity stream*, não deve ver qualquer actividade referente a um clip que tenha sido feito por ele próprio.

Remoção de actividades de clips: Dado um autor de um bundle, quando remove um dos clips do bundle, devem ser também removidas as actividades que lhe façam referência.

US18 - Proteger bundles privados

Como **autor de um bundle privado**, quero que só os **utilizadores que tenham acesso ao bundle** possam **segui-lo**, de forma a proteger os meus conteúdos.

Horas Estimadas: 7.

Acceptance Stories:

Remover acesso a ex-colaboradores: Dado um colaborador de um bundle privado, quando deixa de ser colaborador desse bundle, deve parar automaticamente de seguir esse bundle, deixando de ter acesso às suas actividades.

Alertar para a remoção de seguidores: Dado um utilizador com Plano Profissional, quando altera a visibilidade de um dos seus bundles de público para privado, deve ser avisado que o bundle perderá todos os seus seguidores, excepto os seus colaboradores.

Remover seguidores ao passar para privado: Dado um bundle público, quando a sua visibilidade é alterada para privado, devem ser removidos todos os seus seguidores, excepto colaboradores e o próprio autor.

B.4 Categorização e Exploração de bundles

US19 - Atribuir categoria ao criar

> Como **utilizador autenticado**, quero poder **atribuir uma categoria a um bundle público** durante a sua criação, de forma a colocá-lo na página dessa categoria.

Horas Estimadas: 10.

Acceptance Stories:

Escolha na criação: Dado um utilizador autenticado, quando tenta criar um novo bundle no website, deverá conseguir escolher uma das seguintes categorias: Arts & Design, Business, Education & Science, Entertainment, Lifestyle, News, Technology e Places.

Escolha na criação pelo *bookmarklet*: Dado um utilizador autenticado, quando tenta criar um bundle novo através do *bookmarklet*, deverá conseguir escolher uma das categorias definidas acima.

US20 - Editar a categoria

> Como **autor de bundles**, quero poder **alterar a categoria dos meus bundles**, de forma a conseguir colocá-lo na página dessa categoria.

Horas Estimadas: 4.

Acceptance Stories:

Editar a categoria: Dado um utilizador autenticado, quando acede à página de um dos seus bundles, deve conseguir editar a categoria do bundle através de um elemento gráfico da página.

US21 - Explorar por categorias

> Como **utilizador**, quero poder **explorar o conteúdo da plataforma por categorias**, de forma a encontrar conteúdo relevante.

Horas Estimadas: 12.

Acceptance Stories:

Ver categorias na página de exploração: Dado um utilizador qualquer, quando acede à página de exploração, deve conseguir ver os bundles públicos de cada categoria.

Ver bundles categorizados na página da categoria: Dado um utilizador qualquer, quando acede à página de uma categoria específica, deve conseguir ver todos os bundles públicos dessa categoria.

US22 - Ver a categoria de um bundle

Como **utilizador**, quero conseguir **saber qual a categoria de um bundle** se a tiver, de forma a perceber melhor temática do bundle.

Horas Estimadas: 4.

Acceptance Stories:

Categoria na página do bundle: Dado um utilizador qualquer, quando acede à página de um bundle público, deve conseguir ver qual a sua categoria.

Link para categoria na página do bundle: Dado um utilizador qualquer, quando acede à página de um bundle público, deve conseguir ir directamente para a página da categoria do bundle.

US23 - Bundles populares por categoria

Como **utilizador**, quero poder **ver uma selecção dos bundles mais populares** actualmente para **cada categoria**, de forma a saber o que é mais relevante neste momento em cada uma delas.

Horas Estimadas: 24.

Acceptance Stories:

Bundles populares na página de exploração: Dado um utilizador qualquer, quando acede à página de exploração, deve conseguir ver uma selecção dos 4 bundles mais populares actualmente em cada categoria.

US24 - Notificar quando popular

> Como **administrador**, quero que o autor de um bundle **seja notificado por email** quando este **se torna popular**, de forma a dar-lhe os parabéns pelo seu trabalho.

Horas Estimadas: 16.

Acceptance Stories:

> **Enviar email:** Dado um utilizador, quando um dos seus bundles é seleccionado como popular numa das categorias, deve receber um email do Bundlr a informá-lo e a dar-lhe os parabéns pelo seu trabalho.

US25 - Não apresentar bundles privados

> Como **administrador**, quero que **bundles privados não sejam apresentados na página de exploração**, de forma a garantir a segurança do seu conteúdo.

Horas Estimadas: 5.

Acceptance Stories:

> **Ocultar da página de exploração:** Dado um utilizador, quando acede à página de exploração, não deve ver nenhum bundle privado.
>
> **Ocultar da página de categoria:** Dado um utilizador, quando acede à página de uma categoria, não deve ver nenhum bundle privado.
>
> **Não devem ser populares:** Dado um bundle privado, não deve poder ser seleccionado como popular na sua categoria.

US26 - Destacar utilizadores com planos pagos

> Como **administrador**, quero que **sejam destacados utilizadores com planos pagos** na página de exploração, de forma a dar-lhes uma maior audiência.

Horas Estimadas: 4.

Acceptance Stories:

> **Utilizadores destacados na página de exploração:** Dado um utilizador, quando acede à página de exploração, deve ver utilizadores com planos pagos na secção utilizadores destacados.

US27 - Categorizador

> Como **administrador**, quero **poder atribuir categorias a bundles** antes do lançamento da página de exploração, de forma a não existirem categorias vazias inicialmente.

Horas Estimadas: 18.

Acceptance Stories:

Categorizador: Dado um administrador, quando acede à página do categorizador, deve ver um bundle sem categoria escolhido aleatoriamente e poder-lhe atribuir uma das 8 categorias.

Próximo bundle: Dado um administrador, quando acede à página do categorizador, deve ter a opção de passar o bundle actual à frente, para ser categorizado mais tarde.

B.5 Sugestão de conteúdo

US28 - Sugestões de bundles

> Como **utilizador autenticado**, quero ter acesso a **sugestões de bundles personalizadas** de acordo com o meu historial de clipping, de forma a encontrar subscrever conteúdo relevante.

Horas Estimadas: 12.

Acceptance Stories:

Página de sugestões: Dado um utilizador autenticado, quando acede à página de sugestões da *activity stream*, deve ver sugestões de bundles.

Utilizadores sem clips: Dado um utilizador autenticado que não criou clips suficientes para ser possível ter sugestões, quando acede à página de sugestões da *activity stream*, deve ver uma selecção dos bundles mais activos e vistos no Bundlr.

US29 - Sugestões de utilizadores

> Como **utilizador autenticado**, quero ter acesso a **sugestões de utilizadores personalizadas** de acordo com o meu historial de clipping, de forma a encontrar subscrever conteúdo relevante.

Horas Estimadas: 12.

Acceptance Stories:

Página de sugestões: Dado um utilizador autenticado, quando acede à página de sugestões da *activity stream*, deve ver sugestões de utilizadores.

Utilizadores sem clips: Dado um utilizador autenticado que não criou clips suficientes para ser possível ter sugestões, quando acede à página de sugestões da *activity stream*, deve ver uma selecção dos utilizadores mais activos no Bundlr.

B.6 *Full-Text Search*

US30 - Procura de conteúdo

> Como **utilizador**, quero que a **pesquisa seja feita no conteúdo dos clips** e não apenas no título e descrição do bundle, de forma a poder encontrar exactamente o que procuro.

Horas Estimadas: 20.

Acceptance Stories:

Pesquisa pelo conteúdo: Dado um utilizador qualquer, quando faz uma pesquisa no website, deve ver os clips que contêm os termos da pesquisa agrupados pelos bundles de que fazem parte.

Incluir as notas: Dado um utilizador qualquer, quando faz uma pesquisa no website por uma palavra presente na nota de um clip, deve ver esse clip nos resultados.

US31 - Procura no meu conteúdo

> Como **utilizador autenticado**, quero poder **pesquisar apenas no conteúdo dos meus bundles**, de forma a aceder rapidamente a informação que guardei.

Horas Estimadas: 5.

Acceptance Stories:

Pesquisa do conteúdo do utilizador: Dado um utilizador autenticado, quando faz uma pesquisa na secção do conteúdo do utilizador, deve ver apenas resultados de bundles dos quais é autor ou colaborador.

Pesquisa genérica: Dado um utilizador autenticado, quando faz uma pesquisa genérica (sem ser específica ao seu conteúdo), deve ver resultados de conteúdo seu e de outrem.

Limitar a utilizadores autenticados: Dado um utilizador não autenticado, quando tenta fazer uma pesquisa na secção do conteúdo do utilizador, deve ver os resultados de uma pesquisa genérica.

US32 - Pesquisa de utilizadores

Como **utilizador**, quero poder **pesquisar utilizadores** com base no seu nome, *username* e descrição, de forma a encontrar o utilizador que procuro.

Horas Estimadas: 5.

Acceptance Stories:

Pesquisa de utilizadores: Dado um utilizador qualquer, quando faz uma pesquisa no website, deve ver não só os resultados da pesquisa de conteúdo como os utilizadores resultantes da pesquisa de utilizadores.

US33 - Esconder conteúdos não seguros

Como **administrador**, quero que os resultados para utilizadores não autenticados **excluam conteúdo previamente marcado como não seguro**, de forma a proporcionar uma boa experiência de utilização.

Horas Estimadas: 2.

Acceptance Stories:

Ocultar conteúdos não seguros: Dado um utilizador não autenticado, quando faz uma pesquisa no website, deve ver apenas conteúdo que não foi marcado previamente pela equipa como não seguro.

Incluir conteúdos não seguros: Dado um utilizador autenticado, quando faz uma pesquisa no website, deve ver todo o conteúdo público (independentemente se foi marcado como inseguro ou não).

US34 - Esconder conteúdos privados

Como **administrador**, quero que os resultados **apenas incluam conteúdos a que o utilizador esteja autorizado a aceder**, de forma a proteger o conteúdo de bundles privados.

Horas Estimadas: 2.

Acceptance Stories:

Ocultar conteúdos privados: Dado um utilizador não autenticado, quando faz uma pesquisa no website, deve ver apenas conteúdo público.

Visivel para o autor: Dado um utilizador autenticado autor de um bundle privado, quando faz uma pesquisa no website em que esse bundle seja um dos resultados, deve poder ver o bundle na página de resultados.

Visivel para colaboradores: Dado um utilizador autenticado colaborador de um bundle privado, quando faz uma pesquisa no website em que esse bundle seja um dos resultados, deve poder ver o bundle na página de resultados.

Apêndice C

Gráficos *Burndown*

Este apêndice apresenta os Gráficos *Burndown* referentes a cada sprint deste estágio.

C.1 Outubro - Sprint 1

Neste sprint foram feitos os 2 primeiros módulos - Bundles Privados (secção 5.1) e Limitar Colaboração (secção 5.2).

O início do sprint foi lento, muito porque o estagiário ainda estava a definir algumas partes da metodologia e a adaptar-se ao trabalho.

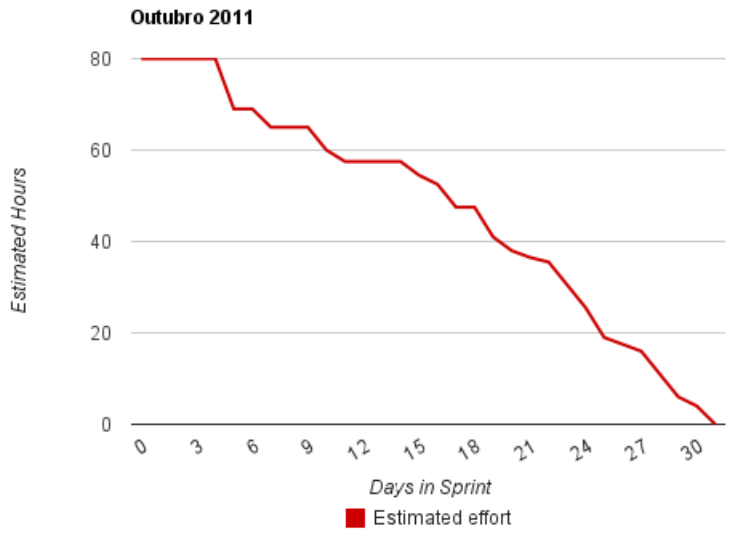

Figura C.1: Gráfico *Burndown* do primeiro sprint, em Outubro de 2011.

C.2 Novembro - Sprint 2

Neste sprint foi feita a análise e implementação da arquitectura do módulo 3 - Subscrição de bundles e utilizadores (secção 5.3).

O início deste sprint foi marcado pela configuração de métodos de teste automatizados através da biblioteca Cucumber e criação dos respectivos testes.

O trabalho foi interrompido do dia 9 até ao dia 12 já que a equipa se deslocou a Lisboa para participar no Codebits, uma conferência de programação.

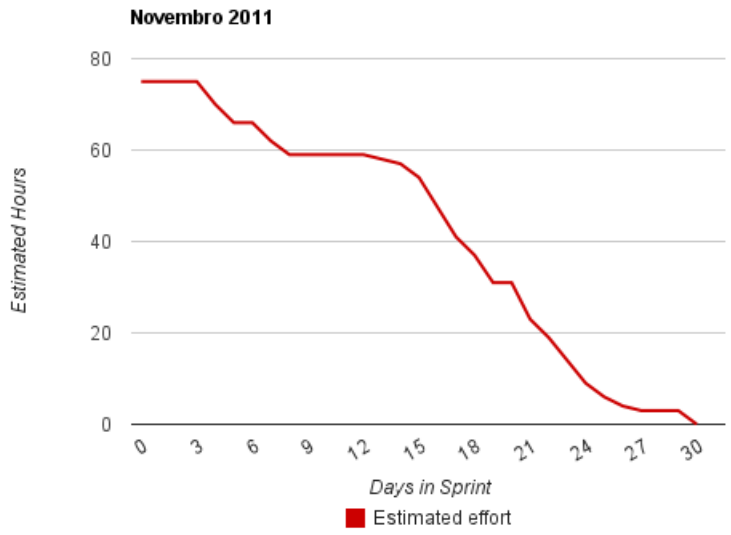

Figura C.2: Gráfico *Burndown* do segundo sprint, em Novembro de 2011.

C.3 Dezembro - Sprint 3

Neste sprint foi concluído o módulo 3, através da implementação das suas *User Stories*.

O decorrer deste sprint foi intercalado com entregas de trabalhos de disciplinas curriculares do estagiário e pelo Natal.

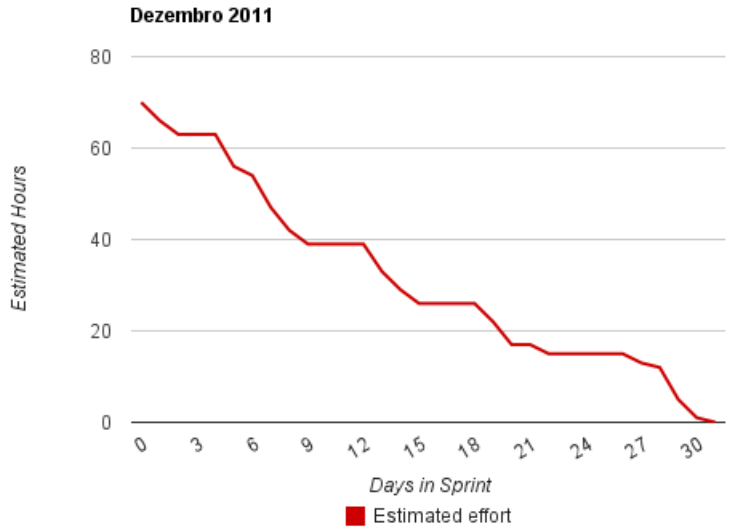

Figura C.3: Gráfico *Burndown* do terceiro sprint, em Dezembro de 2011.

C.4 Fevereiro - Sprint 4

Neste sprint foi implementado o módulo 4 - Categorização e exploração de bundles (secção 5.4).

A Defesa do Relatório Intermédio, inicialmente planeada para os primeiros três dias do mês foi adiada para dia 16, o que acabou por atrasar todo o sprint. O atraso de 6 dias não voltou a ser compensado, acabando por atrasar alguns dias todo o planeamento.

Note-se que este gráfico *burndown*, tal como os restantes, não tem início no primeiro dia do mês, portanto o eixo *Days in Sprint* não é equivalente aos dias do mês.

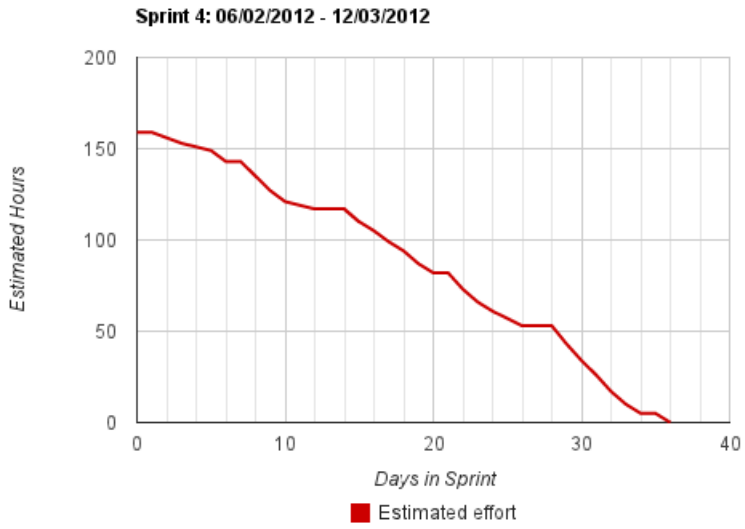

Figura C.4: Gráfico *Burndown* do quarto sprint, em Fevereiro de 2012.

C.5 Março/Abril - Sprint 5

Neste sprint foi iniciada a análise do módulo 5 - Sugestão de conteúdo (secção 5.5).

De notar neste sprint o início lento, que se deve a uma dificuldade inicial em encontrar bibliografia apropriada para o tema da sugestão de conteúdo.

Note-se também que houveram tarefas (análises de *Tag Retrieval*, *Tag Extraction* e métodos de comparação de grupos de tags) que não foram concluídas a tempo do final do sprint, passando para o sprint seguinte.

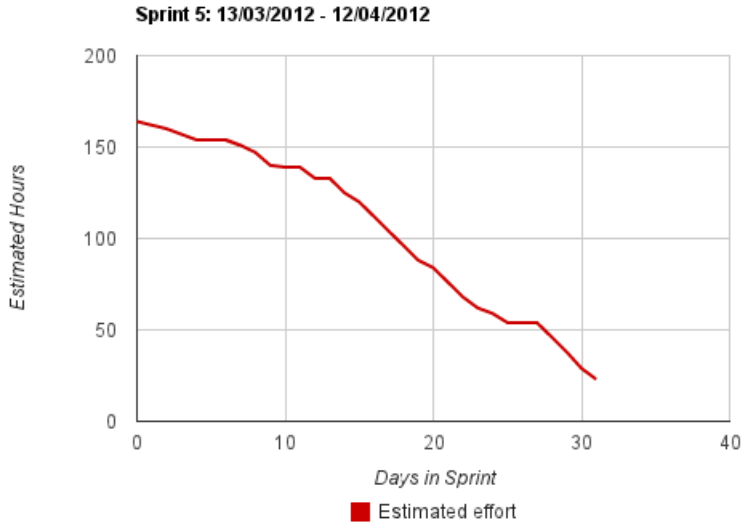

Figura C.5: Gráfico *Burndown* do quinto sprint, em Março/Abril de 2012.

C.6 Abril/Maio - Sprint 6

Neste sprint foi concluída a análise do módulo 5 e feita a sua implementação.

Este foi um dos sprints mais complexos do estágio, por várias razões. Além das tarefas do sprint passado que tinham ficado por acabar, foi preciso implementar todo o sistema de sugestão em relativamente pouco tempo.

Este sprint teve também algumas situações que revelaram que tarefas que já se consideravam concluídas afinal ainda precisavam de algumas horas de trabalho. O momento principal foi no 19º dia de sprint, em que se descobriu que usar a AlchemyAPI seria demasiado desvantajoso financeiramente. Esta constatação obrigou à implementação e teste de soluções de *Tag Extracting*, dando depois origem ao estudo do apêndice K.

Todas estas situações levaram a um atraso de alguns dias na conclusão do sprint, que acabou com o estudo referido acima por completar. Este estudo só foi terminado depois do último sprint, durante a escrita do Relatório Final.

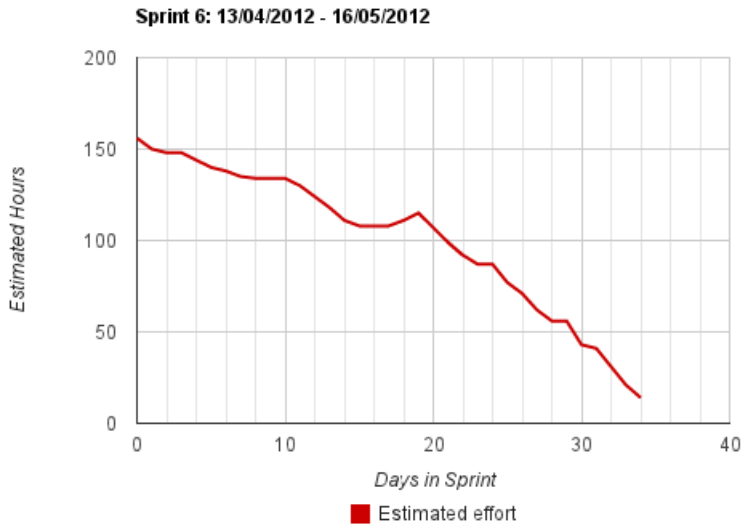

Figura C.6: Gráfico *Burndown* do sexto sprint, em Abril/Maio de 2012.

C.7 Maio/Junho - Sprint 7

Neste sprint foi implementado o módulo 6 - *Full-Text Search* (secção 5.6).

Este módulo foi relativamente simples, sem os problemas que se verificaram no sprint anterior. Mesmo assim, durante o 9º dia de sprint houve uma subida no tempo estimado para conclusão da tarefa "Procura de conteúdo", que foi a descoberta da dificuldade do Solr em lidar com a ordenação de grupos de resultados.

No 23º dia de sprint, fez-se a configuração da ligação do código do Bundlr com o servidor Solr, alojado no serviço WebSolr, o que deu origem a alguns problemas que aumentaram o tempo estimado de conclusão.

A parte final deste módulo foi intercalada com a escrita do Relatório Final.

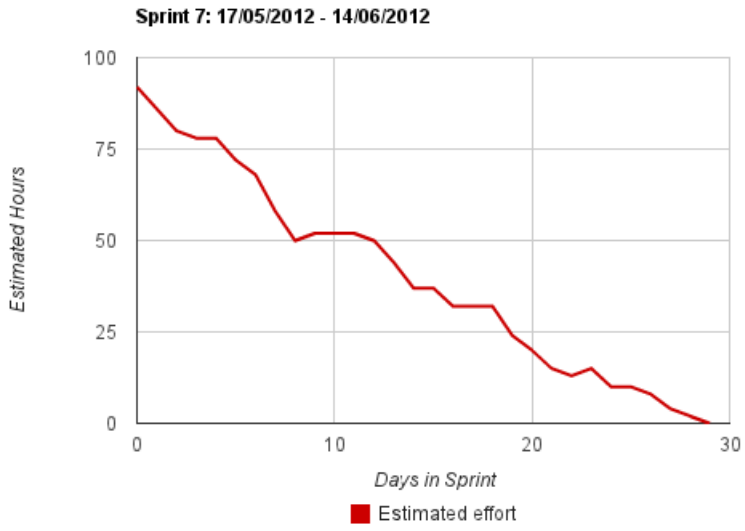

Figura C.7: Gráfico *Burndown* do sétimo sprint, em Maio/Junho de 2012.

Apêndice D

Diagramas de Gantt

O planeamento do estágio tal como inicialmente pensado no Relatório Intermédio é apresentado na figura D.1. Tal como previsto acabaram por haver algumas alterações durante o segundo semestre. A figura D.2 apresenta o planeamento final.

As alterações foram a substituição dos módulos API de leitura e API de escrita pelo módulo *Full-Text Search*, a alteração da data da Defesa Intermédia para dia 16 de Fevereiro e o consequente atraso nos restantes módulos do planeamento, descrito na análise do *burndown* do módulo 4 (apêndice C.4).

ID	Task Name	Duration	Start	Finish
1	**Iniciação**	**22 days**	**Thu 01-09-11**	**Fri 30-09-11**
2	Integração no projecto Bundlr e análise da competição	8 days	Thu 01-09-11	Sun 11-09-11
3	Estudo da arquitectura e código-fonte	11 days	Mon 12-09-11	Sun 25-09-11
4	Definição da metodologia	5 days	Mon 26-09-11	Fri 30-09-11
5	M1 - Bundles privados	12 days	Sat 01-10-11	Sun 16-10-11
6	M2 - Limitar colaboração	11 days	Mon 17-10-11	Mon 31-10-11
7	M3 - Subscrição de bundles e utilizadores	45 days	Tue 01-11-11	Sat 31-12-11
8	**Defesa Intermédia**	**25 days**	**Sun 01-01-12**	**Fri 03-02-12**
9	Elaboração do relatório intermédio	12 days	Sun 01-01-12	Sun 15-01-12
10	Revisão do relatório intermédio	6 days	Mon 16-01-12	Mon 23-01-12
11	Entrega do relatório intermédio	1 day	Tue 24-01-12	Tue 24-01-12
12	Preparação da defesa intermédia	5 days	Wed 25-01-12	Tue 31-01-12
13	Defesa intermédia	3 days	Wed 01-02-12	Fri 03-02-12
14	M4 - Categorização e exploração de bundles	19 days	Sat 04-02-12	Wed 29-02-12
15	M5 - Auto-categorização e sugestões de conteúdo	43 days	Thu 01-03-12	Mon 30-04-12
16	M6 - API de leitura	15 days	Tue 01-05-12	Sun 20-05-12
17	M7 - API de escrita	16 days	Mon 21-05-12	Sun 10-06-12
18	**Defesa Final**	**33 days**	**Mon 11-06-12**	**Wed 25-07-12**
19	Elaboração do relatório final	16 days	Mon 11-06-12	Sun 01-07-12
20	Revisão do relatório final	8 days	Mon 02-07-12	Wed 11-07-12
21	Entrega do relatório final	1 day	Thu 12-07-12	Thu 12-07-12
22	Preparação da defesa final	7 days	Fri 13-07-12	Sun 22-07-12
23	Defesa final	3 days	Mon 23-07-12	Wed 25-07-12

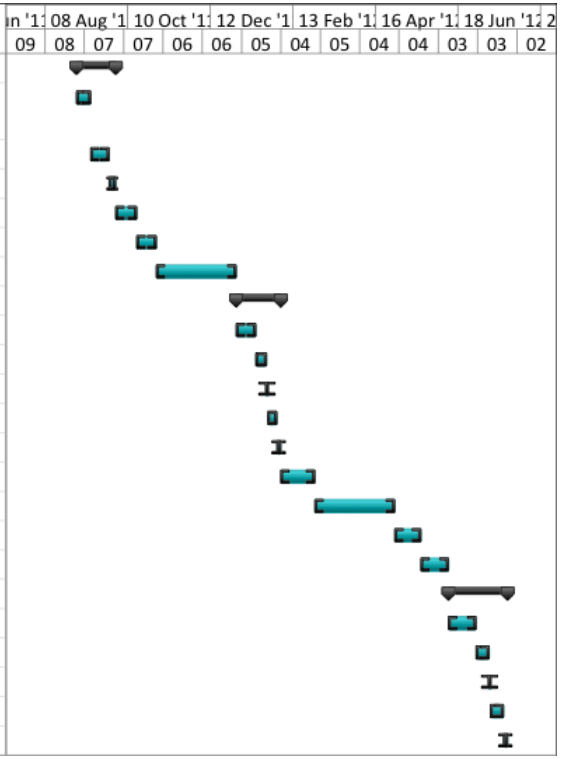

Figura D.1: Diagrama de Gantt com o planeamento inicial, tal como apresentado no Relatório Intermédio. À esquerda encontram-se as tarefas e à direita o diagrama.

ID	Task Name	Duration	Start
1	**Iniciação**	**22 days**	**Thu 01-09-11**
2	Integração no projecto Bundlr e análise da competição	8 days	Thu 01-09-11
3	Estudo da arquitectura e código-fonte	11 days	Mon 12-09-11
4	Definição da metodologia	5 days	Mon 26-09-11
5	M1 - Bundles privados	12 days	Sat 01-10-11
6	M2 - Limitar colaboração	11 days	Mon 17-10-11
7	M3 - Subscrição de bundles e utilizadores	45 days	Tue 01-11-11
8	**Defesa Intermédia**	**34 days**	**Sun 01-01-12**
9	Elaboração do relatório intermédio	12 days	Sun 01-01-12
10	Revisão do relatório intermédio	6 days	Mon 16-01-12
11	Entrega do relatório intermédio	1 day	Tue 24-01-12
12	Preparação da defesa intermédia	16 days	Wed 25-01-12
13	Defesa intermédia	1 day	Thu 16-02-12
14	M4 - Categorização e exploração de bundles	26 days	Mon 06-02-12
15	M5 - Sugestão de conteúdo	47 days	Tue 13-03-12
16	M6 - Full-Text Search	21 days	Thu 17-05-12
17	**Defesa Final**	**37 days**	**Tue 05-06-12**
18	Elaboração do relatório final	16 days	Tue 05-06-12
19	Revisão do relatório final	8 days	Mon 02-07-12
20	Entrega do relatório final	1 day	Thu 12-07-12
21	Preparação da defesa final	7 days	Fri 13-07-12
22	Defesa final	3 days	Mon 23-07-12

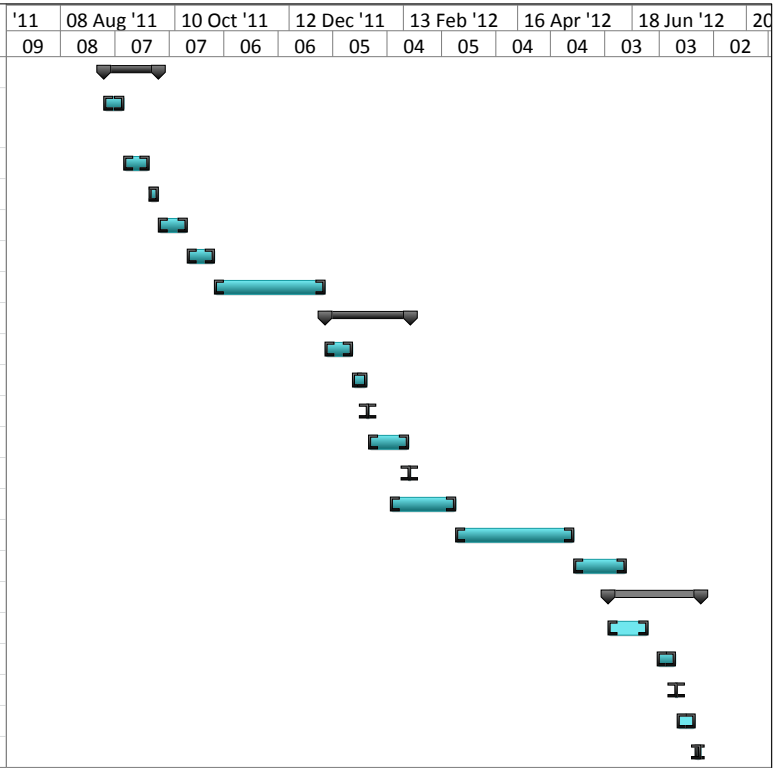

Figura D.2: Diagrama de Gantt com o planeamento final. À esquerda encontram-se as tarefas e à direita o diagrama.

Apêndice E

Diagramas de modelos e controladores

Neste apêndice apresentam-se diagramas de modelos e de controladores, que pretendem dar uma visão mais técnica sobre o sistema.

Os diagramas de modelos apresentam os modelos do padrão MVC, apresentando os seus atributos e métodos. Nestes diagramas são também apresentados alguns módulos e classes que interagem com os modelos e que facilitam a sua compreensão.

Já os diagramas de controladores apresentam os controladores do padrão MVC, tal como utilizados pelo Ruby on Rails. Estes apresentam apenas os métodos de cada controlador, já que são estes que respondem aos pedidos à aplicação.

Estes diagramas são, no fundo, diagramas de classes, embora se tenha optado por não incluir os atributos das funções, já que não são fundamentais para a compreensão do diagrama e iriam torná-lo bastante mais complicado de analisar.

Na primeira secção do apêndice apresentam-se os diagramas inicias, seguindo-se os diagramas das alterações em cada módulo.

E.1 Estrutura inicial

E.1.1 Modelos

Para ser possível representar os modelos inicias foi necessário esconder os seus métodos, por serem em grande número. Foram também excluídos os módulos pela mesma razão, dando origem ao diagrama simplificado E.1.

Este diagrama mostra os modelos mais importantes do sistema, que são

descritos de seguida:

User Modelo que representa os utilizadores da aplicação. O utilizador pode ter vários *Bundles*, vários *CollaborationRequests*, vários bundles favoritos e um *NotificationSettings*.

Bundle É um agregado de vários *Clips*, com um *title*, *description* e *photo* que ajudam a descrever o tópico dessa agregação. Um *Bundle* pertence a um *User* e pode ter, para além dos clips um *BundleStats* e vários *Collaborators*.

Clip Unidade de conteúdo do Bundlr. Em cada clip são armazenados dados específicos da sua fonte (Twitter, Vimeo, Flickr...), para além dos listados. Um *Clip* pertence a um *Bundle* e pode ter uma *Note*.

Note Um *Clip* pode incorporar uma nota, que é uma anotação textual sobre o seu conteúdo.

Collaborator Representa a funcionalidade de colaboração. Na arquitectura original, um utilizador podia colaborar livremente em bundles de outras pessoas, que o convidavam através de um *CollaborationRequest*

CollaborationRequest Um pedido de colaboração de um autor de um bundle a outro utilizador. O utilizador tem de aceitar o pedido para passar a ser colaborador.

NotificationSettings Conjunto de booleanos que descrevem as escolhas do utilizador no que toca ao envio de emails de notificação.

BundleStats Um bundle tem associadas uma série de estatísticas relativas à sua popularidade e ao seu número de visualizações, que são guardadas neste modelo.

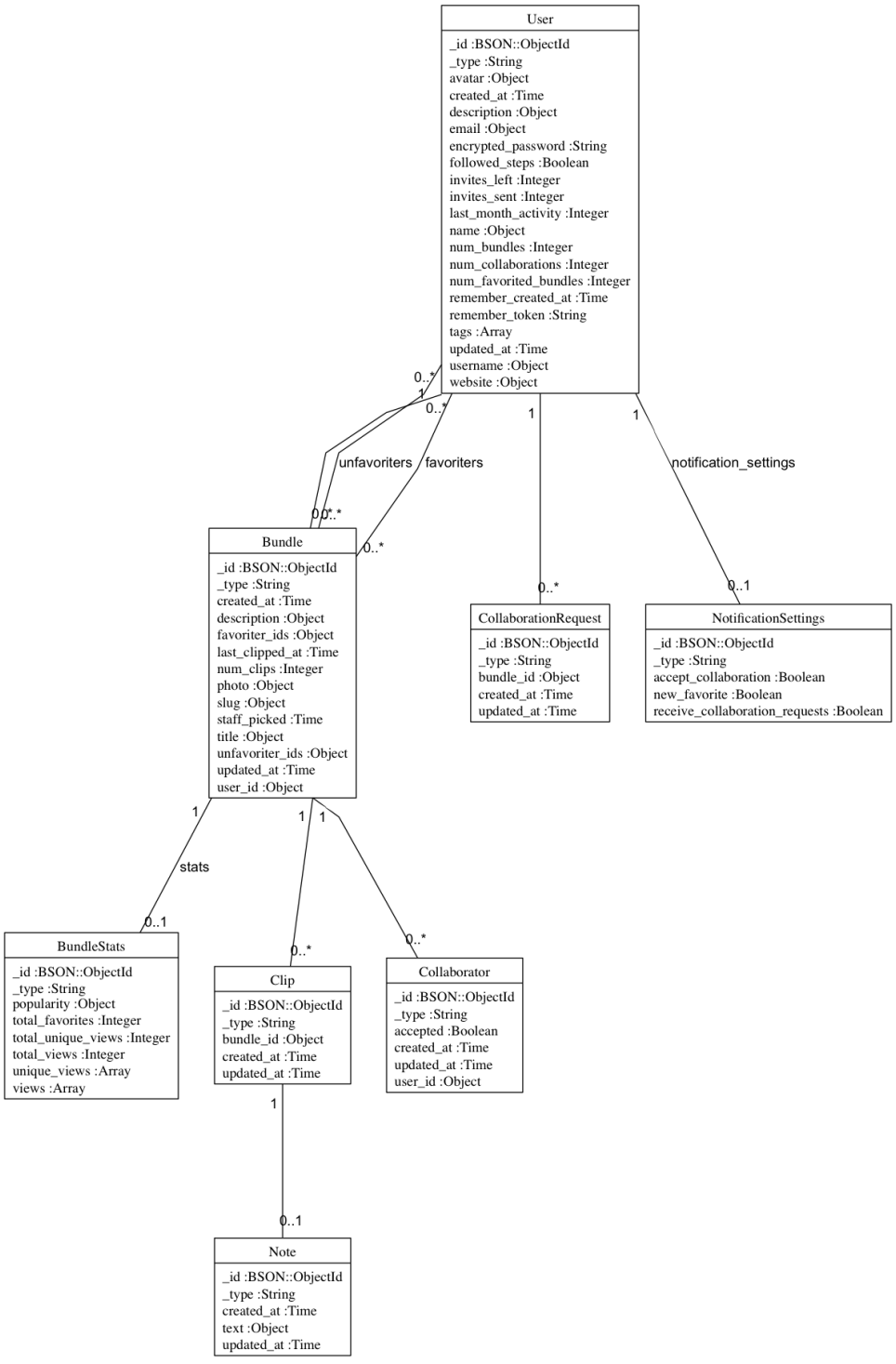

Figura E.1: Diagrama de Modelos simplificado (sem métodos nem módulos) da arquitectura inicial do sistema.

E.1.2 Controladores

Os controladores são chamados conforme a página que o utilizador visitar. Por exemplo, se o utilizador for para a página de um bundle, o controlador responsável por responder ao pedido será o *BundlesController*, através do método *show*. Se o utilizador abrir a página de edição de um bundle, será activado o método *edit* do *BundlesController*, etc.

O diagrama E.2 apresenta os controladores do sistema original, que são descritos de seguida:

AccountController Gere as acções relacionadas com a página de perfil de um utilizador.

BackController Contém as acções do *Dashboard*, uma secção do website apenas acessível a administradores. O nome deste controlador foi mais tarde alterado para *DashboardController*.

BookmarkletController É responsável pelos pedidos feitos no *bookmarklet* e na extensão do Bundlr, onde o utilizador pode criar clips.

BundlesController Gere as acções da página de bundle, incluindo a sua criação e edição.

ClipsController Responsável pelas acções de edição da nota do clip, pela imagem de capa do bundle e pela destruição do próprio clip.

CollaborationRequestController Gere as acções de resposta a um pedido de colaboração.

CollaboratorsController Permite listar e editar a lista de colaboradores de um bundle.

FrontController Faz a gestão das páginas públicas ou institucionais do serviço.

SearchController Responsável pela pesquisa no website.

Por convenção do Ruby on Rails, todos os controladores estendem do *ApplicationController*, que contém métodos de utilidade transversal.

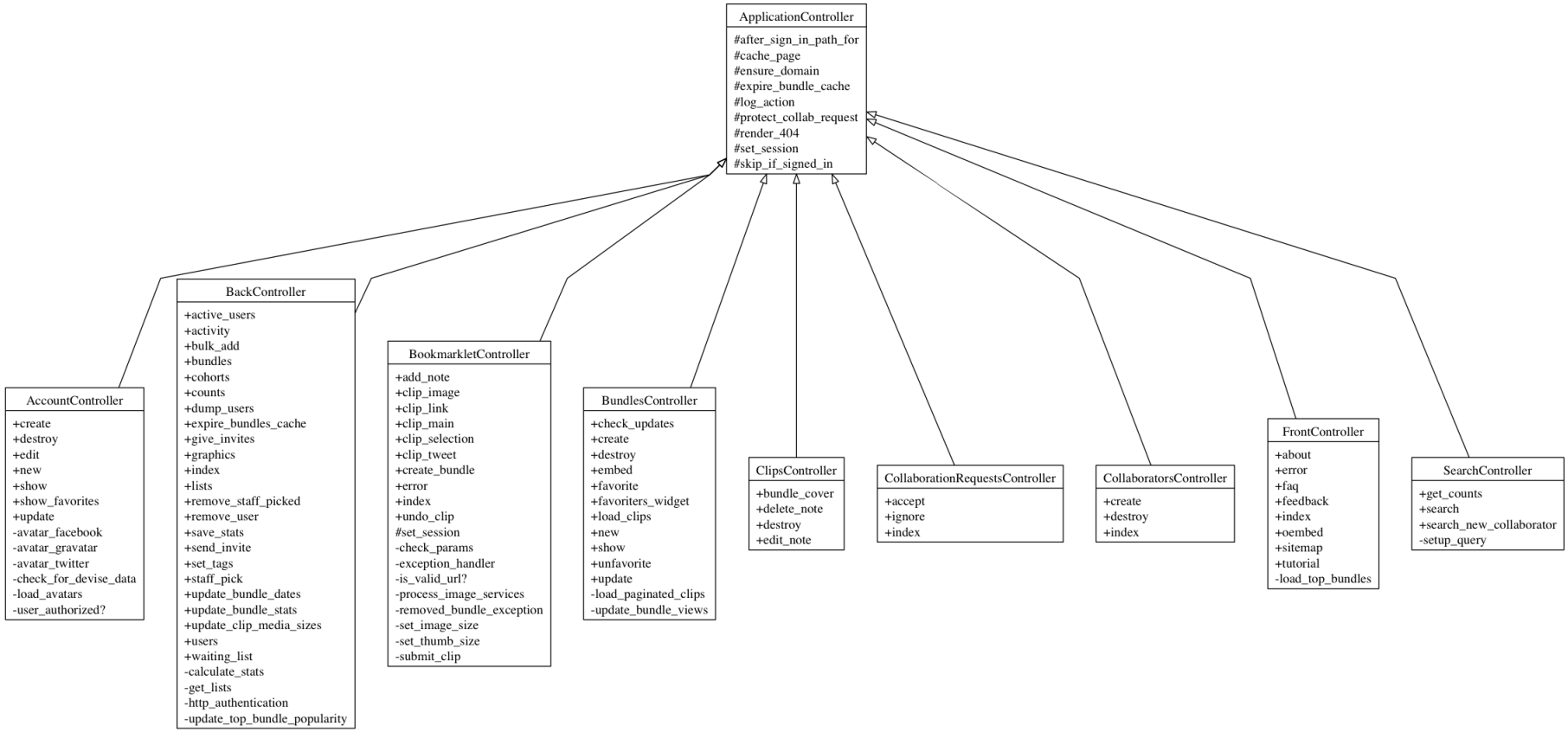

Figura E.2: Diagrama de Controladores da arquitectura inicial do sistema.

E.2 Alterações no Módulo 1

E.2.1 Alterações nos Modelos

As alterações feitas durante este módulo nos modelos da arquitectura MVC do sistema são apresentadas no diagrama E.3.

Destas, importa destacar o método que verifica se o utilizador pode criar bundles privados (*can_set_private?*), os métodos de controlo de acesso aos bundles (*can_admin?*, *can_edit?* e *can_view?*) e as alterações nos métodos estáticos que fazem a recuperação dos bundles da base de dados (*all_public*, *popular*, *staff_picked* e *visible*).

O método *visible* é o responsável pelas alterações na pesquisa que se analisaram durante a secção de implementação deste módulo. Os métodos *all_public*, *popular* e *staff_picked* foram alterados para não devolverem bundles privados. O método *visible_bundle*, no modelo *User* devolve os bundles da instância utilizador, excluindo aqueles a que o *current_user* (o utilizador que está a utilizar o Bundlr) não tem acesso.

E.2.2 Alterações nos Controladores

As alterações feitas durante este módulo nos controladores da arquitectura MVC do sistema são apresentadas no diagrama E.4.

Antes de mais, alguns dos métodos dos controladores foram protegidos tendo em conta a existência dos bundles privados. Os métodos *show_favorites*, *check_updates*, *embed*, *favorite*, *favoriters_widget*, *load_clips*, *show_favorites*, *unfavorite*, *update* e *show* do controlador *BundlesController* passaram a verificar se o *current_user* tem permissões para aceder ao bundle em questão, mostrando um erro HTTP 404 caso contrário.

Os métodos *submit_clip* e *create* passaram a poder criar bundles privados, e o *show* do *AccountController* passou a esconder os bundles privados do utilizador quando mostra a sua página.

Finalmente, no *SearchController* foram alterados os métodos correspondentes à funcionalidade de pesquisa.

E.3 Alterações no Módulo 2

E.3.1 Alterações nos Modelos

As alterações feitas durante este módulo nos modelos da arquitectura MVC do sistema são apresentadas no diagrama E.5.

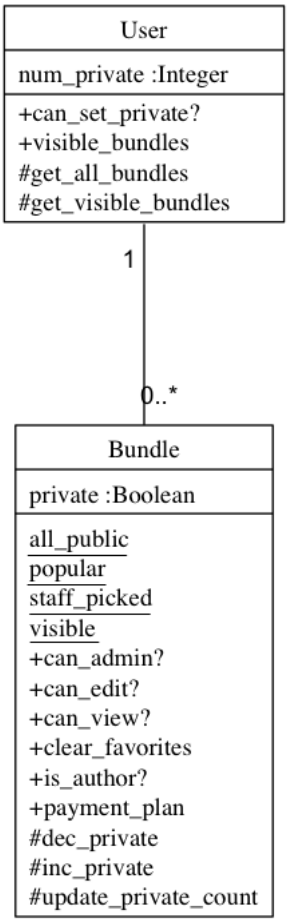

Figura E.3: Diagrama de Modelos das alterações no Módulo 1: Bundles Privados.

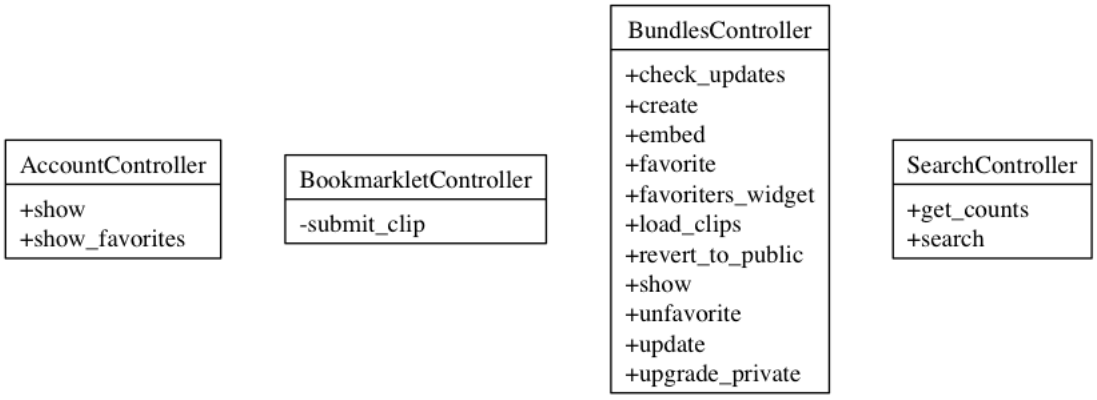

Figura E.4: Diagrama de Controladores das alterações no Módulo 1: Bundles Privados.

Este módulo teve alterações muito simples, que passaram pela alteração de alguns métodos de controlo de acesso aos bundles e pela criação do método *can_add_collaborators?* que verifica se um utilizador pode ou não adicionar colaboradores aos seus bundles.

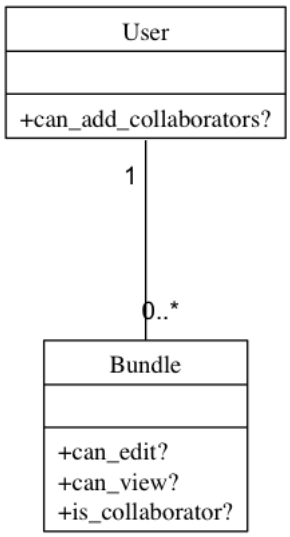

Figura E.5: Diagrama de Modelos das alterações no Módulo 2: Limitar Colaboração.

E.3.2 Alterações nos Controladores

As alterações feitas durante este módulo nos controladores da arquitectura MVC do sistema são apresentadas no diagrama E.6.

Estas alterações também foram simples, focando-se principalmente no reforço da protecção de alguns métodos para permitir a colaboração apenas a utilizadores pagantes. Métodos como o *remove_collaborators* e o *upgrade_collab* tratam de alguns casos em que o utilizador já não paga pelo serviço e quer remover os colaboradores para poder continuar a actualizar o bundle.

E.4 Alterações no Módulo 3

E.4.1 Alterações nos Modelos

As alterações feitas durante este módulo nos modelos da arquitectura MVC do sistema são apresentadas no diagrama E.7.

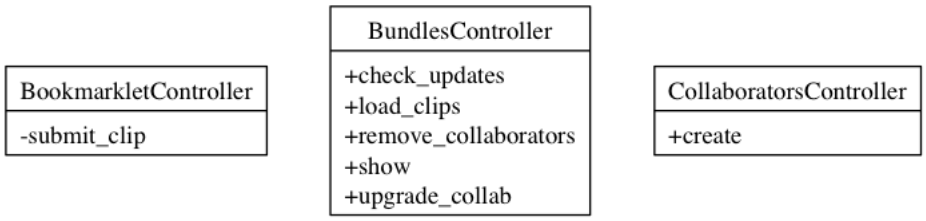

Figura E.6: Diagrama de Controladores das alterações no Módulo 2: Limitar Colaboração.

Durante este módulo foram removidas as ligações da funcionalidade bundles favoritos, agora substituída pela subscrição. Foram criadas as relações que representam as interacções sociais (*followed_bundles*, *followed_users* e *followers*) e os métodos para as gerir (*add_followers*, *followed_by?*, *follows?*, *num_followed_bundles*, *num_followed_users*, *remove_follower* e *total_followers*).

Acrescenta-se também a classe *StreamObserver*, que é um *Mongoid::Observer* e que monitoriza os modelos *User*, *Bundle* e *Clip*. Quando detecta alguma actualização (*after_update*), verifica se foi alterado algum dos campos a que o Streama faz *cache* (*updated_cached_fields?*) e, se assim for, actualiza o registo dessa *cache* na base de dados (*refresh_data*). Qualdo uma instância dos modelos observados é destruída (*after_destroy*), são também destruídas todas as suas actividades.

O modelo *StreamEvent* guarda as actividades. Além dos campos definidos pelo Activity Streams 1.0, é definido o atributo *receivers* que guarda a lista de receptores de cada actividade. Não são mostradas as relações com as outras tabelas por não serem implementadas de forma explícita - o Streama guarda um dicionário chave-valor com o identificador do objecto e o seu tipo de classe. O atributo *receivers* é um *array* destes dicionários.

É também adicionado o *StreamMailer*, que se encarrega de enviar os emails de notificação quando um utilizador ou um dos seus bundles ganha um novo subscritor. São também criadas duas variáveis no *NotificationSettings*, que permitem que o utilizador desactive o envio de emails nestas ocasiões.

No modelo *User* adicionam-se alguns atributos e métodos para facilitar a distinção entre actividades lidas e não lidas, para depois ser possível a distinção na *interface*.

E.4.2 Alterações nos Controladores

As alterações feitas durante este módulo nos controladores da arquitectura MVC do sistema são apresentadas no diagrama E.8.

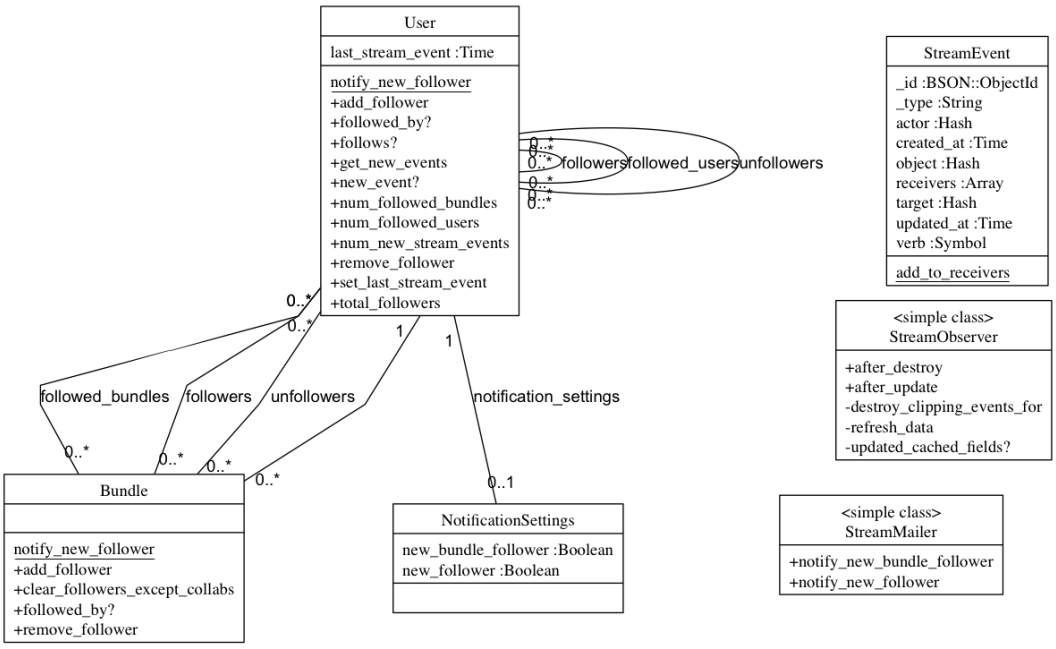

Figura E.7: Diagrama de Modelos das alterações no Módulo 3: Subscrição de bundles e utilizadores.

Foram criados os métodos para lidar com as interacções sociais no *AccountController* e no *BundlesController*. Foi também criado o *ActivityStreamController* que gere a página da *activity stream* de cada utilizador.

As restantes alterações foram feitas em métodos que passaram a gerar actividades quando são executados. Um exemplo destes métodos é o *staff_pick*, que assim que executado cria uma actividade.

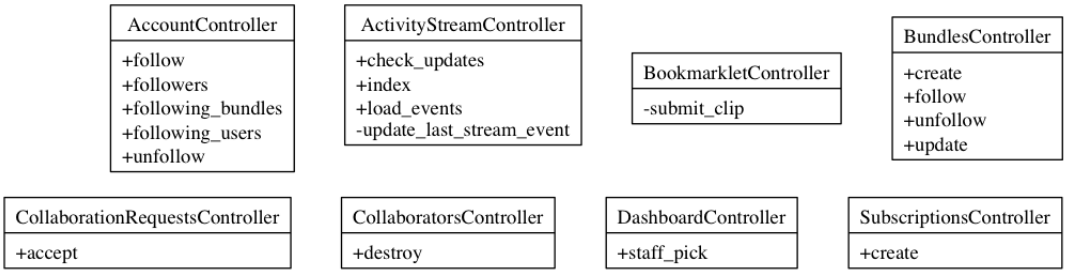

Figura E.8: Diagrama de Controladores das alterações no Módulo 3: Subscrição de bundles e utilizadores.

E.5 Alterações no Módulo 4

E.5.1 Alterações nos Modelos

As alterações feitas durante este módulo nos modelos da arquitectura MVC do sistema são apresentadas no diagrama E.9.

Este módulo implementa as funcionalidades de um modelo que tem categorias no módulo (de *module*) *Categories*. Este módulo cria o campo *category* nos modelos que o incluem, o que é o caso do *Bundle*, e define também as 8 categorias existentes.

Foi também alterada a função do cálculo de popularidade e criada a classe *ExploreMailer*, que notifica o utilizador quando o seu bundle é seleccionado para aparecer na página de exploração. O atributo *homepaged* do modelo *Bundle* faz o registo temporal dessa selecção.

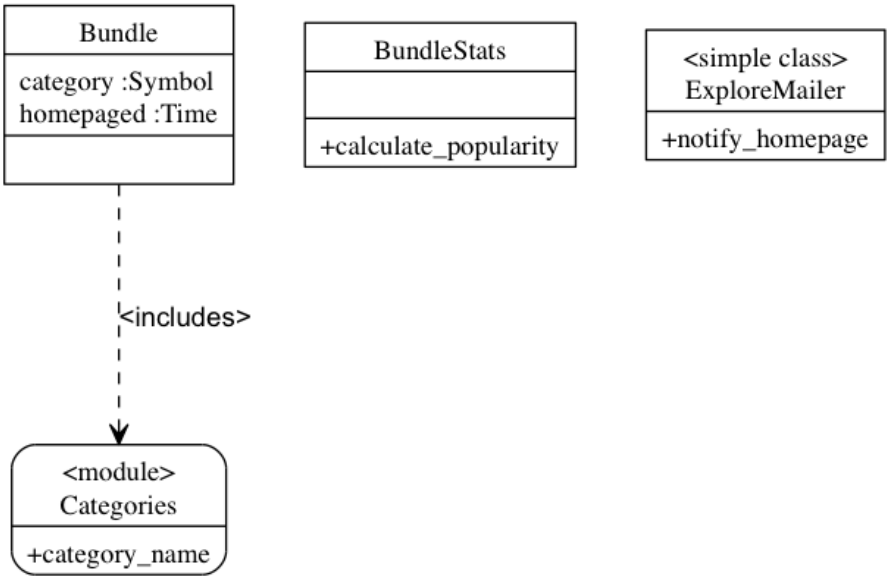

Figura E.9: Diagrama de Modelos das alterações no Módulo 4: Categorização e exploração de bundles.

E.5.2 Alterações nos Controladores

As alterações feitas durante este módulo nos controladores da arquitectura MVC do sistema são apresentadas no diagrama E.10.

No *DashboardController* estão presentes os métodos que dão origem ao Categorizador, a secção do *dashboard* (uma área a que apenas os administradores têm acesso) que permite fazer a categorização prévia de bundles.

O *ExploreController* gere a página de exploração e as páginas das categorias.

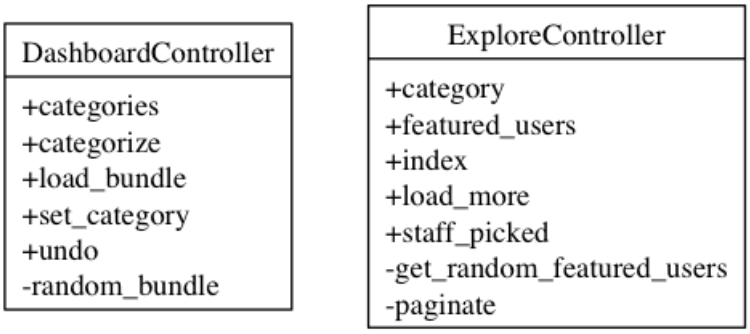

Figura E.10: Diagrama de Controladores das alterações no Módulo 4: Categorização e exploração de bundles.

E.6 Alterações no Módulo 5

E.6.1 Alterações nos Modelos

As alterações feitas durante este módulo nos modelos da arquitectura MVC do sistema são apresentadas no diagrama E.11.

Este módulo levou a um grande número de alterações nos modelos do sistema, reflectindo o trabalho de implementação do sistema de sugestão.

A ligação do motor de sugestão com o resto do sistema começa pela inclusão do módulo *Tagger::Taggable* por parte dos modelos *User*, *Bundle* e *Clip*. Quando incluído, este módulo cria uma relação 1 para 1 de nome *tag_list* com o modelo *Tagger::TagList*. Este modelo tem a função de guardar a lista de tags do modelo que incluíu o módulo *Tagger::Taggable*. Desta forma, ao guardar as tags de um bundle através do método *tags=* definido pelo módulo, é criada a instância do modelo *Tagger::TagList* que as guarda no seu atributo *tags*.

O modelo *Tagger::TagList* contém também dois métodos estáticos que permitem recuperar da base de dados todos os objectos *taggable* de determinado tipo (neste caso utilizadores, bundles ou clips). Além disso, este modelo inclui o módulo *Suggestion::HasSuggestions*, que implementa todos os métodos de gestão das listas de sugestões.

Como se pode ver pelos métodos *bundle_suggestions*, *clip_suggestions* e *user_suggestions*, o módulo *Suggestion::HasSuggestions* torna possível criar e guardar sugestões de vários tipos. A decisão de guardar as sugestões nas

TagLists vem de uma tentativa de não encher os modelos principais (*User*, *Bundle* e *Clip*) com dados que são raramente utilizados. Desta forma, não se diminui o desempenho da sua recuperação da base de dados.

Definida a forma como são guardados os dados, resta obter as tags e calcular as sugestões. O processo começa no *SuggestionObserver*, que monitoriza o modelo *Clip* à espera que algum seja criado. Quando um clip é criado, é extraído o seu conteúdo através do Readability modificado pela equipa, que é invocado no método *get_article* do módulo *Retriever*. Depois de extraído o conteúdo, o *SuggestionObserver* executa o método *get* do módulo *Tagger*.

O método *get* executa o processador indicado para cada tipo de clip. Por exemplo, no caso de um clip de um vídeo do YouTube, o *get* executa o método *tag* do *Tagger::YoutubeProcessor*, que neste caso utiliza a biblioteca youtube_it para recolher as tags do vídeo da API do YouTube. O processador *Tagger::LinkProcessor* implementa a solução de *Tag Extraction* escolhida na análise do Apêndice K.

Finalmente, depois de obter a resposta do método *get* do *Tagger* sob a forma de um *array* de tags, o *SuggestionObserver* guarda as tags no clip, no seu bundle e no seu autor.

Para gerir a tarefa do cálculo de sugestões, foi criado o módulo *Suggestion* que é utilizado pelos *scripts* diário e semanal, não representados no diagrama.

Por último, as classes *Suggestion::JaccardCoefficient*, *Suggestion::DiceCoefficient*, *Suggestion::WeightedCoefficient*, *Suggestion::OverlayCoefficient* e *Suggestion::CommonTagsPercentageRepeat*, implementam o método *calculate_similarity*, que calcula a semelhança entre dois *arrays* de tags.

E.6.2 Alterações nos Controladores

As alterações feitas durante este módulo nos controladores da arquitectura MVC do sistema são apresentadas no diagrama E.12.

O baixo número de alterações no diagrama de controladores reflecte as poucas *User Stories* implementadas neste módulo. Fez-se apenas a alteração do método *suggestions* no controlador *ActivityStreamController*, que anteriormente sugeria bundles e utilizadores populares, passando a fazer sugestões personalizadas.

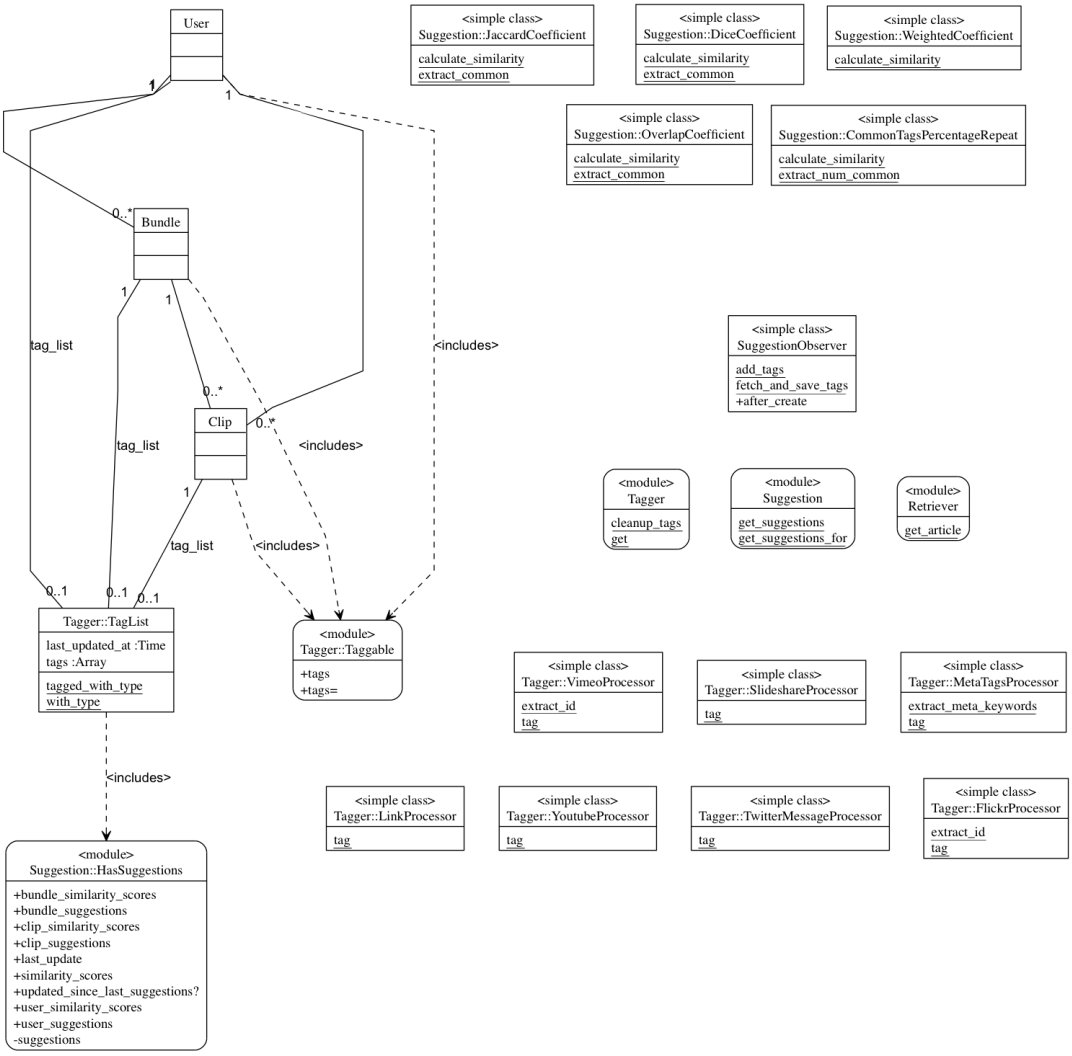

Figura E.11: Diagrama de Modelos das alterações no Módulo 5: Sugestão de conteúdo.

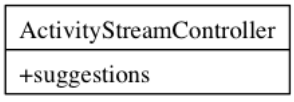

Figura E.12: Diagrama de Controladores das alterações no Módulo 5: Sugestão de conteúdo.

E.7 Alterações no Módulo 6

E.7.1 Alterações nos Modelos

As alterações feitas durante este módulo nos modelos da arquitectura MVC do sistema são apresentadas no diagrama E.13.

Neste módulo, os modelos *User* e *Clip* passam a incluir o módulo *Sunspot::Mongoid*, que define os métodos *searchable*, *searchable?* e *solr_search*. Estes métodos são a ponte de ligação do sistema com as bibliotecas Sunspot e Sunspot Mongoid e permitem definir os atributos a indexar no modelo e fazer a pesquisa.

No modelo *User* são criados métodos devolvem uma lista de IDs dos bundles privados e de todo o conjunto de bundles de um utilizador, que são depois usados para filtrar a pesquisa nos bundles bundles pessoais desse utilizador.

Os modelos *Bundle* e *Note* definem métodos para que o clip possa ser notificado quando são alterados, para que seja feita a actualização do índice do servidor Solr.

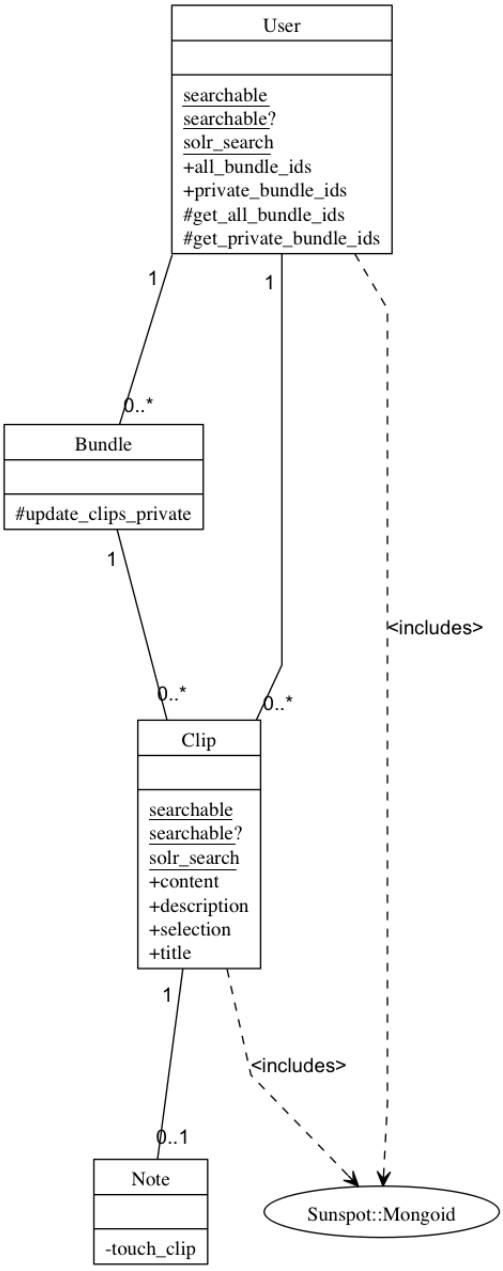

Figura E.13: Diagrama de Modelos das alterações no Módulo 6: *Full-Text Search*.

E.7.2 Alterações nos Controladores

As alterações feitas durante este módulo nos controladores da arquitectura MVC do sistema são apresentadas no diagrama E.14.

Estas alterações foram concentradas no controlador *SearchController*, que está encarregue da pesquisa. Foi alterado o método principal da pesquisa e foram divididas as procuras em dois tipos: *search_content* e *search_users*.

O método *get_content_results* trata dos problemas de paginação descritos neste módulo.

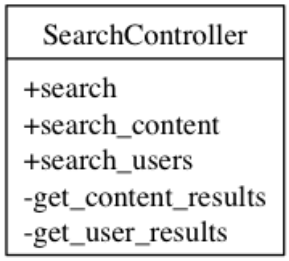

Figura E.14: Diagrama de Controladores das alterações no Módulo 6: *Full-Text Search*.

Apêndice F

Estado da Arte do Serviço

O mercado actual de serviços que pretendem seleccionar e organizar conteúdo é bastante concorrido. Nesta categoria, podem ser identificados variados serviços, desde serviços de utilização privada a serviços que apenas servem para publicar o conteúdo seleccionado para determinada audiência. Para facilitar a distinção do Bundlr neste conjunto de serviços concorrentes, identificaram-se os seguintes grupos de ferramentas:

Ferramentas de Curadoria Focam-se em disponibilizar as funcionalidades necessárias à filtragem e selecção de conteúdo dentro de determinado tópico e à sua publicação para a audiência do utilizador. O valor encontra-se nos diferentes tipos de *media* que estas plataformas normalmente suportam (não são só hiperligações) e no facto do conteúdo ser filtrado pelo *curador*, que certifica a sua qualidade.

Ferramentas de *Bookmarking* Privado Têm como objectivo principal a organização e manutenção de conteúdo seleccionado ou criado pelo utilizador. São normalmente criadas para uso pessoal, podendo ter a opção de partilha.

Ferramentas de *Bookmarking* Social Partem da abertura das ferramentas de *Bookmarking* Privado à rede social do utilizador, que deixa de a usar apenas como um repositório de hiperligações online para passar a ser também um local de partilha dessas hiperligações com os seus contactos.

Para cada um dos grupos foram seleccionados alguns serviços que serão usados na maioria das comparações ao longo deste trabalho. A tabela F.1 associa-os ao seu grupo e a tabela F.2 apresenta uma comparação geral dos serviços (incluindo o Bundlr), seguindo-se depois uma análise mais detalhada

de cada um. Finaliza-se a secção com uma pequena comparação desses serviços com o Bundlr. No Capítulo 5 é feita uma análise mais detalhada da competição, sob o prisma das funcionalidades a implementar no estágio.

F.1 Ferramentas de Curadoria

F.1.1 Storify

O Storify[1] é um serviço em que os utilizadores juntam conteúdo e os seus próprios comentários para criar histórias temáticas, que vão actualizando ao longo do tempo. O público-alvo do serviço é constituído principalmente por jornalistas, que o usam, por exemplo, para criar relatos cronológicos de eventos reais através das reacções nas redes sociais.

O serviço contém algumas funcionalidades especialmente pensadas para este tipo de utilização, como a capacidade de incorporar (*embed*) a página noutros websites, útil para jornalistas que não queiram enviar a sua audiência para o website do serviço, e a actualização do conteúdo em tempo-real.

F.1.2 Scoop.it

O Scoop.it[2] opta por uma estratégia diferente: o utilizador cria páginas temáticas em estilo revista que outros utilizadores podem seguir. Este serviço não tem o foco no jornalismo que o Storify tem, dirigindo-se mais a *bloggers*, especialistas e entusiastas.

Este serviço disponibiliza também um pequeno *slideshow* que se pode incorporar noutros websites e que mostra as últimas selecções feitas pelo utilizador num determinado tópico.

[1] http://storify.com
[2] http://scoop.it

Tipo de Ferramenta	**Curadoria**		***Bookmarking* Privado**	***Bookmarking* Social**		
Serviços	Storify	Scoop.it	Evernote	Delicious	Zootool	Gimme Bar

Tabela F.1: Serviços competidores ao Bundlr agrupados por tipo.

Serviço	**Lançamento**	**Aplicação Móvel**	**Colecções**	**Forma de Selecção**	**Tipo de Conteúdo Suportado**
Storify	2010	Não	Stories	Bookmarklet, Pesquisa interna	Links, Vídeos, Imagens, Mensagens em redes sociais, Resultados de pesquisas, Feeds
Scoop.it	2010	iOS + Android	Topics	Bookmarklet, Sistema de sugestão	Links, Vídeos, Imagens, Mensagens em redes sociais, Resultados de pesquisas, Artigos, Feeds
Evernote	2008	iOS + Android + Win. Phone	Notebooks	Bookmarklet, Formulário	Links, Imagens, Documentos
Delicious	2003	iOS + Android (de terceiros)	Stacks	Bookmarklet, Extensões	Links
Zootool	2007	iPhone	Packs	Bookmarklets, Extensões	Links, Imagens, Vídeos, Documentos
Gimme Bar	2011	iPhone	Collections	Bookmarklet	Links, Imagens, Vídeos, Mensagens em redes sociais
Bundlr	2011	Não	Bundles	Bookmarklet, Extensões	Links, Imagens, Vídeos, Mensagens em redes sociais, Artigos, Selecção de texto

Tabela F.2: Comparação geral entre os serviços analisados.

F.2 Ferramentas de *Bookmarking* Privado

F.2.1 Evernote

O Evernote[3] é um serviço muito popular, que tem como objectivo guardar as notas dos utilizadores. Com presença em *Desktop*, *iPhone*, *Android*, *BlackBerry*, *Windows Phone* e, claro, na Internet, é um dos serviços de captura de conteúdo privado mais usados, com 11 milhões de utilizadores[34].

O utilizador pode criar pastas, adicionar entradas com texto e imagens e até mesmo guardar integralmente páginas da Internet, não apenas o link. Além disso, pode também criar colecções em colaboração com outros utilizadores, embora esse não seja o foco do serviço, já que não é dado qualquer destaque a essa funcionalidade.

F.3 Ferramentas de *Bookmarking* Social

F.3.1 Delicious

O Delicious[4] é um dos serviços de online *bookmarking* mais antigos (2003). O seu propósito é simples: guardar os links favoritos dos seus utilizadores e permitir que cada um possa seguir outros. Ao mesmo tempo, o utilizador é encorajado a categorizar os links que adiciona, tornando-os assim indexáveis pelo serviço em si, que os apresenta depois em listas de categorias. O serviço foi recentemente alvo de uma grande actualização que passou a permitir aos utilizadores criarem conjuntos temáticos de links, chamados *Stacks*. Com essa alteração, e numa tentativa de atingir o público em geral, retiraram também algumas funcionalidades avançadas, causando uma resposta negativa dos utilizadores mais experientes, levando a que muitos abandonassem o serviço[35].

F.3.2 Zootool

Ao contrário do Delicious, o Zootool[5] faz parte de um novo conjunto de ferramentas de *bookmarking* que permite guardar mais do que o link para as páginas de interesse. De facto, o utilizador pode guardar imagens, vídeos e documentos, para além dos já esperados links. A componente social do serviço traduz-se na possibilidade de subscrever a lista de selecções da nossa rede de contactos e na disponibilização do conteúdo público agregado pela

[3] http://evernote.com
[4] http://delicious.com
[5] http://zootool.com

comunidade. O conteúdo pode ser categorizado e até agrupado em *Packs*, equivalentes aos *Stacks* do Delicious, que são por defeito privados. A publicação destas colecções é uma das funcionalidades pagas do serviço.

F.3.3 Gimme Bar

O Gimme Bar[6] é um serviço que se foca mais na colecção de elementos visuais, como imagens e vídeos. O utilizador também pode adicionar links, sendo automaticamente recolhido um *screenshot* do site que melhora a apresentação da colecção de conteúdo. Este serviço também permite a subscrição do conteúdo seleccionado por outros utilizadores e passar facilmente uma colecção de pública para privada e vice-versa.

F.4 O que distingue o Bundlr

Uma das principais características distintivas deste serviço passa pela experiência de utilização, que foi idealizada para facilitar a utilização mesmo por utilizadores com menos experiência tecnológica. A selecção do conteúdo, feita através do *bookmarklet* e das extensões dos *browsers*, pode ser conseguida em poucos cliques, ao contrário de soluções mais complexas como a do Storify e do Scoop.it.

Comparativamente com serviços como o Storify, Scoop.it, Zootools e Gimme Bar, o Bundlr destaca-se por se apresentar como uma ferramenta adequada também para o público menos técnico, uma ferramenta para o público em geral, enquanto que as outras plataformas tendem a seguir um âmbito específico, como notícias, moda ou conteúdo visual.

Em termos tecnológicos, o Bundlr distingue-se de alguns dos seus competidores por oferecer colaboração, actualização do conteúdo das colecções em tempo-real e por permitir incorporar (*embed*) um bundle no website ou blogue do utilizador.

[6] http://gimmebar.com

Apêndice G

Análise: Representação de Requisitos

Este apêndice apresenta uma comparação entre algumas formas de representação dos requisitos e justifica a utilização das *User Stories* durante este estágio.

Apresentam-se de seguida as 3 formas mais utilizadas de descrever requisitos, para comparação:

Listas de requisitos Cada requisito é descrito de forma simples e contida, sem fazer referência aos outros requisitos ou às ligações entre eles.

Casos de Uso Os requisitos são descritos como interacções entre o utilizador e o sistema ou entre dois sistemas. O processo é descrito em detalhe e deve ser suficientemente esclarecedor para que possa ser percebido sem conhecimento do resto do sistema.

User Stories Tal como nos Casos de Uso, os requisitos são descritos como interacções, mas de uma forma menos formal. São escritos usando linguagem não técnica e são bastante menos específicos que os Casos de Uso, deixando lugar para alterações de pequena escala. É também comum anexar a cada uma um conjunto de *Acceptance Criteria* que servem tanto para clarificar questões mais ambíguas como para fornecer uma plataforma de validação e de teste para os intervenientes na sua concepção.

Neste estágio a escolha recaiu sobre as *User Stories*[36] em conjunto com as *Acceptance Stories* (também conhecidas por *Acceptance Criteria*[37]), pelas seguintes razões:

Formalização simples As *User Stories* proporcionam uma forma simples de formalizar os requisitos, que se torna interessante tendo em conta a agilidade da metodologia usada. Mantém-se a simplicidade da Lista de requisitos, mantendo também as relações entre eles.

Adaptação à metodologia usada Esta formalização adapta-se bastante à metodologia usada no estágio por permitir fazer alterações muito rápidas no caso de ser necessária a modificação dos requisitos, devido à sua pequena dimensão.

Proximidade com o cliente A falta de detalhe característica das *User Stories* torna-as adequadas para iniciar a discussão entre o programador e o cliente. Como no decorrer deste estágio o cliente e o autor se encontram sempre no mesmo espaço físico, não há impedimentos a discussões de esclarecimento de detalhes relacionados com os requisitos.

Validação definida Com a utilização das *Acceptance Stories* torna-se fácil completar a *User Story* tanto a nível de funcionalidades não detalhadas como a nível de validações. Tendo as validações definidas, é possível que o cliente confirme se a implementação foi a desejada inicialmente. Além disso, este tipo de validações pode facilmente ser automatizada com recurso a algumas bibliotecas que serão apresentadas na secção de testes ainda neste capítulo, o que as torna bastante interessantes para a confirmação da integridade do sistema.

O formato de *User Story* adoptado foi o definido por Mike Cohn[36] e que se apresenta de seguida:

```
Como <papel desempenhado>,
Quero poder <funcionalidade>,
De forma a <benefício>.
```

Note-se que algumas *User Stories* não contêm a última parte do formato por se poder tornar redundante em alguns casos particulares.

Já para as *Acceptance Stories*, foi usado o formato que se segue[38]:

```
Dado <contexto>,
Quando <evento>,
<resultado>.
```

Apêndice H

Análise de preços Flying-Sphinx / WebSolr

Para comparar os preços dos planos disponíveis em cada um destes serviços considerou-se um cenário em que: cada documento tem 2500 caracteres indexados; um caracter ocupa 1 byte; 1MB tem portanto cerca de 420 documentos. Admite-se também que o índice resultante de um documento ocupa o mesmo que o documento original sem compressão.

As tabelas H.1 e H.2 apresentam a progressão de preços dos dois serviços.

	Plano 1	**Plano 2**	**Plano 3**
Preço (US$/mês)	12	55	150
Armazenamento (MB)	20	500	5120
US$/doc	0.00143	0.00026	0.00007

Tabela H.1: Progressão de preços do Flying Sphinx. O preço é dado em US$/mês e é relacionado com o tamanho em megabytes do índice.

	Plano 1	**Plano 2**	**Plano 3**	**Plano 4**	**Plano 5**	**Plano 6**
Preço	20	50	100	200	400	800
Num. docs	250000	500000	2000000	5000000	10000000	25000000
US$/doc	0.00008	0.00010	0.00005	0.00004	0.00004	0.00003

Tabela H.2: Progressão de preços do WebSolr. O preço é dado em US$/mês e é relacionado com o número de documentos indexados.

As progressões mostram que para documentos com este número de caracteres o WebSolr é o serviço mais barato. De facto, segundo esta configuração, o Flying Sphinx só se torna mais barato (e apenas nos planos 2 e 3) se considerarmos que cada documento tem menos de 500 caracteres.

Tendo em conta que o número de documentos a indexar no Bundlr é cerca de 120000 (105000 clips mais 15000 utilizadores), o WebSolr é a melhor opção.

Apêndice I

Análise: Métodos de *Tag Extraction*

Na maioria dos sistemas de recomendação baseados em conteúdo (*content-based*), os atributos que descrevem os items a comparar são características textuais extraídas de páginas web, emails, artigos ou descrições de produtos[19]. Estes conteúdos são compostos por texto em linguagem natural, o que torna a sua análise por processos automáticos bastante mais complexa devido à sua ambiguidade.

O problema da extracção de *keywords* a partir de um texto tem várias aplicações além deste tipo de sistemas de recomendação. Sistemas de análise de documentos, *Information Retrieval*, indexação, criação de sumários e sugestão de *keywords* para validação humana são apenas alguns exemplos.

[39] apresenta alguns métodos de extracção de informação para geração de sumários de texto. Estes métodos são também relevantes no contexto desta análise já que a extracção de *keywords* é uma simplificação do problema de identificação dos tópicos do texto, um dos métodos mais simples de sumarização.

Este apêndice faz uma breve análise dos métodos de extracção de informação listados por [39], sob o ponto de vista da extracção de *keywords*.

I.1 *Position Methods*

Estes métodos baseiam-se na ideia que a posição das palavras no texto pode ser um indicador da sua relevância. Em geral, as partes do texto que são consideradas mais relevantes são o título, o início e o fim.

No entanto, este pressuposto nem sempre se verifica, dependendo do tipo de texto que se está a analisar. O método *Optimal Position Policy*[39] identi-

fica automaticamente as posições do texto que são mais propícias a conterem termos discriminadores. Este necessita de ser treinado com textos do tipo dos que se pretendem analisar, para determinar quais as melhores posições.

Um algoritmo de extracção de *keywords* com base na sua posição poderia, então, determinar as melhores frases com a ajuda do método *Optimal Position Policy* e partir dessa informação para fazer uma análise de frequência num espaço mais restrito, por exemplo.

Note-se que, no contexto da análise de páginas web, o título das páginas, tal como definido no cabeçalho do ficheiro HTML, através da tag <*title*>, não é sempre igual ao título do artigo. Um exemplo são as páginas da Wikipedia, cujo título contém sempre "Wikipedia, the free encyclopedia", para além do título do artigo. Por essa razão deve-se ter algum cuidado ao usar o título da página, usando o título do artigo sempre que disponível.

I.2 *Cue Phrase Indicator Criteria*

As soluções que usam este método procuram a ocorrência de certas expressões no texto que são geralmente indicadoras da importância das palavras que as sucedem. Expressões como "em resumo", "concluindo" ou "o melhor" podem indicar a presença de conteúdo importante na frase. Da mesma forma, palavras como "impossível" podem indicar frases menos relevantes.

Tal como os métodos baseados na posição dos termos, as *cue phrases* são dependentes do tipo de texto que se analisa e, claro, da língua em que está escrito. Isto torna difícil a tarefa de determinar quais as frases que indicam conteúdo importante e quais as que apontam para conteúdo não relevante.

Assumindo que se encontravam essas frases, uma solução que aplicasse este método poderia então determinar quais as frases mais relevantes e extrair os seus conteúdos.

I.3 *Frequency Criteria*

Esta abordagem assume que a frequência da utilização de uma palavra no texto é indicador da sua importância. Ao contrário do que seria de esperar, as palavras mais utilizadas nem sempre são as melhores para descrever um texto[39]. Palavras como "a", "mas" e "que" podem facilmente ter uma contagem alta no texto, mas não têm qualquer relevância para o descrever.

Para melhorar os resultados deste método é comum a utilização de técnicas como a remoção de *stopwords*, que remove palavras como as listadas acima, e *stemming*, que reduz a palavra à sua raiz ("certamente" para "cert",

por exemplo).

As soluções que utilizam o método de análise de frequência dividem-se em dois grupos: *Corpus Oriented* e *Document Oriented*[40].

I.3.1 *Corpus Oriented*

Este grupo de soluções utiliza métodos estatísticos de análise de frequência num conjunto de textos e não só no texto que está a ser alvo de extracção de *keywords*.

O método mais comum é a utilização da medida $tf \cdot idf$ (*Term Frequency - Inverse Document Frequency*[19, 39]) que dá mais peso a *keywords* que aparecem frequentemente no documento mas não nos restantes documentos do conjunto de textos. Esta medida tem a vantagem de penalizar os termos que não são tão relevantes por fazerem parte de grande parte dos documentos, como por exemplo algumas *stopwords*.

Existem algumas implementações deste tipo de análise de frequência implementadas em Ruby e disponíveis como código livre, como o TextAnalyzer[1] e o Treat[2], uma *framework* de processamento de linguagem natural.

I.3.2 *Document Oriented*

Contrariamente às *corpus oriented*, estas soluções só utilizam o documento que se está a analisar para fazerem a extracção de *keywords*.

Esta opção é bastante interessante em contextos em que não há um conjunto de textos com que se possa trabalhar, em que esse conjunto esteja constantemente em alteração ou em que os textos do conjunto sejam demasiado diferentes (várias linguagens, por exemplo). Outra vantagem é que, por ser uma análise independente do contexto global dos documentos, escala bastante melhor para grandes conjuntos de documentos.

Técnicas como a análise de co-ocorrências (número de vezes que duas palavras aparecem juntas ou na mesma frase) ajudam a melhorar o resultado base da contagem de termos[41].

A biblioteca Pismo, abordada neste estágio, utiliza um método *document oriented* baseado na contagem dos termos, dando também relevância a termos presentes no título do artigo. Além disso, suporta remoção de *stopwords* da língua inglesa.

Dos métodos apresentados nesta análise este é o único que é independente

[1] http://martin.ankerl.com/textanalyzer
[2] https://github.com/louismullie/treat

tanto da linguagem de escrita como do tipo de texto. A utilização de técnicas de remoção de *stopwords* e *stemming*, é, por sua vez, dependente da linguagem.

I.4 *Connectedness Criteria*

Este método sugere o mapeamento do texto num grafo em que as palavras são os vértices e as arestas representam ligações gramaticais, lexicais, semânticas, de adjacência ou de co-referência[42].

A selecção das melhores *keywords* é depois feita através de métricas simples como o número de repetições de vértices (tendo o mesmo efeito que uma análise de frequência) ou de técnicas mais avançadas, como o *Spreading*, que analisa as ligações dos termos mais frequentes. Outra métrica é o *Local Weighting*, que dá mais peso aos vértices com maior número de arestas.

Este método abre a porta a uma série de opções relacionadas com análise linguística, como a utilização do método *Parts-Of-Speech* que classifica as palavras do texto segundo a sua classe gramatical. Esta classificação pode ser depois usada para excluir palavras que não são tão relevantes para a extracção de *keywords*, como os adjectivos, por exemplo.

Serviços como o AlchemyAPI disponibilizam também outros métodos baseados em análises linguísticas, como a extracção de conceitos[43] que pode também ser relevante no contexto de extracção de *keywords*.

Existem também várias soluções como o OpenNLP ou o NLTK, *frameworks* de processamento de linguagem natural, que se dedicam a este tipo de análise linguística, proporcionando uma série de métodos que podem ser interessantes neste contexto.

I.5 Abordagens Híbridas

Também são comuns abordagens que juntam vários conceitos apresentados nos métodos anteriores na sua análise.

O Phrasie, tal como o Topia.TermExtract, em que se baseia, são um exemplo dessas soluções, utilizando métodos de análise linguística como o *Parts-Of-Speech* (extraindo apenas substantivos) em conjunto com alguns métodos simples de análise de frequência.

Apêndice J

Análise: Métodos de *Boilerplate Removal*

Boilerplate Removal (também chamado *Text Extraction from HTML*, entre outras variações), são técnicas de extracção de texto a partir das páginas HTML que o contém, descartando todo o aparato da estrutura do website, como os menus, barras laterais, secções de comentários, publicidade, entre outros[44, 25].

Esta tarefa é importante para vários domínios, como *Information Retrieval* (melhorando os resultados obtidos), *Link Analysis* (que podia considerar os links das secções de publicidade, por exemplo) ou tarefas de adaptação do conteúdo web como a geração automática de *RSS feeds* através das páginas de artigos[45].

Recentemente têm surgido outro tipo de aplicações destes algoritmos, como *scripts* que com um clique "limpam" o conteúdo do website e mostram apenas o texto principal (Readability, Apple Safari) ou aplicações para dispositivos móveis que permitem guardar o texto de artigos para ler mais tarde (Instapaper, Pocket, Readability).

Este apêndice faz uma breve análise dos tipos de estratégias mais utilizadas e apresenta 3 soluções *open-source*, descrevendo sucintamente o seu funcionamento.

J.1 Estratégias mais utilizadas

A vasta diversidade da Internet e da estrutura dos websites faz com que a tarefa de remoção do *boilerplate* não seja trivial. Não existem, até à data, um conjunto de regras fixas que permita identificar as secções relevantes do HTML[44].

Para extrair informação do HTML são observadas algumas características que representam de alguma forma o seu conteúdo e organização. A maior parte dos algoritmos usa uma combinação dessas características para chegar a conclusões, seja ao treinar um classificador ou simplesmente ao comparar as características com algum valor base.

A descrição das estratégias mais utilizadas será feita com base em 4 dos tipos de características mais comuns[44], que serão apresentados de seguida.

J.1.1 *Site-specific Features*

Estas são características específicas a um grupo específico de websites. Por exemplo, um dos métodos constrói a página HTML incluindo os seus ficheiros de estilo e imagens para ter uma representação visual semelhante à que o utilizador tem quando abre a página no seu browser. Comparando essa representação com outras páginas do mesmo website é possível distinguir o template do texto e fazer a extracção.

Outro método faz uma análise do HTML de várias páginas do website (sem representação visual) tentando descobrir padrões que identifiquem a estrutura da página.

A necessidade de ter um conjunto de páginas do mesmo website, a lentidão dos processos de representação visual e o perigo de *over-fitting* tornam estes métodos pouco apelativos para uso em conjuntos de páginas genéricas.

J.1.2 *Structural Features*

Estas são as características estruturais do HTML, como as tags HTML e as suas classes e IDs.

A maior parte dos métodos de remoção de *boilerplate* fazem algum tipo de filtragem das tags HTML. Tags como FORM ou INPUT são indicativas de *boilerplate*, enquanto que tags como H1, P, DIV ou a recente ARTICLE indicam a possível presença de conteúdo.

J.1.3 *Shallow Features*

Estas características, apresentadas pelo criador da biblioteca Boilerpipe[44] e baseadas em métodos de *Quantitative Linguistics* fazem uma análise de alto nível e independente do domínio e da linguagem do documento. A sua análise é feita a nível de blocos de texto não separado por tags HTML (excluindo a tag A - *anchor*).

Algumas destas características são o tamanho médio das palavras e das frases e o número absoluto de palavras.

Estas têm a vantagem de serem muito simples de calcular e podem ser combinadas com outras como densidade de links (descrita de seguida) para ter resultados bastante interessantes e com baixo custo computacional e de implementação.

J.1.4 *Heuristic Features*

Estas são características que não têm uma fundamentação teórica mas que aparentam dar bons resultados.

Exemplos são o número de palavras ou caracteres em cada bloco de conteúdo, a quantidade de vírgulas, o número de palavras capitalizadas ou a densidade de links. Esta última é representada pelo número de palavras dentro de uma tag A a dividir pelo número total de palavras no bloco.

J.2 Análise de soluções *open-source*

J.2.1 Boilerpipe

Este projecto consiste numa família de algoritmos de *boilerplate removal* baseados em árvores de decisão. Os algoritmos foram pensados para extrair textos de páginas de notícias e artigos, mas também são aplicáveis a páginas genéricas.

O projecto surgiu das contribuições do autor em [44] e contém, portanto, a estratégia que combina o número de palavras com a densidade de links na sua lista de algoritmos.

Internamente o sistema é composto por 3 partes essenciais, descritas de seguida:

HTML Parser Este componente processa o HTML e transforma-o em vários blocos de texto, ignorando simultaneamente tags HTML como SCRIPT, OPTION, STYLE, entre outros. Os blocos de texto são formados usando tags HTML como limite, excepto a tag A que é mantida em junção com o texto. São também guardadas algumas métricas como o número de palavras no bloco e dentro das tags A.

Filters Estes são aplicados ao resultado da fase anterior e marcam os vários blocos como *content* ou *boilerplate*. Estes filtros estão divididos em filtros simples, filtros que se adequam principalmente à língua inglesa e filtros heurísticos.

Extractors Estes componentes usam os resultados da aplicação de filtros, limpam o texto e devolvem o resultado final.

J.2.2 Readability

O Readability é um dos algoritmos mais conhecidos do utilizador comum da Internet, principalmente por ter sido disponibilizado sob a forma de um *bookmarklet* que fazia uma limpeza do conteúdo da página que o utilizador estava a ver, mostrando apenas o texto principal, com uma apresentação limpa e fácil de ler.

A versão inicial deste algoritmo é em Javascript, mas existem algumas versões noutras linguagens de programação, como Ruby ou Python.

Recentemente o projecto *open-source* foi descontinuado, dando lugar a um serviço com o mesmo nome, que permite guardar artigos para ler mais tarde. A última versão disponível do código original em Javascript é a versão 1.7.1.

O Readability é interessante pela quantidade de características heurísticas que usa. O algoritmo começa por definir uma série de expressões regulares que marcam nomes de classes CSS como "comment", "menu", "footer" e mesmo classes usadas por alguns websites mais conhecidos, como propícias a ter conteúdo ou não.

Depois disso, é feita a extracção do título do artigo. Nesta fase são feitos uma série de testes a características como o tamanho do conteúdo da tag TITLE e à presença de caracteres como "|" que muitas vezes separam o domínio do website do título real da página. Se depois da remoção dessas partes o algoritmo não estiver satisfeito com o resultado (medido através do tamanho do título extraído), procura tags H1.

Depois disso é feita uma travessia dos blocos de HTML numa estrutura em árvore e são calculadas pontuações para cada um, com base na tag HTML, no número de vírgulas que contêm, nos nomes das classes CSS, na densidade de links e no número de caracteres de texto. Nós com tags proibidas (tal como definido nas expressões regulares) ou com conteúdo suspeito são removidos. Para cada um dos restantes nós é somada a sua pontuação à pontuação dos seu nó pai e somada metade à do seu nó avô. Finalmente é atribuído um bónus aos nós com baixa densidade de links.

Percorrida a árvore, selecciona-se o melhor nó e faz-se uma análise dos seus nós irmãos, na expectativa de encontrar outras partes do texto separadas por conteúdo não relevante, que é depois agrupado com o conteúdo do melhor nó.

J.2.3 Pismo

O Pismo é uma biblioteca em Ruby criada por Peter Cooper e usada no Coder.io[1], um agregador de links para programadores. Dos 3 algoritmos apresentados, este é o menos conhecido.

Tal como no caso do Readability, o sistema base que o Pismo usa faz uma representação em árvore da estrutura da página. O funcionamento do algoritmo também é semelhante ao do algoritmo anterior, definindo conjuntos de nomes de classes CSS e tags HTML que considera ter ou não ter conteúdo, respectivamente.

O sistema ordena os nós com base numa pontuação baseada em características como a percentagem de palavras com mais de 3 letras no texto do nó, o número de palavras dos nós filhos e a quantidade de nós filhos sem conteúdo.

Como resultado, é escolhido o nó com maior pontuação.

A escolha do título do texto tem uma abordagem diferente do Readability, analisando o texto através de expressões regulares (muitas delas de websites específicos) e apenas recorrendo à tag TITLE em último recurso.

[1] http://coder.io

Apêndice K

Análise e Comparação de soluções de *Tag Extraction*

Este estudo pretende comparar algumas soluções para o problema de *Tag Extraction*. Ambiciona-se chegar a uma aproximação daquela que é, de facto, a melhor solução no contexto dos sistemas de recomendação.

Esta análise divide-se em duas experiências que representam dois tipos de temas diferentes: um mais específico - "Al-Qaeda" - e outro mais genérico - "Social Media".

Cada uma destas experiências é composta por 60 artigos web - 30 relacionados com o tópico e 30 aleatórios. Estes artigos são escolhidos em igual proporção de 5 websites diferentes, que serão especificados nas secções das experiências (K.1 e K.2).

Os algoritmos comparados são apresentados de seguida:

Pismo A biblioteca Pismo com *stemming* de palavras com mais de 3 letras (que é o que é usado no resto dos testes onde esta biblioteca é usada, excepto no PismoSemStemming).

PismoSemStemming A biblioteca Pismo configurada para não fazer *stemming* aos termos. Pretende-se analisar o efeito do *stemming* ao comparar com o algoritmo anterior.

PismoR Biblioteca Pismo mas utilizando o algoritmo de remoção de *boilerplate* Readability em vez do algoritmo que a própria biblioteca contém.

PismoRB O mesmo que o algoritmo anterior, mas agora com a versão do Readability alterada pela equipa do Bundlr.

Phrasie A biblioteca Phrasie. Como esta biblioteca não incorpora nenhuma solução de remoção de *boilerplate*, utiliza-se essa funcionalidade da biblioteca Pismo.

PhrasieR Semelhante ao anterior, mas usando agora o Readability como método de remoção do código *boilerplate*.

PhrasieRB O mesmo que o algoritmo anterior, mas agora com a versão do Readability alterada pela equipa do Bundlr.

PismoPhrasie Esta solução junta aos resultados da biblioteca Pismo, os resultados das biblioteca Phrasie que sejam compostos por dois ou mais termos. Tenta-se corrigir o facto do Pismo não considerar *keywords* com mais que uma palavra.

PismoPhrasieR Semelhante ao anterior, mas usando agora o Readability em vez do algoritmo do Pismo para remoção de *boilerplate*.

PismoPhrasieRB O mesmo que o algoritmo anterior, mas agora com a versão do Readability alterada pela equipa do Bundlr.

MetaKeywords Este método foi incluído para testar a qualidade das sugestões quando são consideradas as *Meta Keywords*. Note-se que os artigos web escolhidos para este estudo utilizam todos *Meta Keywords* correctamente, pelo que os resultados desta análise não representam a realidade dos websites genéricos, muitos dos quais não utilizam estas anotações.

Foram também utilizadas diferentes métricas de semelhança para as comparações entre as tags extraídas de cada website:

Jaccard Coefficient Aplicação do coeficiente de Jaccard.

Dice's Coefficient Aplicação do coeficiente de Dice.

Overlap Coefficient Aplicação do *Overlap Coefficient*.

CommonTagsPercentageRepeat Implementação do algoritmo descrito como "Percentagem de tags em comum incluindo repetições" na secção 5.5.4 deste relatório. Abreviado nas tabelas de resultados como CTPR.

WeightedCoefficient Implementação do algoritmo descrito como "Soma ponderada para tags multi-palavra" na secção 5.5.4 deste relatório.

As combinações algoritmo / métrica de semelhança são analisados com base na qualidade das sugestões que fazem. Para analisar essa qualidade, calculam-se as recomendações para os 30 links relacionados com o tópico de cada experiência. Uma combinação óptima recomendaria todos os outros artigos sobre o tema da experiência, não incluindo os artigos de tema aleatório.

Desta forma, a análise do desempenho das soluções começa pela contagem de *true positives*, *true negatives*, *false positives* e *false negatives*, que são depois utilizados para calcular medidas de *precision*, *sensitivity*, *accuracy* e *specificity*. Finalmente, é calculada a *F-measure* dessa solução, que é usada como medida de qualidade global da solução.

Nas próximas duas secções apresentam-se os resultados do estudo para cada uma das experiências. A secção K.3 deste capítulo apresenta as conclusões do estudo.

K.1 Tema: Al-Qaeda

Esta experiência analisa o desempenho das soluções para um tema específico: "Al-Qaeda".

Os 60 links utilizados foram recolhidos dos websites do The Guardian[1], New York Times[2], Reuters[3], Telegraph[4] e National Geographic[5]. Estes têm a particularidade de serem websites de notícias generalistas, cobrindo um largo número de tópicos. A grande maioria dos links recolhidos são, então, artigos de notícias sobre este tópico.

Para facilitar a apresentação de resultados, listam-se na tabela K.1 apenas os cálculos da *F-measure* para cada combinação desta experiência, já que engloba os resultados da *precision* e da *sensitivity*. Os resultados das restantes medidas são listados em detalhe no Anexo externo P.

Ao analisar os resultados da tabela, é visível que não há muita variação entre os diferentes tipos de medidas de semelhança. Entre os coeficientes de *Jaccard*, *Dice* e *Overlap* existem variações das pontuações atribuídas a cada recomendação (não visíveis na tabela) e possivelmente na ordem das sugestões, mas não no desempenho global da solução, sendo sempre recomendados os mesmos links. Como este estudo não analisa a ordem das recomendações, estes resultados apontam para uma equivalência entre estas três métricas de semelhança.

[1] http://guardian.co.uk
[2] http://nytimes.com
[3] http://reuters.com
[4] http://telegraph.co.uk
[5] http://nationalgeographic.com

Algoritmo/Medida	**Jaccard**	**Dice**	**Overlap**	**CTPR**	**Weighted**
Pismo	63.56%	63.56%	63.56%	63.56%	63.56%
PismoSemStem	62.46%	62.46%	62.46%	62.46%	62.46%
PismoR	69.68%	69.68%	69.68%	69.68%	69.68%
PismoRB	69.68%	69.68%	69.68%	69.68%	69.68%
Phrasie	63.53%	63.53%	63.53%	63.53%	63.64%
PhrasieR	58.04%	58.04%	58.04%	58.04%	57.82%
PhrasieRB	58.04%	58.04%	58.04%	58.04%	57.82%
PismoPhrasie	62.06%	62.06%	62.06%	62.06%	65.02%
PismoPhrasieR	69.68%	69.68%	69.68%	69.68%	69.53%
PismoPhrasieRB	69.68%	69.68%	69.68%	69.68%	69.53%
MetaKeywords	36.56%	36.56%	36.56%	36.56%	51.77%

Tabela K.1: Medidas *F-Measure* para todas as medidas de semelhança na experiência "Al-Qaeda".

A métrica *CommonTagsPercentageRepeat* (CTPR) também apresenta sempre resultados iguais às das três primeiras métricas. Estes resultados justificam-se porque os conjuntos de tags extraídas em cada link nunca contêm tags repetidas, como acontece quando se compara as tags dos bundles, por exemplo, que são um aglomerado de tags dos clips, dando lugar a repetições. Embora este estudo não se adeque à análise desta métrica, considera-se que é, de facto, uma boa evolução do coeficiente de *Jaccard*, sendo usada para representar os resultados das 4 primeiras métricas nas restantes tabelas deste apêndice.

Finalmente, a métrica "Soma ponderada para tags multi-palavra", aqui chamada *Weighted*, tem resultados um pouco melhores que as restantes, quando são utilizados algoritmos que suportam *keywords* com várias palavras.

A tabela K.2 apresenta uma simplificação da tabela K.1, considerando agora apenas duas métricas e ordenando os algoritmos por ordem decrescente de *F-measure* para a métrica de semelhança *CommonTagsPercentageRepeat*.

Esta tabela destaca a qualidade dos vários algoritmos testados, tal como medida nesta experiência. Podem-se observar alguns padrões, como o facto de algoritmos que usam o Readability em vez do algoritmo do Pismo para remoção de *boilerplate* parecerem ser melhores que os que o usam (PismoR, PismoRB, PismoPhrasieR, PismoPhrasieRB), excepto quando usam o Phrasie (PhrasieR, PhrasieRB). Observa-se também que a não utilização de *stemming* na biblioteca Pismo parece prejudicar um pouco a qualidade das sugestões e que as alterações feitas pela equipa do Bundlr ao algoritmo Readability parecem manter o desempenho do algoritmo original (visível em

Algoritmo/Medida	**CTPR**	**Weighted**
PismoR	69.68%	69.68%
PismoRB	69.68%	69.68%
PismoPhrasieR	69.68%	69.53%
PismoPhrasieRB	69.68%	69.53%
Pismo	63.56%	63.56%
Phrasie	63.53%	63.64%
PismoSemStem	62.46%	62.46%
PismoPhrasie	62.06%	65.02%
PhrasieR	58.04%	57.82%
PhrasieRB	58.04%	57.82%
MetaKeywords	36.56%	51.77%

Tabela K.2: Ranking de *F-Measure* na experiência "Al-Qaeda".

todos os pares (...)R, (...)RB).

K.2 Tema: Social Media

Esta experiência analisa o desempenho das soluções para um tema mais genérico que o anterior: "Social Media".

Os 60 links utilizados foram recolhidos dos websites Mashable[6], Gizmodo[7], LifeHacker[8], BusinessInsider[9] e CNET News[10]. Estes websites abordam principalmente temas de tecnologia. Tal como no estudo anterior, a grande maioria dos links recolhidos são artigos de notícias sobre o tópico de "Social Media".

Listam-se na tabela K.3 de seguida os cálculos da *F-measure* para cada combinação desta experiência. Os resultados das restantes medidas são listados em detalhe no Anexo externo P.

Os resultados dessa tabela mostram novamente a equivalência das quatro primeiras métricas no contexto desta análise. Identificam-se novamente algumas diferenças para a métrica "Soma ponderada para tags multi-palavra" (*Weighted*), que têm um maior impacto (cerca de 14% de diferença) no algoritmo das *Meta Keywords*. Esta melhoria está relacionada com o tipo de tags que são usadas nas *Meta Keywords*, normalmente multi-palavras.

[6] http://mashable.com
[7] http://gizmodo.com
[8] http://lifehacker.com
[9] http://businessinsider.com
[10] http://news.cnet.com

Algoritmo/Medida	**Jaccard**	**Dice**	**Overlap**	**CTPR**	**Weighted**
Pismo	91.46%	91.46%	91.46%	91.46%	91.46%
PismoSemStem	91.38%	91.38%	91.38%	91.38%	91.38%
PismoR	93.17%	93.17%	93.17%	93.17%	93.17%
PismoRB	93.17%	93.17%	93.17%	93.17%	93.17%
Phrasie	74.34%	74.34%	74.34%	74.34%	74.08%
PhrasieR	45.28%	45.28%	45.28%	45.28%	45.50%
PhrasieRB	45.28%	45.28%	45.28%	45.28%	45.50%
PismoPhrasie	93.78%	93.78%	93.78%	93.78%	93.67%
PismoPhrasieR	95.32%	95.32%	95.32%	95.32%	95.20%
PismoPhrasieRB	95.32%	95.32%	95.32%	95.32%	95.20%
MetaKeywords	59.20%	59.20%	59.20%	59.20%	73.17%

Tabela K.3: Medidas *F-Measure* para todas as medidas de semelhança na experiência "Social Media".

A tabela K.4 apresenta uma simplificação da tabela K.3, considerando agora apenas duas métricas e ordenando os algoritmos por ordem decrescente de *F-measure* para a métrica de semelhança *CommonTagsPercentageRepeat*.

Algoritmo/Medida	**CTPR**	**Weighted**
PismoPhrasieR	95.32%	95.20%
PismoPhrasieRB	95.32%	95.20%
PismoPhrasie	93.78%	93.67%
PismoR	93.17%	93.17%
PismoRB	93.17%	93.17%
Pismo	91.46%	91.46%
PismoSemStem	91.38%	91.38%
Phrasie	74.34%	74.08%
MetaKeywords	59.20%	73.17%
PhrasieR	45.28%	45.50%
PhrasieRB	45.28%	45.50%

Tabela K.4: Ranking de *F-Measure* na experiência "Social Media".

Os resultados da tabela mostram uma melhoria significativa em relação aos resultados do estudo anterior, o que pode apontar para uma melhor adaptação destes algoritmos a temas mais genéricos ou ao estilo de escrita e organização dos websites desta experiência.

No topo da tabela encontram-se novamente os algoritmos PismoPhrasieR e PismoPhrasieRB, com *F-measure* de 95.32%. Observa-se também que os

algoritmos que usam a biblioteca Phrasie para extrair todas as tags tiveram resultados bastante piores (Phrasie, PhrasieR, PhrasieRB) que os que utilizam o Pismo como base.

Tal como na experiência "Al-Qaeda", observa-se novamente uma equivalência da qualidade das alterações da equipa do Bundlr no algoritmo Readability com o algoritmo original.

As conclusões destas duas experiências são apresentadas na secção que se segue.

K.3 Conclusões

Este estudo pretendia fazer uma análise da qualidade de soluções para o problema de *Tag Retrieval*. Foram consideradas duas experiências com os tópicos "Al-Qaeda" e "Social Media", que apontaram as soluções PismoPhrasieR e PismoPhrasieRB como as que fazem recomendações de maior qualidade.

Concluiu-se também que, para as experiências consideradas, as métricas de semelhança *Jaccard*, *Dice*, *Overlap* e *CommonTagsPercentageRepeat* não apresentam, entre elas, diferenças de qualidade dos algoritmos testados. Já a métrica *WeightedCoefficient* leva a melhores resultados no algoritmo *MetaKeywords*.

Devido a limitações temporais, optou-se por aumentar o número de soluções e métricas testadas, em detrimento do número de experiências corridas. Embora cada experiência seja composta por 60 links, as conclusões a que este estudo chega não são estatisticamente representativas de toda a gama de websites que poderão ser guardados no Bundlr. Além da criação de mais experiências, a utilização de um *dataset* que agrupasse links pelo seu tema específico também poderia tornar este estudo mais significativo.

Mesmo assim, as conclusões a que se chegaram serão consideradas para o problema de encontrar uma boa solução de *Tag Extraction*, sob pena de não funcionarem de forma óptima para outros tipos de websites.

Desta forma, as soluções escolhidas para a implementação do Módulo 5 (secção 5.5) foram a PismoPhrasieRB, já que implementa as alterações feitas pela equipa do Bundlr evitando a manutenção de dois métodos de extracção, e a medida de semelhança "Percentagem de tags em comum incluindo repetições" (*CommonTagsPercentageRepeat*).

A escolha desta medida de semelhança justifica-se por ser mais adequada ao tipo de tags extraídas pelo PismoPhrasieRB - tags com poucas ou mesmo uma única palavra. Comparativamente às outras medidas de semelhança, esta também se torna bastante interessante por ter em conta tags repetidas, embora esta vantagem não tenha sido analisada neste estudo.

Apêndice L

Listagem do *spec* dos testes unitários

```
Bundle
  on creation
    should create a new instance given valid attributes
    should create a new instance
  private flag
    should respond to private?
  while managing editing permissions
    in public bundles
      should let the author edit it
      should block others from editing it
    in public bundles with collaborators
      should let the author edit if it has any of the Plans
      should let the collaborators edit if it has the Free Plan
      should let the collaborators edit if it has the Team Plan
      should let the collaborators edit if it has the Pro Plan
    in private bundles
      should let the author edit if it has the Pro Plan
      should NOT let the author edit otherwise
      should block others from editing it
    in private bundles with collaboration
      should let the collaborators edit if it has the Pro Plan
      should NOT let the collaborators edit otherwise
  publishing
    should create a clip event if the clip exists
    should not create a clip event if the clip no longer exists
    should set the created_at field with the time when the event is delayed
  followers and followed
    should add the following user to the followed object's followers list
    should be able to accept new followers even if they are not valid
    clear_followers_except_collabs
      should remove everyone except collaborators from the followers list
  destroying
    should remove the respective stream events
    should remove all bundle related events when a user is destroyed
    should decrement the user's bundle counter

Clip
  on creation
    should try to fetch tags
```

```
40    destroying
41      should remove the respective stream events
42      should remove all related clipping events when a bundle is destroyed
43      should decrement the bundle's clip counter
44    search related
45      on creation
46        should create a Delayed Job that auto-indexes the new clip within Solr
47      on destruction
48        should create a Delayed Job that removes the clip from Solr's index
49      updating indexed clip attributes
50        should update the index via a DelayedJob when the title changes
51        should update the index via a DelayedJob when the description changes
52        should update the index via a DelayedJob when the selection changes
53        should update the index via a DelayedJob when the content changes
54        should update the index via a DelayedJob when the private changes
55        should update the index via a DelayedJob when the updated_at changes
56
57  Collaborator
58    accepted flag
59      should respond to accepted?
60
61  StreamEvent
62    adding to receivers
63      should add a new follower to the user's activities' receivers
64      should add a new follower to the bundle's activities' receivers
65      should not repeatedly insert the hash to the followers array
66
67  User
68    Website Validation
69      should create when invalid
70      should not save when updating and invalid
71    FactoryGirl creating pro users
72      should create a user capable of creating private bundles
73    num_bundles
74      should increment the author's num_bundles field
75      should increment the author's num_bundles field
76      should update the author's num_bundles field in real time
77    private bundles
78      should increment the author's num_private field
79      should decrement the author's num_private field after deletion if
            private
80      should NOT change the author's num_private field after creation/deletion
            if public
81      should update the author's num_private field in real time
82    followers and followed
83      should add the following user to the followed object's followers list
84    updating activity stream's cached fields
85      should update the avatar on the stream event once changed
86    setting the last stream event
87      should save the event's created_at value as last_stream_event
88    destroying
89      should remove all related events when a user is destroyed
90    search related
91      on creation
92        should create a Delayed Job that auto-indexes the new user within Solr
93      on destruction
94        should create a Delayed Job that removes the user from Solr's index
95      updating indexed clip attributes
96        should update the index via a DelayedJob when the name changes
97        should update the index via a DelayedJob when the description changes
98        should update the index via a DelayedJob when the username changes
99
```

```
100
101 AccountController
102   subscriptions
103     should only follow once, even if called multiple times
104     should only unfollow once, even if called multiple times
105     should block activity stream spam by skipping the creation of follow
            events after the first follow
106
107 ActivityStreamController
108   #index
109     should display activities
110   keeping track of viewed stream events
111     should know what is latest date as soon as I view the stream
112     should know how many stream events are left to see
113
114 BookmarkletController
115   #index
116     requires url parameter
117     must have a proper URL
118     should work with an unknown URL
119     should work with a Flickr URL
120
121 BundlesController
122   #show
123     should retrieve an existent bundle
124     should show the collaborators modal box for paying users
125     should show collaborators modal box for free users
126   #show with view
127     should present an existent bundle for
128   #update
129     should not allow signed out users to edit a bundle
130     should validate signed in users before editing a bundle
131   making a bundle private
132     should remove everyone except collaborators from the followers list
133   #remove_collaborators
134     should not do anything if the user is not the author
135     should remove all collaborators otherwise
136   subscriptions
137     should only follow once, even if called multiple times
138     should not allow the author to follow its own bundle (already following)
139     should not allow the author to unfollow its own bundle (already
            following)
140     should not allow a random user to follow a private bundle
141     should not allow a logged out user to follow a private bundle
142     should allow a collaborator to follow a private bundle
143     should not allow a random user to unfollow a private bundle
144     should not allow a logged out user to unfollow a private bundle
145     should allow a collaborator to unfollow a private bundle
146     should block activity stream spam by skipping the creation of follow
            events after the first follow
147   covering from malicious use
148     should not allow making a bundle private if you don't have the correct
            plan
149
150 CollaborationRequestsController
151   #accept
152     should automatically follow the bundle when the collaboration request is
             accepted
153
154 CollaboratorsController
155   #create
156     should not allow a free user to add collaborators
```

```
157     should let a paying user add collaborators
158   #destroy
159     should force the ex-collaborator to unfollow the private bundle if he
            follows it
160     should not force the ex-collaborator to unfollow the private bundle if
            he does not follow it
161
162
163 Suggestion::HasSuggestions
164   using the suggestions getters and setters
165     should be able to assign the clip's suggestions
166     should be able to assign the bundle's tag array
167     should be able to assign the clip author's tag array
168   saving last_updated timestamps
169     should correctly update the last updated field for clip - bundle
170     should correctly update the last updated field for clip - user
171     should correctly update the last updated field for clip - clip
172     should correctly update the last updated field for bundle - user
173     should correctly update the last updated field for bundle - clip
174     should correctly update the last updated field for bundle - bundle
175     should correctly update the last updated field for user - clip
176     should correctly update the last updated field for user - bundle
177     should correctly update the last updated field for user - user
178
179 Suggestion
180   calculating similarity between objects
181     should return 1 for objects with the same tags
182     should return 0 for objects with no tags
183   finding the N closest matches
184     should return [] if there are no other objects
185     should return ordered results
186     should return a high score for bundles about the same thing
187   finding the N closest matches since the last update
188     should accept a set of initial (previously calculated) suggestions to
            work with
189   calculating suggestions for the whole database
190     should be able to suggest, for every Clip, similar Bundles
191     should be able to suggest, for every Clip, similar Users
192     should be able to suggest, for every Clip, similar Clips
193     should be able to suggest, for every Bundle, similar Clips
194     should be able to suggest, for every User, similar Clips
195
196 Tagger
197   should correctly cleanup http://www2.maxima.xl.pt/Novidades/
          DetalheNovidades/tabid/310/itemId/6665/Default.aspx
198   should correctly cleanup http://www.cmjornal.xl.pt/detalhe/noticias/outros
          /domingo/tinhamos-ordens-para-destruir-tudo
199   should correctly cleanup http://www.record.xl.pt/
200   should correctly cleanup http://www.record.xl.pt/Futebol/Internacional/
          interior.aspx?content_id=755883
201   should correctly cleanup http://mashable.com/2012/05/09/google-oracle-fair
          -use/
202   should correctly cleanup http://global.nytimes.com/
203   should correctly cleanup http://www.nytimes.com/pages/world/europe/index.
          html
204   should correctly cleanup http://thecaucus.blogs.nytimes.com/2012/05/09/
          obama-likely-to-speak-about-same-sex-marriage-in-interview/?ref=global
          -home
205   should correctly cleanup http://lifehacker.com/5908903/why-wont-this-web-
          site-load-correctly-and-how-can-i-fix-it
206   should correctly cleanup http://gizmodo.com/5908931/why-building-the-death
          -star-is-a-bad-investment
```

```
207   should correctly cleanup http://allfacebook.com/military-suicide-
          prevention_b88310
208   should correctly cleanup http://life.time.com/culture/teenage-wasteland-
          japanese-youth-in-revolt-1964/#1
209   should correctly cleanup http://www.fubiz.net/2012/05/02/simon-davidson-
          photography/
210   should correctly cleanup http://www.washingtonpost.com/
211   should correctly cleanup http://www.washingtonpost.com/world/national-
          security/cia-unraveled-bomb-plot-from-within/2012/05/08/
          gIQA5tKOBU_story.html?hpid=z1
212   should correctly cleanup http://www.ted.com/talks/
          jp_rangaswami_information_is_food.html
213   should correctly cleanup http://blog.ted.com/2012/05/06/the-color-of-x-
          seeing-red-in-this-photo-set-from-tedxsummit/
214   on clip creation
215     should propagate the tags upstream
216   cleaning up the tags after retrieval
217     should apply the defined rules of trimming and cleanup
218     should remove the default keywords
219     should detect wether or not these are the default tags
220
221 Tagger::TagList
222   using the tags getter and setter
223     should be able to assign the clip's tag array
224     should be able to assign the bundle's tag array
225     should be able to assign the clip author's tag array
226     should be able to assign the bundle's owner tag array
227     should be able to read the tag array in the clip model
228     should be able to read the tag array in the bundle model
229     should be able to read the tag array in the user model
230     should be able to append to the clip's tag array
231     should be able to append to the bundle's tag array
232     should be able to append to the clip author's tag array
233     should be able to append to the bundle's owner tag array
234     should be able to sum and append another tag array to the clip
235     should be able to sum and append another tag array to the bundle
236     should be able to sum and append another tag array to the clip author
237     should be able to sum and append another tag array to the bundle's owner
238     should auto-save when on the clip
239     should auto-save when on the bundle
240     should auto-save when on the clip author
241     should auto-save when on the bundle's owner
242     should not create a TagList entry if tags is nil
243     should create a TagList entry otherwise
```