Iuri de Vilhena Gonzalez Rodrigues

Licenciado em Engenharia Electrotécnica e de Computadores

Desenvolvimento de um motor de busca e comparação na Web

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletrotécnica e de Computadores

Orientador: João Paulo Pimentão, Professor Auxiliar, FCT - UNL

Júri:

Presidente: Prof. Doutor, Tiago Oliveira Cardozo, FCT - UNL

Arguentes: Prof. Doutor, João António Barata de Oliveira, FCT - UNL

Vogais: Prof. Doutor, João Paulo Pimentão, FCT - UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

23 de Setembro, 2016

Resumo

Com o aumento da competição entre empresas causada pela globalização dos mercados, e devido à crescente obrigação para os clientes encontrarem o negócio mais vantajoso por motivos económicos, é necessária uma pesquisa extensa por parte do cliente de forma a encontrar o melhor bem/serviço em termos de preço/qualidade.

Por outro lado também podemos observar um aumento na popularidade de websites com o intuito de dar a conhecer ao utilizador o negócio mais vantajoso, com comparações entre produtos de variados websites que proporcionam essa informação voluntariamente. O problema emerge quando nem todas as empresas têm interesses em ter os seus produtos comparados, conduzindo à criação de listas defeituosas.

Esta dissertação propõe a criação de uma aplicação web para Android capaz de suportar os seus utilizadores na escolha dos melhores produtos disponíveis através da Internet. Para o desenvolvimento da aplicação é proposto a criação de uma base de dados capaz de indexar todo o tipo de produtos listados, e de um WebCrawler que correrá periodicamente através dos sites escolhidos, retornando o seu código Html que será processado por um Parser (capaz de organizar o código recebido numa estrutura de dados legível) para encontrar qualquer informação relevante para a base de dados.

Palavras-chave: Android, Webcrawler, Ontologia, Html, Parser, Aplicação, Webservices, Base de Dados, MySQL;

Abstract

With increasing competition between firms brought about by market globalization, and because there is a growing need for customers to find the most advantageous deal, extensive research on the part of the customer to find the best good/service in terms of price/quality is required.

On the other hand we can also observe an increase in popularity of websites dedicated to make known to you, the user, the most advantageous deal, with comparisons between various products from diferente websites, that provide this information voluntarily. The problem arises when not all companies have interests in having their products compared, leading for creation of faulty lists.

This paper proposes the creation of a web application for Android able to support their users in finding the best products available via the Internet. For the development og the application, it is proposed the creation of a database able to index all kinds of listed products, and a WebCrawler that will run periodically through the chosen sites, returning the Html code that will be processed by a Parser (able to organize code received into a readable data structure) to find any relevant information to the database.

Keywords: Android, Webcrawler, Ontologia, Html, Parser, Aplicação, webservices, Data Base, MySQL;

Índice

Lista de Figuras

1

1. Introdução

1.1. Enquadramento

De acordo com um estudo feito pela empresa Email Brokers, em Portugal cerca de 90% do comércio pela internet (eCommerce) não é aproveitado por empresas portuguesas, isto é, apenas 10% das compras pela internet feita por portugueses é realmente direcionada para empresas em Portugal, sendo o resto para empresas situadas no estrangeiro [1].

Esta disparidade entre o comércio eletrónico português e o eCommerce estrangeiro, deve-se à falta de credibilidade dos websites que oferecem este serviço. Falta de informações de contacto, como número de telefone, fax ou mail, preços que muitas vezes estão desatualizados ou nem estão presentes, ou até falta da informação geográfica referente á localização da loja [1, 2]. Obviamente isto é um problema que pode ser apenas solucionados pelos donos de cada website, e sem esta confiança necessária para realizar negócios, os portugueses preferem fazer compras em sites estrangeiros ou perto de suas casas sem tomar proveito nas vantagens que eCommerce pode trazer.

Para além disto, apenas cerca de 6% de micro empresas (representando cerca de 95% do mercado ecommerce) e 35% das grandes empresas utilizam serviços de e-commerce [3]. Outro problema, é a falta de manutenção que muitos destes websites oferecem, levando em mais de metade dos sites portugueses mais de 1 ano a atuali-

zarem o seu conteúdo, e apenas menos de 5% possuem atualizações relativamente regulares [1].

As vantagens provenientes do eCommerce podem ser sumarizadas por aumento da competitividade do mercado, estando todas as empresas ao alcance do utilizador, isto permite a escolha de produtos tendo em conta todos os produtos "rivais" que possam ter melhor qualidade ou preço, e também pela comodidade visto que a grande maioria de empresas presentes em eCommerce possuem um serviço de entrega ao domicílio, removendo qualquer vantagem que uma empresa com um produto de gama inferior possa ter apenas devido à sua proximidade ao cliente.

Infelizmente devido ao leque de escolhas disponível ao utilizador, este pode não conseguir decidir qual o melhor produto ou poderá fazer a escolha de produtos menos benéficos. Esta falta de filtragem sobre o conteúdo fornecido ao utilizador é prejudicial à experiencia do cliente ao utilizar a internet para comércio e aumentando ainda mais a falta de confiança no eCommerce.

1.2. Problema e Motivação

O eCommerce em Portugal possui um grave problema de falta de confiança entre o utilizador e o serviço em si que o possibilita [2], visto que grande parte dos problemas derivados dessa falta de confiança só poderão ser resolvidos do lado das empresas, propusemos-mos a desenvolver uma aplicação capaz de analisar vários sites de empresas diferentes, categorizar os seus produtos, e dar ao utilizador os resultados dessa pesquisa filtrada com as melhores opções entre produtos idênticos.

1.3. Objetivos

O objetivo deste projeto é a criação de uma aplicação em android com um motor de busca para produtos de empresas com o propósito de dar ao utilizador um método simples de comparação entre os produtos disponíveis. Para desenvolver esta aplicação pretendemos:

- Procurar cooperação por parte das empresas no desenvolvimento na aplicação.

- Criar um Web Crawler, para procurar as páginas Web de interesse.
- Desenvolver um Parser, ou implementar um parser já existente no programa principal, para analisar o código html de cada empresa.
- Estudar a ontologia dos produtos, para a sua categorização depois da análise dos Websites.
- Criar uma base de dados para guardar toda a informação proveniente das páginas analisadas.
- Estudar e desenvolver interações servidor-cliente para implementação no programa.
- Converter o programa para android e realizar testes na plataforma.

2

2. Estado da Arte

Neste capítulo são abordadas aplicações e websites com funcionalidades semelhantes ao projeto da dissertação. É feito um estudo na ontologia computacional para ser entendido os problemas existentes na classificação dos produtos desejados.

Será feita uma breve comparação entre aplicações ou serviços concorrentes a este tipo de aplicação.

Também é realizado um breve resumo sobre as tecnologias necessárias para o desenvolvimento de uma aplicação ***web*** para *Android*, com um motor de busca com a capacidade de comparar preços de produtos idênticos provenientes de diferentes estabelecimentos, que seja capaz de propor ao utilizador a melhor opção para qualquer tipo de transação através da Internet.

2.1. Projetos de comparação entre produtos ou Shopbots

O uso da tecnologia para comparação de preços de produtos eCommerce, toma usualmente a forma de um Website ou de uma aplicação, denominadas de Ferramentas analíticas de preços, Agentes comparativos de compras, Shopbots, Pricebots ou motores de busca e comparação de compras [4, 5].

Estas ferramentas utilizam maioritariamente dois métodos para obter a informação a ser usada no seu serviço [5].

- O método mais comum, resume-se á procura de dados por Websites pelo menos uma vez por dia, guardando-os numa base de dados. O serviço disponibi-

liza esses dados ao utilizador quando algum pedido é feito. Este método permite ao Shopbot responder rapidamente e independentemente do estado do Website. No entanto pode induzir a erros nos dados disponibilizados devido á possível desatualização da informação.

- Um outro método pode ser descrito por uma querry direta aos Websites relevantes. Quando o Shopbot recebe um pedido do utilizador, procura nos Websites de interesse pelos dados necessários, que leva um certo tempo a processar. Este método está dependente do estado do Website, no entanto, qualquer informação dada ao utilizador estará atualizada.

Ambos estes métodos usam um tipo de software designado por WebCrawlers, o qual será exposto no subcapítulo 2.3.

2.1.1. Projetos existentes

Tipicamente, serviços de comparação entre produtos não cobrão aos seus utilizadores pelos serviços prestados. Em vez disso, são financiados através das empresas que se inscrevem para ser listadas no website ou aplicação. Dependendo do modelo de negócios praticado por parte dos donos do website, as empresas com os produtos a listar, pagam pelo numero de clicks nos seus produtos, ou, pelo numero de acções completadas sobre os seus produtos, tal como compras ou registo com o seu e-mail.

Também existem, companhias especializadas apenas na recolha de dados, e consolida-los com a correta ontologia com o propósito de os utilizar na comparação de preços, cobrando aos seus clientes o acesso a estes dados. Quando estes produtos são adicionados aos seus Websites, o custo de obter os dados é pago com o dinheiro proveniente de clicks que resultem em compras efetuadas nos websites de onde o produto originou. Os resultados de pesquisas, ou até produtos visíveis nas páginas de entrada de websites de comparação, são muitas vezes ordenados dependendo da receita dos sites vendedores [6].

Fazendo uma análise a websites de comparação de e-commerce em Portugal, encontra-mos uma tendência para a recolha de dados de empresas vendedoras fora do mercado de retalho, onde apenas produtos como, carros, eletrodomésticos, telemóveis, acessórios, roupas, brinquedos, informática, livros, e até combustíveis, entre outros, eram realmente comparados. Esta tendência foi quebrada a 12 de Janeiro de 2016 com o lançamento do website da companhia Maiscarrinho, com o foco exclusivo na comparação de produtos entre os hypermercados portugueses [7]. Com esta inovação no catálogo dos shopbots portugueses, outros websites começaram a disponibilizar o mesmo leque de serviços, em que o website de comparação de produtos mais visitado em Portugal, KuantoKusta, rapidamente acompanhou. Foi feita uma análise aos sites com shopbots em Portugal (figura 1) que possuam um foco para além de hotelaria ou viagens.

Serviços	Website	Apk Android	Apple	Carrinho de compras / Listagem	Retalho	Fora Retalho	Preço	Promoções	Preço por quantidade	Rating
Buscapé	✓	✓	✓	×	×	✓	×	×	×	×
Kelkoo	✓	✓	×	×	×	✓	✓	✓	×	×
Izideal	✓	×	×	×	×	✓	✓	×	×	✓
Shopmania	✓	✓	✓	✓	×	✓	✓	✓	×	×
Maiscarrinho	✓	✓	✓	✓	✓	×	✓	✓	✓	×
Kuantokusta	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figura 1 - Análise de empresas com comparação de produtos

2.2. Ontologia

Quando o conhecimento de um domínio é representado com um formalismo declarativo, o conjunto de objetos que podem ser representados é denominado por universo de discurso [8].

Em ciência computacional, tal como em ciência de informação, uma ontologia é a designação formal e definição das categorias, propriedades, e inter-relacionamentos das entidades existentes no universo de discurso. Por outras palavras, uma ontologia proporciona um vocabulário partilhado que pode ser usado na modelação de um domínio [9].

No âmbito deste projeto, uma ontologia é utilizada na designação e organização de todos os produtos, com o intuito de apresentar ao utilizador produtos presentes na mesma categoria de interesse de uma forma correta.

2.2.1. Componentes Ontológicos

Independentemente do contexto em que são utilizadas, existe uma similaridade nas estruturas básicas dos componentes. A maioria das ontologias descreve pelo menos, indivíduos, classes ou conceitos, atributos e relações.

- **Indivíduos:** indivíduos ou instâncias, são o componente básico, de mais baixo nível na árvore ontológica. Os indivíduos de uma ontologia podem incluir tanto objetos concretos, como pessoas, animais, plantas, tal como

indivíduos abstratos como números e palavras. Uma ontologia pode até não incluir indivíduos, no entanto o propósito de uma ontologia existe na capacidade de classificar indivíduos, mesmo que estes indivíduos não façam parte da ontologia.

- **Classes:** de acordo com a definição na representação de conhecimento, uma classe é um conjunto de indivíduos ou objetos [10]. É um conceito muitas vezes denominado por tipo, categoria ou. Existe uma variedade de ontologias em relação á utilização de classes, tal como se estas classes podem conter outras classes, se a classe pode pertencer a ela mesma, ou se poderá existir uma classe universal (ou seja, uma classe que contenha todas as entidades da ontologia).
- **Atributos:** entidades em uma ontologia podem ser descritos relacionando-os a outros objetos, tipicamente através de "partes" da entidade. Estas "partes" são denominadas de atributos. Cada atributo pode ser uma classe ou um individuo. O tipo de objeto e o tipo de atributo determina o seu relacionamento. Por exemplo, utilizando um refrigerante como o objeto, podemos estipular os seus atributos como sendo, a sua quantidade, o seu preço, sabor, marca, ou até o seu próprio nome. Existem dois tipos de atributos, atributos de objetos ou atributos de dados. Os atributos de objetos criam relacionamentos entre indivíduos, enquanto que atributos de dados, ligam um individuo a um tipo de dados com um certo valor [11].Ontologias só poderão existir se houver uma relação entre conceitos (com atributos que os relacionem, como pode ser observado na figura 2).
- **Relações:** as relações entre objetos dentro de uma ontologia especificam como estes objetos estão relacionados entre si. Tipicamente a relação é de um certo tipo, que especifica em que sentidos os objetos estão relacionados dentro da ontologia. Por exemplo, a utilização de um atributo que define um certo objeto B como o sucessor do objeto A, que permite criar um relacionamento entre entidades dentro da mesma classe, sem ser necessário chamar a classe "mãe" para estabelecer esse relacionamento. Estes relacionamentos são automaticamente estabelecidos na criação de estruturas hierárquicas.

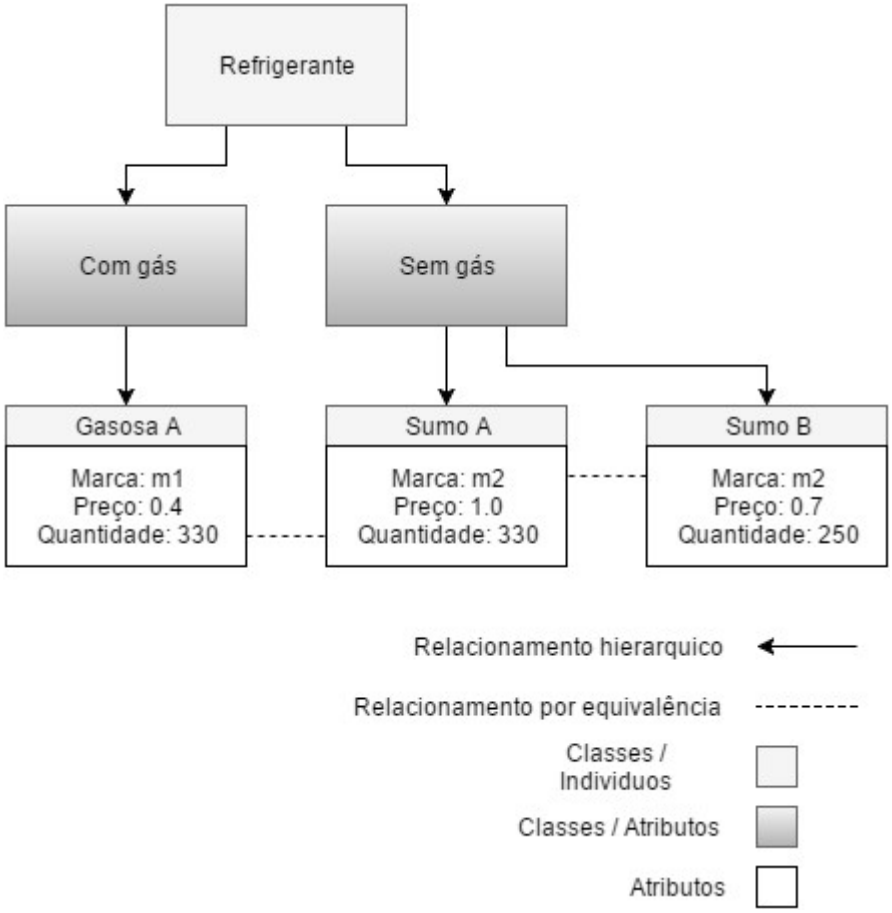

Figura 2 - Exemplo de uma Ontologia

Para além dos componentes mais comuns, os seguintes componentes também poderão ser utilizados na descrição de entidades em sistemas com funcionalidades mais específicas (normalmente em ciência computacional, ou de informação):

- **Termos funcionais:** estruturas complexas, formadas a partir de relacionamentos que poderão ser usados no lugar de um termo individual.
- **Restrições:** descrições formais do que deve ser valido para que certas afirmações sejam aceites como entradas no sistema.
- **Regras:** declarações condicionais (if→then) que descrevem as inferências logicas que podem ser tiradas de certas afirmações.
- **Eventos:** entidades que quando ativas possuem a capacidade de alterar atributos ou relações

2.2.2. Linguagem Ontológica para a semântica da Web

Em ciência computacional tal como em inteligência artificial, linguagens ontológicas são consideradas linguagens formais usadas na construção de ontologias. Este tipo de linguagens permite a codificação de conhecimentos sobre certos domínios, incluindo regras logicas para suportar o processamento desse conhecimento. Elas geram uma linguagem natural, integram informação inteligente, proporcionam um acesso à Web baseado em semântica, e extraem informação de textos para alem de serem usadas por diferentes aplicações para explicitamente declarar conhecimentos enraizados nestes textos [12].

Nos subcapítulos seguintes, são apresentados os tipos de linguagens ontológicas utilizadas para descrever entidades na Web, baseadas em XML.

2.2.2.1. Resource Description FrameWork (RDF)

O esquema da RDF proporciona um vocabulário modelado para dados. Este esquema é acompanhado por uma serie de documentos que descrevem os conceitos base, e a abstração da sintaxe a ser utilizada. As classes e propriedades do RDF são semelhantes a tipos de sistemas com linguagens de programação orientadas a objetos tal como java. Este esquema difere na definição de classes em termos das propriedades das instâncias que a constituem, descrevendo essas propriedades consoante os recursos da classe ao qual estão aplicados. Por exemplo, poderíamos definir o atributo eg:autor dentro do domínio eg:documento com o "leque" de eg:pessoa, enquanto num sistema clássico o atributo eg:autor seria definido com o tipo eg:pessoa [13]. O benefício desta abordagem está no facto de permitir a extensão da descrição de recursos existentes, neste caso do "leque" eg:pessoa.

2.2.2.2. Ontology Interchange Language (OIL)

Devido ao monopólio natural da Internet em termos de linguagem a ser utilizada, deve ser formulada uma ontologia com a sintaxe correspondente aos padrões da Web. OIL, possui uma sintaxe bem definida baseada em documentos em XML e num esquema compatível com o esquema usado em XML. OIL pode ser considerado uma extensão de RDF [14]. O benefício claro da utilização de OIL, foi a sua criação com um foco na semântica da Web, sendo rapidamente adotado por Web developers. No entanto OIL, possui uma grave falha visto que não possui a capacidade de comparar os valores de dados, permitindo apenas relações binarias. Existem ferramentas desenhadas para OIL que facilitam a criação e modificação das ontologias [14]:

- **Editor de Ontologias:** desenhados para construir novas ontologias de raiz.
- **Ferramentas de anotação baseadas em Ontologia:** com o propósito de ligar fontes de informação sem estrutura e fontes semiestruturadas a ontologias.
- **Ferramentas logicas Ontológicas:** permite a implementação de serviços com pedidos-resposta com ontologias, suportam a criação de novas ontologias, e ajudam ao mapeamento entro diferentes ontologias crias as relações necessárias para efetuar os pedidos corretos.

2.2.2.3. Ontology Web Language (OWL)

OWL, é uma linguagem desenhada especificamente para processar informação proveniente da Web, tornando-se em Fevereiro de 2004 uma recomendação da W3C [15]. Esta linguagem foi construída com base na estrutura de RFD, mas proporcionando um integração e interoperabilidade com os dados de diferentes domínios. Relativamente a XML, que proporciona uma sintaxe base para documentos estruturados, mas que impõe restrições semânticas no significado destes documentos, OWL, adiciona mais vocabulário na descrição de propriedades e classes, entre outras entidades, permite o estabelecimento de relações entre classes, características em atributos, e anumeração de classes.

2.3. Web Crawlers

Web Crawlers têm sido amplamente utilizados por motores de busca com o propósito de recolher conteúdo proveniente da Web. Estes crawlers tendem a ser autónomos com a sua manutenção a ser feita manualmente. Com o crescimento rápido de serviços online, que usam Web Crawlers com o intuito de recolher paginas Web, as suas funcionalidades têm vindo a evoluir e a tornarem-se mais diversas.

O seu funcionamento típico inclui pedidos a paginas web para indexação de texto e pesquisas, extração de mails e de dados pessoais para o seu uso em negócios (em grande parte publicidade) e também para uso malicioso [18].

2.3.1. Tipos de Web Crawlers

Neste subcapítulo iremos descrever os tipos existentes de crawlers referentes ao método pesquisa e da escolha das prioridades de páginas escolhidas [19]:

- **Crawlers clássicos**: recebem como entrada um pedido de um utilizador que descreve o tópico, um conjunto de URLs de paginas, que guiam o Crawler para paginas de interesse. Neste pedido esta incorporado um critério de prioridades para dar a certos links maior prioridade de download baseado na probabilidade desse mesmo link conter dados sobre o tópico pedido pelo utilizador. O Crawler procede sistematicamente para os links contidos na página acedida.
- **Crawlers semânticos**: são considerados uma variação dos Crawlers clássicos. As prioridades de download são concedidas a páginas que sejam semanticamente semelhantes ao critério, ou seja, uma pagina e um tópico podem estar relacionados caso partilhem conceptualmente termos semelhantes. Estas semelhanças entre termos podem ser definidas utilizando ontologias.
- **Crawlers inteligentes**: é aplicado um processo de treino para designar prioridades de visita sobre páginas web e para guiar o processo de crawling. Este tipo de crawlers é caracterizado pela forma que paginas web relevantes e caminhos por links na web para sites com conteúdo relevante é aprendido pelo

crawler. Tipicamente o Crawler inteligente é posto perante um treino composto por páginas Web relevantes e não relevantes, que são utilizadas para treinar o crawler. São designadas prioridades de visita sobre os links extraídos da web consoante a sua relevância perante o tópico escolhido. É tido em conta não só o conteúdo da página e a correspondente classificação da página como relevante ou não relevante, mas também a estrutura de links subjacentes e a probabilidade dessa estrutura ser relevante consoante o número de saltos entre páginas necessários para encontrar o conteúdo pretendido.

2.3.2. Arquitetura (Design) de Web Crawlers

O funcionamento de um Web Crawler pode ser descrito através dos seguintes passos (todo este processo pode ser observado na figura 3) [19]:

1. **Parametrização:** O Crawler, recebe como dados de entrada um conjunto de URLs e de uma descrição do tópico pretendido. Esta descrição pode ser um conjunto de palavras-chave para crawlers clássicos e semânticos ou um tipo de treino no caso de ser um crawler inteligente.
2. **Download:** É feito o download do conteúdo das páginas dadas ao crawler e os links contidos nestas mesmas páginas são colocados em queue. Dentro da queue os links serão ordenados por ordem de relevância ou no caso de o link não ser relevante é excluído da queue.
3. **Processamento do Conteúdo:** As páginas transferidas no passo dois são lexicalmente analisadas e reduzidas a vetores, e o conteúdo é filtrado e guardado (este processamento de conteúdo é realizado por parsers que serão abordados no subcapítulo 2.4).
4. **Designação de prioridade:** Os links extraídos no passo dois são postos numa fila de espera ordenada com base no tipo de crawler utilizado e das especificações do utilizador. Estas especificações podem ir deste critérios simples como importância da página ou relevância com o tópico pedido.
5. **Expansão:** Os URLs escolhidos são usados para expandir a pesquisa do crawlers, utilizando-os como parâmetros de entrada e repetindo o passo

dois. Este processo repete-se até um critério previamente estabelecido ser cumprido (como por exemplo numero de paginas limite a transferir) ou até os recursos do sistema estejam esgotados.

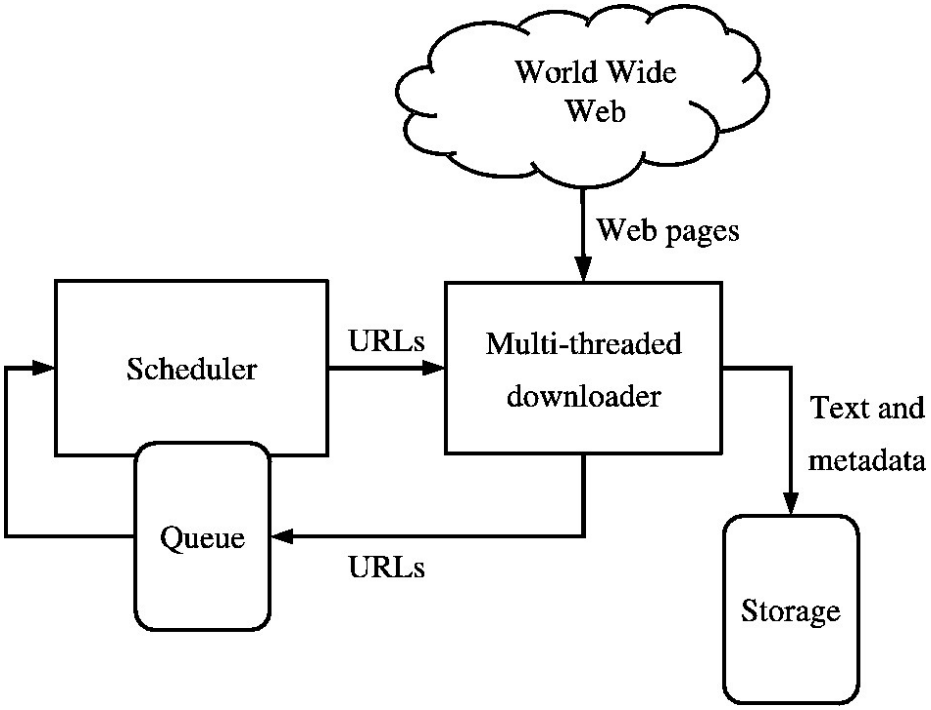

Figura 3 - Arquitetura de alto nível de um Web crawler [20]

2.4. Parsers

Neste subcapítulo iremos fazer uma descrição de Parsers e da sua função no ramo da ciência computacional, tal como da sua estrutura logica.

2.4.1. Parsing

Parsing é o processo realizado por Parsers (que são apenas um programa informático) para analisar uma sequência de caracteres, sejam estes caracteres parte de uma estrutura linguística natural ou de uma linguagem informática, sendo esta analise feita respeitando as regras gramaticais da língua em questão. O termo possui significados ligeiramente diferentes quando comparado entre o seu uso no ramo linguístico e no ramo da ciência de computadores, mas iremos focar-nos unicamente no ramo computacional.

A sua funcionalidade básica é composta pela analise de strings dadas ao programa, separando o conteúdo da string nos seus componentes básicos, chamados tokens (representados pelos elementos de mais baixo nivel na figura 4), e criando uma "arvore" que mostre a relação sintática entre os componentes, que no caso de a string dada ao Parser ter origem num código html seria de esperar uma separação pelo menos entre a head, body, title ou qualquer outro componente de interesse.

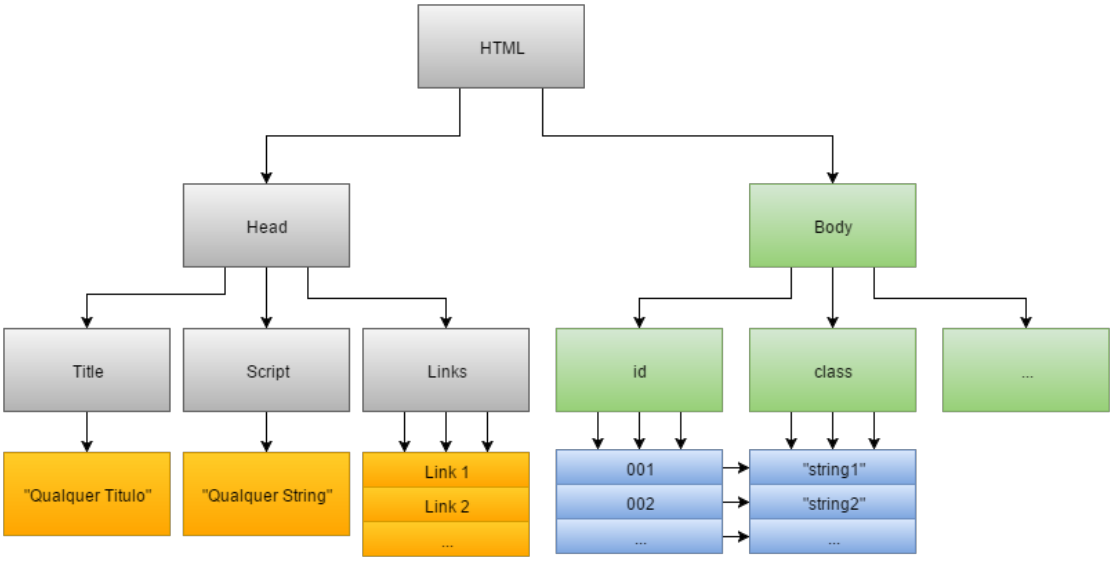

Figura 4 - Exemplo da "Arvore" de Parsing de um código html

Parser são normalmente considerados uma estrutura parcial de um programa maior, como por exemplo crawlers, interpretes ou compiladores. Os Parsers utilizados nestes tipos de programas, recebem uma entrada e criam uma "árvore" consoante as necessidades do programa principal. A "árvore" resultante possui tipicamente uma estrutura hierárquica que descreve a relação entre os tokens parsados [21].

2.4.2. Arquitetura de um Parser

Um Parser é composto por duas partes [21]:

1. **Analisador lexical:** este componente encarrega-se da análise lexical da string recebida, produzindo um conjunto de tokens que representam a string de entrada de acordo com a gramatica lexical da linguagem de programação analisada (representados na figura 4 pelas caixas de nível

inferior). Esta separação de tokens pode ser feita filtrando certos caracteres chave, tais como: “espaço”, _ , . , > , < , ….

2. **Analisador sintático:** como pode ser observado na figura 5 os tokens criados são passados para o Analisador sintático. Analisando a ordem e a disposição dos tokens, o Analisador cria a “árvore” de parsing e devolve-a ao programa principal.

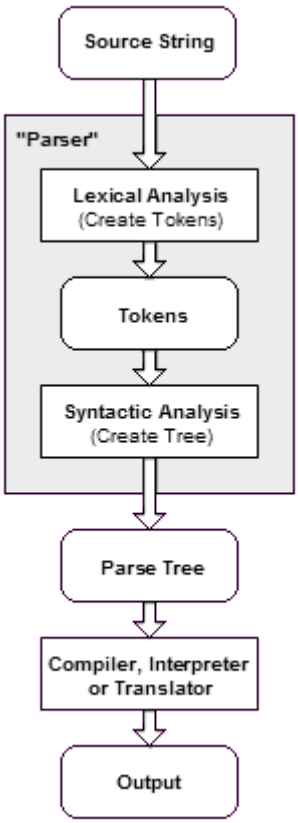

Figura 5 - Estrutura de um Parser [22]

2.4.3. Tipos de Parsers

O propósito de um Parser, é essencialmente determinar como é que os tokens presentes numa string podem ser organizados a partir do símbolo inicial da gramatica presente. Isto pode ser executado de duas formas:

- **Parsing Top-down:** este tipo de parsing pode ser visto como uma tentativa de encontrar as derivações dos tokens mais a esquerda na entrada de dados, procurando em “arvores” de Parsers dentro da gramatica usada, que sejam compatíveis com essa mesma derivação [22].

- **Parsing Bottom-up:** este tipo de parser começa apartir do input do token, e tenta chegar ao símbolo de maior nível (símbolo inicial). Este parser separa primeiro os elementos mais básicos (tokens), e de seguida agrupando-os nos elementos que os contêm [22].

2.4.4. Parsers existentes para html

Aqui faremos uma breve listagem de alguns Parsers atualmente mais utilizados para análise de código html:

- **Html Agility Pack:** este parser esta desenvolvido para C# sendo uma biblioteca em .NET que permite o parsing de ficheiros html extraídos directamente da net. O parser é bastante tolerante ao HTML usado no "mundo real" [23].
- **Jsoup:** é uma biblioteca Java muito usada na análise de HTML. Capaz de fazer parsing de HTML a partir de um URL, ficheiro ou string, e manipular elementos, atributos ou texto. [24]
- **Nokofiri:** é um parser para Ruby que permite uma simples extração de dados de documentos HTML. Menos robusto que outros parsers dos mais utilizados em Ruby. [25]
- **jQuerry:** é uma biblioteca feita para JavaScript, que permite completa manipulação do documento, inclusive interação direta com eventos, animações utilizando uma API capaz de funcionar em vários browsers. [26]
- **PHP Simple HTML DOM Parser:** este parser requer php5+ suportando html inválido, sendo capaz de extrair conteúdos de HTML presentes numa só linha sem ter de devolver o resto do código. [27]

3

3. Proposta

Neste capítulo abordaremos o problema proposto para a dissertação, onde será feita uma análise á solução escolhida. Será também listado os requisitos de funcionamento da aplicação proposta, e apresentada a arquitetura de um sistema que seja capaz de encapsular todos os requisitos e solucionar o problema proposto.

3.1. Análise do Problema

Tal como já foi referido anteriormente, as vantagens provenientes do eCommerce podem ser sumarizadas por aumento da competitividade do mercado, estando todas as empresas ao alcance do utilizador, isto permite a escolha de produtos tendo em conta todos os produtos "rivais" que possam ter melhor qualidade ou preço, e também pela comodidade visto que a grande maioria de empresas presentes em eCommerce possuem um serviço de entrega ao domicílio, removendo qualquer vantagem que uma empresa com um produto de gama inferior possa ter apenas pela sua localização.

Infelizmente devido ao leque de escolhas disponível ao utilizador, este pode não conseguir decidir qual o melhor produto ou poderá fazer a escolha de produtos menos benéficos. Esta falta de filtragem sobre o conteúdo fornecido ao utilizador é prejudicial á experiencia do cliente ao utilizar a internet para comércio e aumentando ainda mais a falta de confiança no eCommerce.

Isto cria uma procura no mercado por ferramentas que deem ao utilizador os meios para não só encontrar os produtos com os melhores preços mas também de ser

capaz de os encomendar com facilidade independentemente da empresa em que estes produtos se encontram disponíveis.

O desenvolvimento desta aplicação têm como propósito dar ao utilizador uma ferramenta base, que seja capaz de não só mostrar ao utilizador os produtos disponíveis em empresas diferentes, comparando-os, mas que também que seja capaz de escolher pelo utilizador qual o produto especifico que lhe é mais benéfico de entre os diferentes fornecedores.

3.2. Especificações do Sistema

Foi estipulado que a aplicação a desenvolver deveria de ser suportada por a plataforma Android, capaz de aceder a um conjunto de dados provenientes de diferentes fontes e capaz de guardar para o utilizador as suas preferências. Em suma as especificações do sistema são as seguintes:

- Compatibilidade com a plataforma Android;
- O sistema deve dar ao utilizador acesso aos dados pretendidos, independentemente do estado em que a fonte desses dados se encontra;
- Dar a capacidade ao utilizador de criar uma lista de produtos a serem tratados pela aplicação;
- Possuir uma interface intuitiva;
- Deve restringir o utilizador na manipulação de dados, de forma a proteger a integridade do sistema;
- O sistema deve ser desenhado com a prospeção na escalabilidade do sistema;

3.3. Solução Proposta

Neste subcapítulo será demonstrada a estrutura para o sistema proposto e também dos seus componentes, através de diagramas e modelos de forma a dar uma visão mais abrangente do projeto desenhado.

3.3.1. Arquitetura do Sistema

Com o intuito de desenhar um sistema com o objetivo de solucionar o problema apresentado no subcapítulo 3.1 e capaz de cumprir com os requisitos do projeto especificados no subcapítulo 3.2, foi definida uma arquitetura baseada em Web Services, composta por uma base de dados, uma API (*Application Program Interface*), um Web Crawler (Aplicação Java) e uma aplicação criada na plataforma Android para servir de interface para o utilizador.

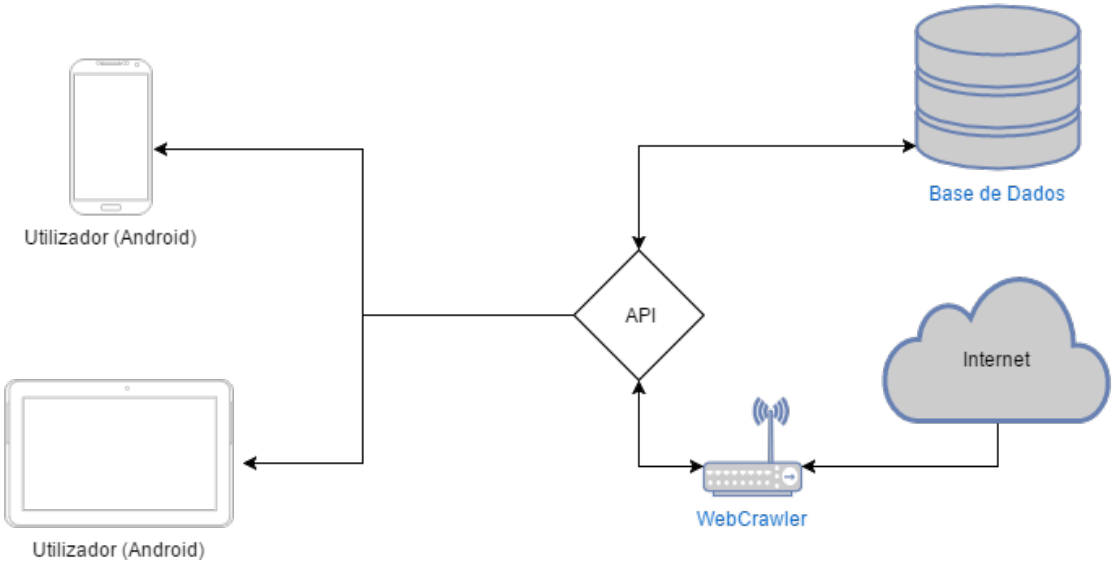

Figura 6 - Arquitetura de Alto Nivel

Com o tipo de arquitetura modular, observada na figura 6, temos a possibilidade de expandir as capacidades do sistema sem a necessidade de para o seu funcionamento, facilitando qualquer manutenção necessária ao projeto tal como melhoramentos de software sem por em risco a integridade estrutural da base de dados ou de qualquer outro subsistema existente.

3.3.2. Estrutura da API

A API (*Application Program Interface)* terá a função de tratar de qualquer tipo de comunicação entre os módulos do sistema, impedindo qualquer interação direta entre os serviços e com o utilizador.

Com o propósito de reduzir a carga na rede do servidor, foi escolhida uma arquitetura REST (subcapítulo 2.2.3), que estabelece o método de comunicação utilizado entre os vários serviços e clientes, no qual a API foi construída. São utilizadas as operações básicas CRUD (Create, Read, Update, Delete) para estabelecer a comunicação entre a base de dados e qualquer serviço que necessite, limitando o uso dessas operações para o utilizador onde apenas é permitida a leitura de dados pela aplicação Android (como pode ser observado na figura 7).

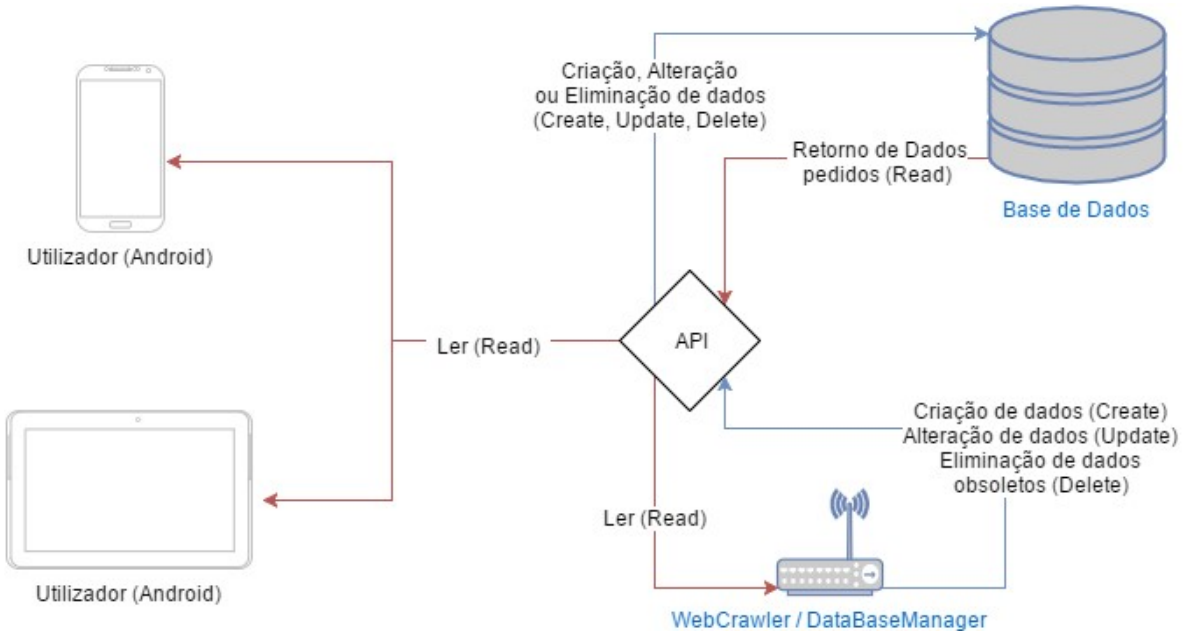

Figura 7 - Modelo de comunicação alto nível da API

3.3.3. Estrutura da Base de Dados

Este projeto utiliza o JavaDB (Derby), como a sua base de dados relacional, que é inteiramente implementada em Java. Nesta secção será explicada a estrutura funcional da base de dados, tal como, as suas tabelas, campos e relações.

- **Tabela: LOJA**

Nesta tabela, esta presente toda a informação necessária para a aplicação em referência às empresas. De acordo com os objetivos do projeto, é nesta tabela que é guardada o URL âncora de cada empresa correspondente, no campo "URL_LOJA". Este URL será complementado com campos presentes nas tabelas CATEGORIA e SUBCATEGORIA.

Campo	**Conteúdo**
LOJA_ID	Identificador numérico único (atribuído automaticamente).
NOME_LOJA	Nome da instituição.
URL_LOJA	URL base utilizado na procura de produtos (sob a forma de uma STRING).

- **Tabela: CATEGORIA**

A tabela CATEGORIA contêm o nome das classes de mais alto nível na ontologia dos produtos.

Campo	**Conteúdo**
CATEGORIA_ID	Identificador numérico único (atribuído automaticamente).
NOME_CATEGORIA	Nome da categoria utilizada na criação do URL final a ser executado no WebCrawler.

- **Tabela: SUBCATEGORIA**

Todas as subcategorias dos produtos possuem uma relação com a tabela CATEGORIA, a partir do identificador único presente. Esta relação está presente para dar a conhecer ao produto, a categoria a que este está inserido apenas com o identificador da tabela SUBCATEGORIA, tal como pode ser observado na figura 8.

Campo	**Conteúdo**
SUBCATEGORIA_ID	Identificador numérico único (atribuído automaticamente).
CATEGORIA_ID	Identificador único relacionado a tabela CATEGORIA
NOME_SUBCATEGORIA	Nome da subcategoria utilizada na criação do URL final a ser executado no WebCrawler.

- **Tabela: PRODUTO**

Nesta tabela, temos toda a informação que é dada ao utilizador acerca de cada produto. São utilizadas as chaves identificadoras das tabelas LOJA e SUBCATEGORIA para obter as origens do produto.

Campo	**Conteúdo**
PRODUTO_ID	Identificador numérico único (atribuído automaticamente).
SUBCATEGORIA_ID	Identificador único relacionado á tabela SUBCATEGORIA
LOJA_ID	Identificador único relacionado á tabela LOJA
NOME_PRODUTO	Nome do produto
MARCA	Noma da marca do produto
PRECO	Preço do produto guardado como um DECIMAL
PRECO_QUANTIDADE	Preço do produto em relação à quantidade total (obtido diretamente da empresa alvo, no entanto caso não seja possível obter este valor uma operação aritmética pode ser utilizada utilizando os valores de PRECO, QUANTIDADE e PESO_MIL ou LITROS_MIL.
PESO_MIL	Guarda o valor do peso em gramas se for possível. Caso não seja guarda "0".
LITROS_MIL	Guarda o valor do peso em mililitros se for possível. Caso não seja guarda "0".
QUANTIDADE	Quantidade do produto à venda.

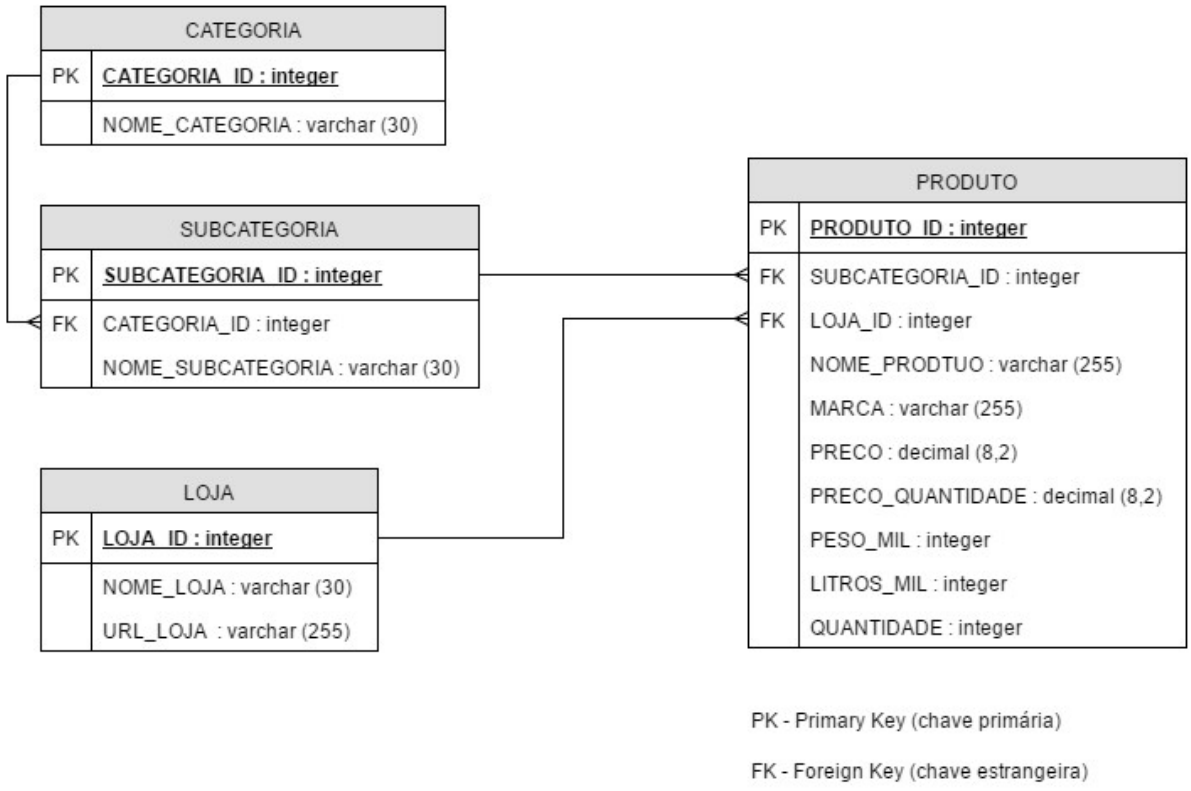

Figura 8 - Modelo Relacional de Entidades

3.3.4. Estrutura do WebCrawler

O Webcrawler tem como propósito percorrer a World Wide Web de forma metódica e até um certo ponto automática, obtendo dados relevantes à aplicação. Neste caso os dados relevantes que o Webcrawler procura são as informações presentes em websites com produtos relevantes ao projeto.

Presente na base de dados está um conjunto de URLs (Uniform Resource Locator) correspondentes a cada empresa de interesse. O Webcrawler utiliza esses URLs presentes na base de dados para estabelecer a base (ou ancora) dos URLs finais a serem utilizados na procura de dados.

O administrador escolhe qual a empresa á qual pretende executar o crawler, tal como o tipo de informação que quer adquirir (presente na base de dados sobre a forma de categorias), e acedendo de novo á base de dados, o Webcrawler adiciona ao URL base (que depende da empresa escolhida) o texto necessário presente nas categorias de interesse, obtendo assim os URLs finais a serem adicionados a lista de procura (queue).

Com a lista de URLs obtida a partir dos parâmetros iniciais estabelecidos pelo administrador, o crawler executa um pedido GET ao primeiro URL da lista para obter o seu código em html. Este código é colocado numa string que passará por um parser (subcapítulo 2.4) com o propósito de categorizar o conteúdo do html e filtrar os dados pretendidos, que depois de processados serão adicionados á base de dados como pode ser observado na figura 9, ou no caso de existirem dados obsoletos presentes na base de dados, estes são substituídos pelos dados mais recentes.

Por fim qualquer URL relevante presente no html será guardado e adicionado a queue de URLs a pesquisar, repetindo o processo até que a lista de URLs se encontre vazia.

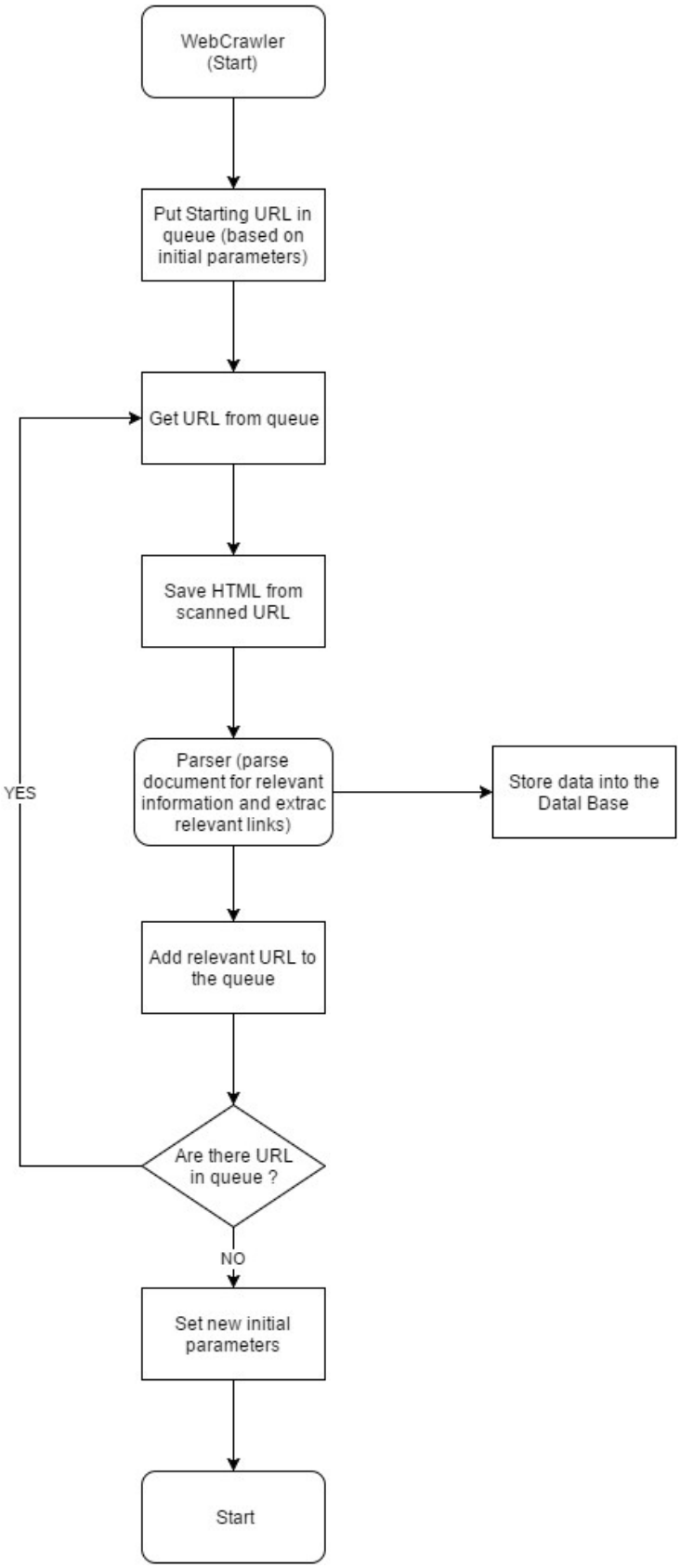

Figura 9 - Fluxograma do WebCrawler

3.3.5. Estrutura da Aplicação

Uma aplicação Android é composta por múltiplas atividades, de uma forma semelhante a um programa feito em java possui vários Jframes para interagir com o utilizador. Estas atividades [28] são usadas para a criação da interface, ou seja, a classe Activity tem a responsabilidade de criar a janela (ecrã – figura 10) que é usada para se inserir a UI (interface de utilizador) da aplicação. Sendo normalmente utilizadas para apresentar ao utilizador janelas que ocupem o ecrã inteiro, as atividades podem também ser usadas para criar pequenas janelas flutuantes ou inserir a sua janela dentro de outra atividade.

Desta forma cada atividade pode ser usada para iniciar uma atividade diferente para realizar diferentes ações. Cada vez que uma atividade é iniciada, a atividade anterior é parada, no entanto o sistema preserva a atividade num stack (back stack). Esta back stack usa um mecanismo de ultimo a entrar, primeiro a sair, logo, quando o utilizador terminou a sua utilização da atividade em focos e pressiona o botão de retroceder, a atividade em focos é removida (e destruída) e a atividade anterior é resumida. [29]

Neste projeto a aplicação possui três atividades principais. A atividade inicial (Main Menu), que é criada quando a aplicação é executada, contem os métodos para criar as outras duas atividades (Motor de Pesquisa, Lista de Compras).

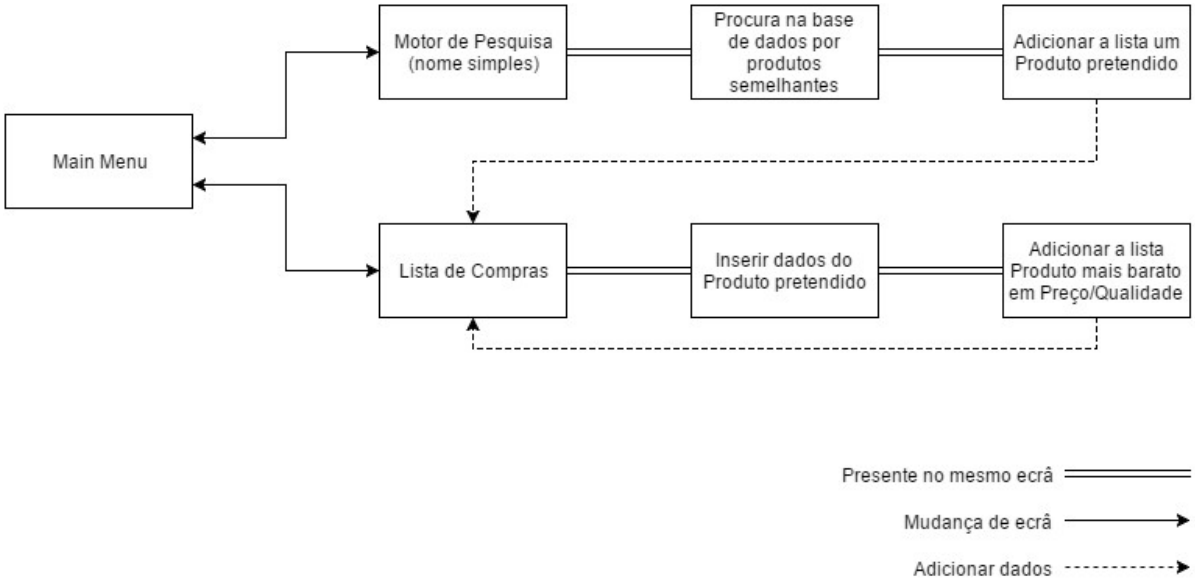

Figura 10 - Estrutura do projeto Android

Ainda presente na atividade Main Menu, está uma caixa de texto a ser preenchida, que é usada para enviar uma string de texto para a atividade Motor de Pesquisa quando esta é criada. Na criação da atividade Motor de Pesquisa, esse texto proveniente do Main Menu, é usado para obter todos os produtos que façam parte da categoria equivalente, ou subcategoria caso não seja encontrada uma categoria relevante. Depois de obtidos os produtos semelhantes, é feita uma procura por equivalência em relação á string inicial que devolve os produtos por ordem de relevância, com o propósito de filtrar os produtos desejados pelo utilizador para uma comparação posterior.

A lista filtrada é mostrada ao utilizador com a capacidade de a ordenar per ordem crescente, ou decrescente, relativamente ao seu preço consoante a quantidade do produto. A partir desta lista, poderemos adicionar produtos que interessem ao utilizador a uma lista de compras que será guardada localmente na memória do dispositivo.

Retornando ao Main Menu, também poderá ser criada a atividade da Lista de compras. Esta janela permite visualizar uma lista de produtos de interesse guardada. No topo da janela, também existe a possibilidade de inserir dados mais específicos (nome do produto, marca, quantidade) em relação ao produto, permitindo que seja adicionada automaticamente á lista, o produto com o preço mais barato correspondente aos dados inseridos.

4

4. Validação

4.1. Implementação e Decisões

Com o objetivo de dar aos utilizadores os dados de variados produtos, de forma a estes poderem fazer uma decisão informada entre quais os produtos mais económicos, foi necessário fazer o estudo de um meio de obter esta informação. Na altura de escolher o método a implementar para obter os dados desejados de cada empresa, procurou-se obter cooperação das empresas através do acesso a uma API (Aplication Program Interface) que nos permitisse obter os dados dos seus produtos. Visto que esta opção requereria um nível de cooperação global entre todas as empresas de interesse e o projeto, opção que requer demasiado tempo em relação ao deadline do projeto, teria de ser pensada uma solução em que a participação destas mesmas empresas toma-se um formato passivo no caso de esta cooperação não ser obtida atempadamente.

Após ter sido estudado um método viável de obter estes dados, conclui-se que a construção de um WebCrawler capaz de eficazmente obter os dados desejados seria a melhor opção. Estes dados seriam guardados em uma base de dados, acedidos como um webservice, de forma a dar o acesso à informação desejada independentemente do estado dos sites das empresas em questão.

Devido aos pré-requisitos da dissertação em criar a interface dos utilizadores numa aplicação Android, só restava estudar a melhor plataforma onde iria ser implementado o WebCrawler. Não existindo uma plataforma superior em todos os aspetos, foi escolhido fazer este WebCrawler em Java devido as seguintes razões:

- Possui uma fundação estável e testada ao longo de 20 anos por uma empresa (Sun/Oracle).

- Uns dos melhores IDEs no mercado (Integrated Development Environment ou Ambiente de Desenvolvimento Integrado), tal como Eclipse, NetBeans ou IntelliJ, com a integração de bons debuggers, compiladores e servidores.
- O número de bibliotecas disponíveis (especialmente open source), capazes de solucionar a maioria de problemas inerentes a qualquer programa e de se tornarem a fundação de projetos. Um dos problemas desta dissertação, presentes na utilização de xmls do tipo JSON para comunicar com a API, que seria solucionável pela utilização de um outra linguagem como o Node.js, possui diversas bibliotecas dedicas a este propósito em Java.
- Threads múltiplas por parte do Java Web servers, mantendo estabilidade nos servidores ao custo de um maior tempo na sua criação e de alguma memória.

4.1.1. Construção e funcionamento do WebCrawler

Esta secção dá uma visão abstrata de como o WebCrawler desta dissertação foi desenhado, e fará uma descrição do seu funcionamento interno. Podemos então observar na figura 11 uma abstração do seu funcionamento modular.

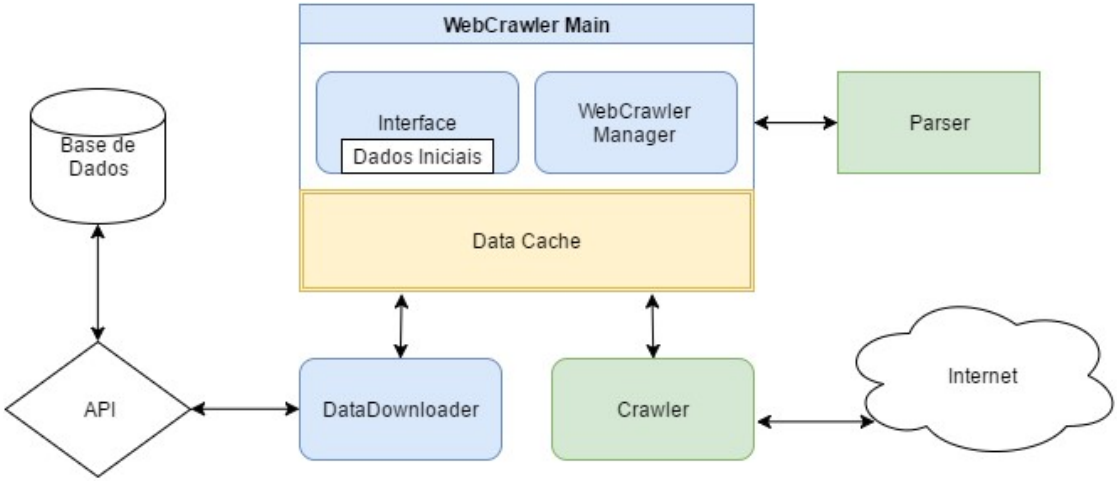

Figura 11 - Abstração funcional do WebCrawler

- **WebCrawler Manager:** é responsável pela chamada e comunicação entre todos os módulos do sistema e organização de dados. É ativado com o botão Start presente na interface do programa. Também é responsável pela comparação entre os dados retirados dos websites e os dados já existentes na base de dados, de forma a saber que dados podem ou não ser substituídos por dados mais recentes.
- **Interface:** utilizada para escolher os dados iniciais para "alimentar" o crawler, iniciar o crawler, dar inicio ao crawling ou fechar o programa. Em vez de os URLs ou dados iniciais poderem ser inseridos directamente, é dada ao administrador do crawler a opção de escolher a empresa em questão, tal como a categoria de dados (sob a forma de um inteiro, equivalente ao identificador único na tabela CATEGORIA da base de dados) para efetuar o crawling, tal como pode ser vista na figura 12. Este formato foi escolhido para a eventual automatização completa do processo e também de forma a limitar o erro humano.

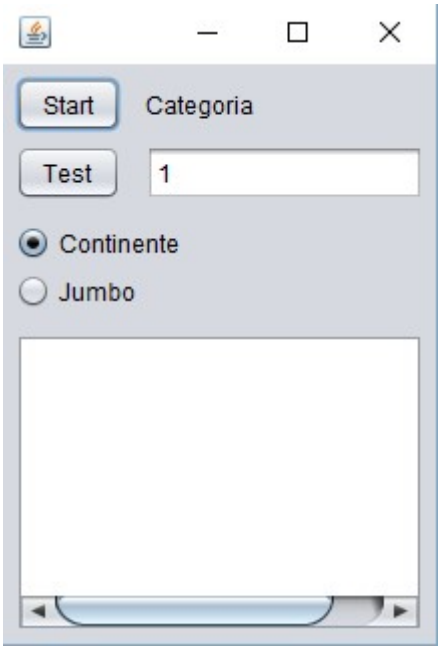

Figura 12 - Interface do WebCrawler

- **Data Cache**: contem várias estruturas de dados utilizadas pelo programa com o propósito de limitar as comunicações necessárias com a base de dados. É onde fica guardada a queue dos URLs, a informação obtida dos websites, a listagem de produtos proveniente da base de dados tal como os produtos retirados do html pelo parser.
- **Crawler:** é responsável por gerir os downloads das páginas de uma forma ordenada. É composto por várias classes, no entanto a sua modulação pode ser simplificada em dois módulos: o downloader, e um manager. O manager obtém os links existentes na queue, e retorna o código html ao Data Cache retirado do site pelo downloader. O manager fornece ao downloader os links retirados da queue, com um intervalo de tempo entre cada link cedido, associado ao limite de pedidos num certo espaço de tempo que uma empresa aceita provenientes do mesmo ip (se este intervalo de tempo for demasiado curto, o site alvo poderá bloquear temporariamente pedidos, atrasando todo o processo de crawling ou até causar a recuperação de dados incompletos). No caso de se fazer uma procura múltipla com empresas diferentes, poderá ser criada uma thread do downloader para cada empresa.
- **Parser:** quando o Data Cache contém um html, o parser é responsável pelo parsing de dados (explicado no subcapítulo 2.4). É de notar que para cada

empresa diferente será necessária a criação de uma classe específica para lidar com o html dessa mesma empresa, devido a alterações nas estruturas e nomes das classes únicas a cada empresa. Depois de os dados serem retirados do html e organizados, são adicionados à data cache (tanto os produtos como os links retirados, sob a forma de estruturas separadas).

- **DataDownloader:** Este módulo possui os métodos de comunicação com a API, e esta responsável por obter os dados necessários da base de dados e de fazer qualquer alteração necessária á base de dados consoante a informação recolhida pelo crawler.

4.1.2. Funcionamento da aplicação Android

Neste subcapítulo será demonstrado a logica interna da aplicação que tem por objetivo tratar da interação entre o utilizador e os serviços do sistema.

Para cumprir com o objetivo principal, é necessário dar ao utilizador a capacidade de poder encontrar produtos semelhantes ao que procura, independentemente da empresa de origem. A lista de produtos a ser mostrada ao utilizador, também deve estar organizada de forma a facilitar a comparação entre os vários produtos do mesmo género. Outro dos objetivos a ser cumprido é a capacidade de criar uma lista de produtos de interesse. Foi criado então no menu principal os botões que criaram esses mesmos serviços, tal como podemos observar na figura 13 que demonstra a aplicação depois de construída no simulador integrado do Android Studio, Gradle.

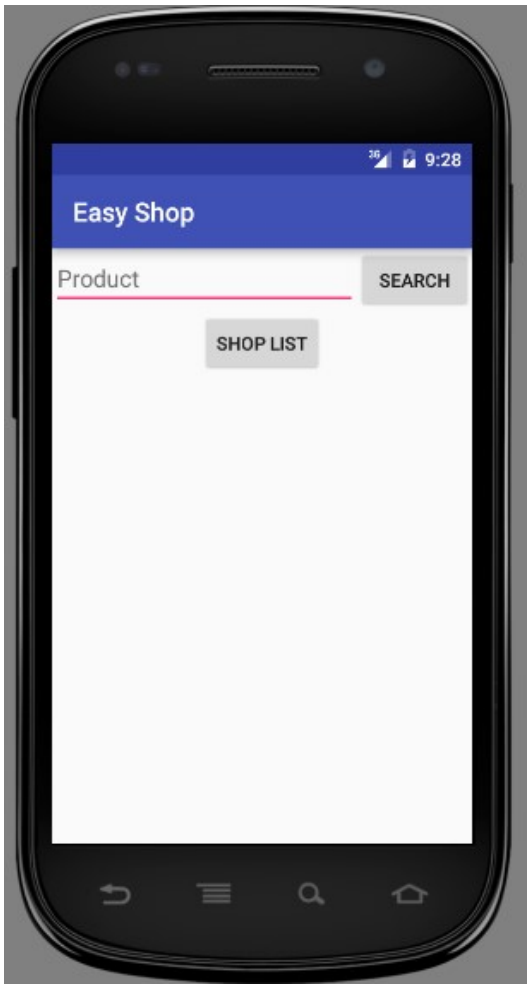

Figura 13 - Main Menu da aplicação

Ainda no Main Menu tal como pode ser verificado na figura 13, podemos inserir um nome que identifique o género de produtos que pretendemos procurar. Esse nome pode ser o próprio nome do produto, a categoria em que esta inserido, a marca do suposto produto, ou uma combinação de dados que identifique o produto para uma procura mais especifica.

A obtenção dos dados é realizada por um módulo presente na aplicação que trata de toda a comunicação necessária com o WebService através da API. Foi também estudada a melhor forma de obter os dados pretendidos da base de dados, em que três opções distintas foram encontradas (testes foram executados que demonstravam uma média de 6 kilobytes por cada 10 produtos presentes na mensagem).

- Utilizar a API para fazer um pedido direto à base de dados de forma a obter dados semelhantes até um certo grau. Este método tem a vantagem de ser o menos consumista dos recursos da rede para cada pedido individual, visto que diminui o tamanho das mensagens para o menor possível. No entanto existe o problema de se passar parte do processamento para a base de dados por cada pedido em vez de esta ser tratada em cada dispositivo individualmente, que poderá aumentar o tempo de resposta da base de dados ligeiramente quando o sistema for escalado em termos de utilizadores.
- Passar todos os dados de produtos para uma cache dentro da aplicação na criação do Main Menu, que permite à aplicação correr sem problemas de rede depois da sua construção, visto que toda a informação pretendida passará a estar disponível localmente. Com este método as interações com a base de dados diminui drasticamente, no entanto o tamanho do pacote de dados iniciais seria o maior de todos os métodos. Assim com o escalamento de utilizadores o efeito no servidor é bastante reduzido, no entanto com o escalamento do lado do serviço com a adição de demasiados produtos, poderá existir demasiada informação irrelevante para que este método possa sequer ser viável.
- Por fim, temos um método intermédio, entre os dois previamente demonstrados. A aplicação ao fazer o pedido por dados, especifica a categoria em que este pedido foi feito, ou o tipo de marcas pretendida, pedindo à base de dados através da API todos os produtos pertencentes à

categoria alvo, guardando essa categoria em cache mantendo-a disponível para o utilizador no caso de este pretender obter mais dados presentes nesta categoria. Só no caso de uma nova categoria (ainda não presente na cache) ser procurada é que um novo pedido será feito à base de dados, para ser adicionado à cache local. Este método não necessita de pedidos sucessivos à base de dados para qualquer procura que o utilizador pretenda, reduzindo o tráfico na rede em relação ao primeiro método, e não necessitando de mensagens com um pacote de dados demasiado grande como é demonstrado no segundo método.

Ao pressionar o botão SEARCH é aberta uma janela (visível na figura 14), que utiliza o texto obtido na caixa de texto no Main Menu, para chamar a classe que trata na comunicação com a API para obter os dados necessários (no caso de estes ainda não se encontrarem disponíveis na cache locar da aplicação), colocando-os na cache. De seguida é chamada uma classe que trata da criação e ordenação de uma lista (ListView), a ser mostrada no ecrã. Esta lista é preenchida com os dados mais próximos ao texto introduzido no Main Menu, utilizando o método de relevância para fazer a filtragem dos produtos menos desejados ao atribuir valores numéricos a cada produto dependendo da sua proximidade ao texto inicial, e retornando os produtos com o maior valor de relevância.

Figura 14 - Search List da aplicação

Estes produtos são então ordenados de acordo com o seu preço em relação à quantidade do produto, com a capacidade de reordena-los por ordem crescente ou decrescente a partir dos botões situados no topo de ecrã.

Ao clicar num dos produtos desta lista, esse produto será adicionado a uma lista local disponível ao utilizador no ecrã Lista de Compras. Quando o produto é adicionado é dada a confirmação ao utilizador com uma mensagem visível na imagem 15.

Figura 15 - Mensagem de confirmação na adição do produto

Por fim se o utilizador pressionar a partir do Main Menu o botão Shop List (visível na figura 13), irá ser apresentada uma janela com a lista de produtos guardada localmente. Nesta mesma janela foi adicionada a opção de inserir dados mais específicos em relação ao produto, para que a aplicação possa escolher o produto mais barato em vez do utilizador, adicionando-o à lista de compras automaticamente, tal como é demonstrada na figura 16.

Figura 16 - Lista de Compras da aplicação

Ao clicar em algum produto da lista, este produto será removido da cache local e a lista será atualizada no ecrã (figura 17).

Figura 17 - Eliminação do item da lista de compras

4.2. Validação da comunicação da API entre a base de dados e o Webcrawler

A comunicação que a API possibilita entre a base de dados e o webcrawler poderá ser demonstrada com a visualização da passagem dos dados corretos entre os serviços. Para esta validação, será demonstrado o funcionamento prático do webcrawler em uma pequena escala, seguido do envio dos dados tradados para a base de dados.

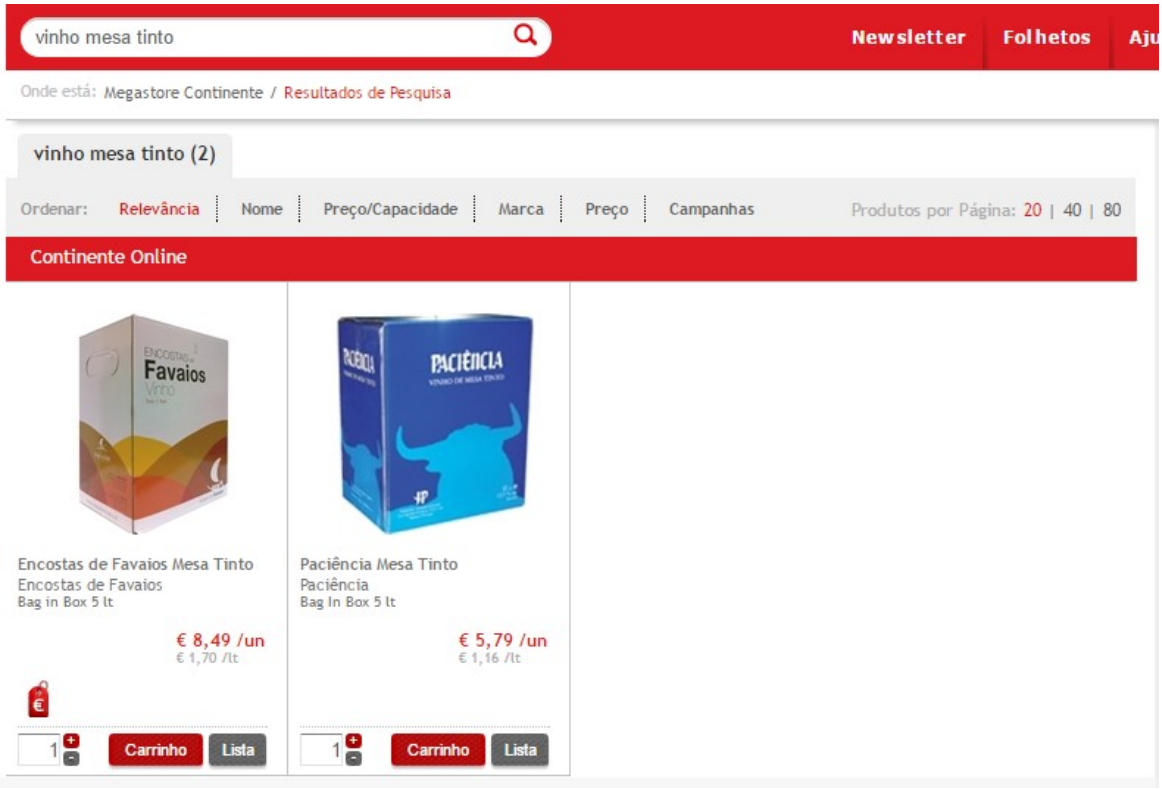

Figura 18 - Pagina web de validação

Na figura 18, é mostrada a página Html onde o código será obtido e tratado pelo Webcrawler. A confirmação que o Crawler obteve os URLs de cada produto e foi capaz de tratar os seus dados, numa estrutura de dados compatível com a tabela PRODUTO da base de dados, pode ser observada na figura 19, que apresenta o runtime do Webcrawler.

```
run:
Obter o URL da loja
ListData(45):6
entrou no main cicle, lista categoria:[vinho mesa]
inicio do main cicle, lista categoria:[vinho mesa] categoria_id:6
getNomeSubCategoria(33): procurar por subcategorias relevantes ate ao id 45
inicio do submain cicle, lista subcategoria:[tinto, branco, rose]
Iniciou o processo de crawling, pesquisa(cat, subcat):vinho mesa tinto
tamanho da url cache:2
https://www.continente.pt/stores/continente/pt-pt/public/Pages/ProductDetail.aspx?ProductId=46
[0, 0, 0, Encostas de Favaios Mesa Tinto, Encostas de Favaios, 8.49, 1.70, 0, 5000.0, 1, 0]

https://www.continente.pt/stores/continente/pt-pt/public/Pages/ProductDetail.aspx?ProductId=40
[0, 0, 0, Paciência Mesa Tinto, Paciência, 5.79, 1.16, 0, 5000.0, 1, 0]

Tamanho do cache:0
Obter a Lista de Produtos da base de dados
Entrar no db_manager com a lista:[[0, 0, 0, Encostas de Favaios Mesa Tinto, Encostas de Favaio
missing list output:[56, 828, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 857]
fim do submain cicle, lista subcategoria:[branco, rose]
inicio do submain cicle, lista subcategoria:[branco, rose]
Iniciou o processo de crawling, pesquisa(cat, subcat):vinho mesa branco
```

Figura 19 - WebCrawler runtime

Conseguimos observar os dados dos dois produtos extraídos por baixo dos seus URLs respetivos. Resta apenas comprovar a receção dos dados por parte da base de dados, que é demonstrada na figura 20 com um querry a base de dados derby. Estes dados foram inseridos na tabela PRODUTO com os identificadores únicos 56 e 828 (sendo os identificadores únicos com o valor mais baixo disponível, estes identificadores disponíveis são localizados pelo crawler e apresentados na consola em "missing list output").

	SUBCATEGORIA_ID	LOJA_ID	NOME_PRODUTO	MARCA	PRECO	PRECO_QUANTIDADE
855	11	1	Concentrado Frutas Tropicais	Sunquick	4.69	5.58
856	11	1	Concentrado Pêssego e Laranja	Sunquick	4.69	5.58
56	17	1	Encostas de Favaios Mesa Tinto	Encostas de Favaios	8.49	1.70
828	17	1	Paciência Mesa Tinto	Paciência	5.79	1.16

Figura 20 – Resultados da querry á base de dados

4.3. Validação da comunicação da base de dados com a aplicação

Nesta secção confirmaremos que os pedidos feitos pela aplicação à base de dados através da API são processados corretamente.

Com um simples teste através do ecrã da Lista de Compras, que permite realizar um pedido específico, é então requisitada á base de dados uma água da marca Luso, com uma quantidade de cerca de 1,5 Litros. O produto relevante ao pedido de teste, pode ser identificado com um sombreado a azul na lista da base de dados da figura 21.

	SUBCATEGORIA_ID	LOJA_ID	NOME_PRODUTO	MARCA	PRECO	PRECO_QUANTIDAD	PESO_MIL	LITROS_MIL	QUANTIDADE
3	1	1	Água sem Gás	Luso	0.45	0.45	0	1000	1
4	1	1	Água sem Gás Mineral Nat...	Luso	0.65	0.87	0	750	1
6	1	1	Água sem Gás	Luso	1.29	0.24	0	5400	1
9	1	1	Água sem Gás	Luso	1.46	0.21	0	7000	1
5	1	1	Água sem Gás	Luso	1.69	0.85	0	330	6
23	2	1	Água com Gás	Luso	1.76	1.57	0	250	6
8	1	1	Água sem Gás Mineral Nat...	Luso	1.85	0.93	0	330	6
7	1	1	Água sem Gás Mineral Nat...	Luso	2.29	0.76	0	500	6
2	2	1	Água com Gás Limão	Luso	2.35	2.35	0	250	4
10	1	1	Água sem Gás	Luso	3.42	0.38	0	1500	6
42	4	1	Refrigerante sem Gás Coc...	Luso	3.61	1.39	0	0	0

Figura 21 - Resultado parcial da querry á base de dados da marca luso

A confirmação da receção dos dados por parte da aplicação pode ser observada na figura 22, que mostra no ecrã Lista de Compras o resultado do pedido por um produto com as características inseridas pelo utilizador com o preço mais barato possível.

Figura 22 - Ecrã Lista de Compras da aplicação

4.4. Resultados experimentais e problemas conceptuais

Tendo em conta os objetivos técnicos desta dissertação, irá ser discutido neste capítulo os problemas e soluções encontradas no desenvolvimento do projeto. Ao longo do tempo de vida do projeto, entre Novembro 2015 até Junho 2016, foram realizados números testes de crawling entre os websites das empresas escolhidas para a dissertação (Continente e Jumbo), tal como em websites fora destas empresas no estudo do funcionamento de WebCrawling em si.

Durante esse período de tempo, foram detetadas duas alterações na estrutura do código usado pela empresa continente, tornando o parsing do html dos seus sites ineficaz, devido à alteração no nome das classes utilizadas. Nestas duas situações, o módulo que tratava exclusivamente do parsing dos websites controlados pela em-

presa continente, necessitou de alterações à sua logica interna para se tornar novamente funcional. Uma terceira alteração, ao seu site, causou a recusa de pedidos que não fossem provenientes de um browser, sendo necessário introduzir no módulo de crawling um biblioteca capaz de simular os pedidos efetuados por um browser (é de notar que esta biblioteca não emulava um browser em si, mas simplesmente os pedidos efetuados).

Devidos as alterações regulares necessárias aos módulos de parsing e crawling, em resposta aos sites alvo, foi decidido subdividir cada um desses módulos para cada website alvo (passando a existir eficazmente dois módulos distintos dedicados a cada empresa).

Ao longo do estudo de webcrawling, é também de notar que em testes a websites com conteúdo por detrás de um javascript, um simples pedido pelo código html poderá já não ser suficiente para obter toda a informação necessária. Nestes casos a solução encontrada foi a utilização de um biblioteca chamada htmlunit, construída pela Apache Maven, que é capaz de emular um browser a escolha (sendo o Chrome ou Mozilla Firefox os mais utilizados) e deixar executar o javascript, fazendo o pedido pelo código posteriormente.

Os pedidos sucessivos a websites poderá causar um bloqueamento temporário ao IP. Este evento pôde ser verificado não só devido ao reduzido tempo entre pedidos mas também pelo número de pedidos dentro de um certo intervalo de tempo. É então necessário criar temporizadores entre pedidos para não causar o bloqueamento do ip, atrasando por sua vez o processo de crawling. O tempo a definir depende de website para website, sendo necessários testes sucessivos para encontrar o tempo a dar ao temporizador que irá otimizar todo o processo (os testes feitos na identificação do tempo a usar entre pedidos podem ser visualizados na figura 23).

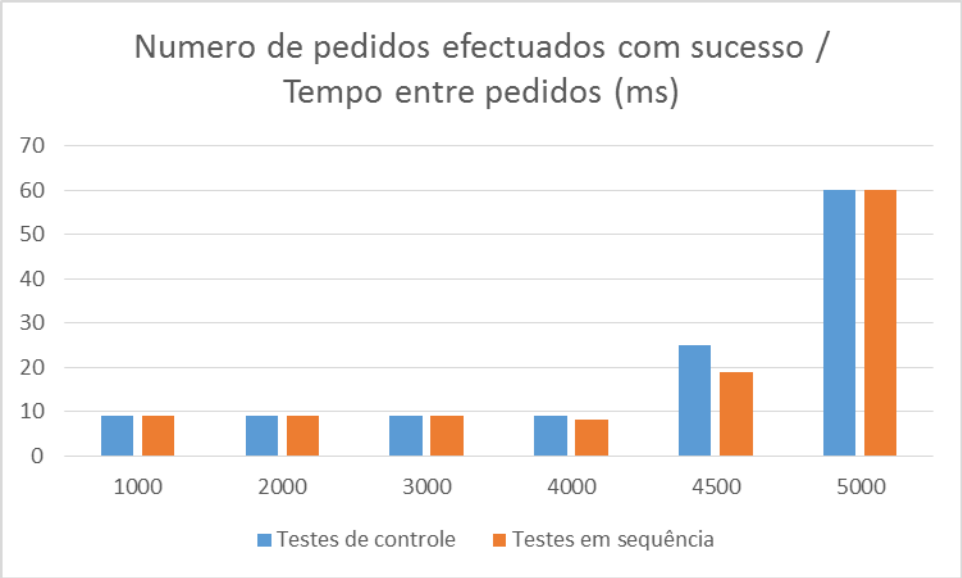

Figura 23 - Testes dos tempos entre pedidos

Analisando o gráfico, que testa 60 pedidos em sequência, podemos concluir que o website testado bloqueia o acesso ao seu conteúdo dependentemente do número de pedidos num certo espaço de tempo, neste caso 9, em vez de bloquear pelo tempo entre pedidos. Quando o tempo entre pedidos é aumentado para 4500 milissegundos verificamos que o bloqueamento do ip se encontra no limiar dos pedidos necessários para ativar esse bloqueamento (a discrepância entre o teste de controle e o teste subsequente é explicada pelo tempo de processamento da pagina pelo parser que poderá variar).

Foram utilizados dois métodos diferentes para efetuar o crawling dos sites:

- No caso do continente foi utilizado o seu próprio motor de pesquisa para encontrar os produtos relevantes. Esta escolha deveu-se a um bom motor de pesquisa por parte do continente que permitia a uma procura mais flexível por parte dos produtos através do URL que acedia ao motor de busca. São criadas novas subcategorias na base de dados, mais específicas em relação ao tipo de produto pretendido, para encontrar qualquer produto relevante à categoria. As amostras pedidas ao website também devem ser pequenas de forma a apenas mostrar os produtos mais relevantes á pesquisa, no caso de amostras demasiado grandes serem requisitadas podem ser fornecidos dados que se encontrem fora da

categoria que se pretende preencher, causando um indexação errada na base de dados.

- No caso do Jumbo que possuía um motor de busca algo impreciso, foi necessário fazer uma busca através das categorias presentes no site. Isto diminui o tempo de procura, visto não ser necessário procurar fora das categorias existenetes, mas restringe a flexibilidade do crawler, sendo necessário criar métodos específicos para aceder as categorias do website. Estes métodos devem utilizar dados da base de dados, e não diretamente no código do programa, de forma a tornar o sistema o mais genérico possível.

Outra alteração no método de crawling entre estes dois websites foi a utilização dos URLs que apontavam diretamente para o produto. No caso do continente, eram procurados os links individuais de cada produto para se conseguir aceder a dados mais específicos e informativos acerca do produto. Este método foi considerado o método mais demorado devido ao tempo de espera necessário entre cada pedido de cada produto, sendo necessário um tempo de espera de cerca de aproximadamente 2 min por cada subcategoria.

No caso do Jumbo, foram obtidos todos os dados necessários na página de apresentação dos produtos. Este método permite a obtenção de dados de uma subcategoria inteira através da mesma página, reduzindo o tempo de espera de 2 min para 5 segundos entre cada subcategoria. Mesmo com uma clara vantagem em termos de tempo, este método não é recomendado para todas as situações pois quebra o acesso a dados específicos de cada produto, presentes na sua pagina individual.

5

5. Conclusão

O conceito de eCommerce ainda não está bem estabelecido em Portugal quando comparado com outros países europeus [3]. Para tentar difundir a ideia e para benefício tanto das empresas como dos clientes, estamos a criar uma aplicação para android capaz de comparar preços entre empresas e dar ao cliente a melhor opção de entre os seus produtos. Se possível e como plano futuro de trabalho para a aplicação, tornar possível a compra direta pela aplicação e não apenas para funcionar como um motor de pesquisa.

Depois da análise feita às empresas é de notar que apenas duas delas possuíam os preços no seu site (e dessa forma acessível pelo seu código html), e que uma delas já possui uma aplicação feita capaz de efetuar compras online. Sem uma colaboração direta com as empresas será necessária a criação de uma base de dados para guardar o conteúdo dos sites, e de um web crawler para encontrar e catalogar todo o conteúdo de interesse.

Infelizmente para as empresas onde os seus produtos não estão listados nos seus respetivos websites ou que não possuam preços sobre os produtos, é impossível obter essa informação com a utilização de um Web Crawler, sendo necessária a cooperação da empresa caso queiramos adicionar o seu catálogo á aplicação.

Com esta dissertação podemos entender a estrutura necessária para a criação de uma aplicação que dê ao utilizador a capacidade de poder observar e comparar com facilidade o preço entre produtos provenientes de fontes diferentes, desde a criação da interface na forma de uma aplicação android, até á forma de obter e transmitir os dados necessários para a aplicação, através do estudo de webcrawling e webservices.

Em suma podemos concluir que os objetivos do projeto foram alcançados com sucesso, tendo sido criada uma base robusta para futura uma expansão em funcionalidades. Ainda existe alguma dificuldade no crawling de páginas dependentes em javascript, sendo um campo a melhorar no futuro. Devido à inabilidade de ligar os códigos de barras aos produtos através dos websites, a secção do projeto que permitiria ao utilizador obter os produtos necessários com um simples scan ao código de barras não pode ser adicionada.

Com um enfase em funcionalidade, a interface projetada tomou um caminho simplista. No entanto o melhoramento do seu design e a adição de novas funcionalidades à aplicação deverá ser considerado como essencial para a sobrevivência de uma aplicação deste género no mercado.

5.1. Trabalhos futuros

Durante o desenvolvimento deste projeto e análise dos seus resultados e problemas, foram observados vários pontos a ser melhorados.

Primeiro e possivelmente o mais importante, seria obter a cooperação das empresas de forma a ter acesso às suas APIs para obter todos os dados sem a necessidade de webcrawling, e também obter os métodos de efetuar as compras diretamente da aplicação. Estas empresas beneficiariam da sua inclusão no projeto sob a forma de publicitar os seus produtos.

Adicionar à aplicação a capacidade de processar códigos de barras e retornando uma string que aplicação seja capaz de utilizar para encontrar produtos equivalente. E posteriormente incluir um leitor de códigos de barras para a aplicação, com o objetivo de dar ao utilizador a possibilidade de adicionar à sua lista de compras produtos desejados com um simples scan.

Por fim, criar clientes individuais de webcrawling para cada empresa, e talvez até para cada categoria, de forma a permitir crawlings múltiplos utilizando a API. Caso estes clientes possuam IPs diferentes, poderá ser resolvido o problema de pedidos em sequência à mesma empresa.

Referências

[1] ComputerWorld, João Nobrega, 90% do potencial do ecommerce português e desaproveitado, http://www.computerworld.com.pt/2012/10/02/90-do-potencial-do-ecommerce-portugues-e-desaproveitado/

[2] Luís Carlos Silva Morais Cavadas, What is inhibiting Portuguese consumers from buying on-line, Universidade Católica Portuguesa, Julho 2015, http://repositorio.ucp.pt/bitstream/10400.14/19287/1/What%20is%20inhibiting%20Portuguese%20consumers%20from%20buying%20on-line.pdf

[3] Alexandre Nilo Fonseca, President, ACEPI, E-Commerce in Europe & Portugal, 2014, www.aicep.pt/framework/download.php?id=207

[4] Shopbots and Pricebots, Amy R. Greenwald, Jeffrey O. Kephart, http://cs.brown.edu/~amy/papers/ijcai.pdf

[5] Shopbots become agentes for business change , D. Clark, 06 de Agosto 2002, http://ieeexplore.ieee.org/document/820034/

[6] Loginworks, How shopping bots really work, 18 de Dezembro 2014, http://www.loginworks.com/informative/shopping-bots-really-work/

[7] Maiscarrinho, 12 de Janeiro 2016, https://maiscarrinho.com/about

[8] Thomas R. Gruber, Knowledge System Laboratoy, A Translation Approach to Portable Ontology Specifications, http://tomgruber.org/writing/ontolingua-kaj-1993.pdf

[9] Fredrik Arvidsson, Annika Flyncht-Eriksson, Ontologies I, http://www.ida.liu.se/~janma56/SemWeb/Slides/ontologies1.pdf

[10] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Description Logics: Foundations for Class-based Knowledge Representations, Universitá di Roma, http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=110B072B7265573684AB8D4F0D6B2306?doi=10.1.1.177.2787&rep=rep1&type=pdf

[11] Denny Vrandecic, Ontology Evaluation, 10 de Junho de 2010, http://www.aifb.kit.edu/images/b/b5/OntologyEvaluation.pdf

[12] A. Gomez-Perez, O. Corcho, Ontology languages for the Semantic Web, Janeiro/Fevreiro de 2002, http://ieeexplore.ieee.org/document/988453/

[13] W3C, RFD Schema 1.1, 25 de Fevereiro de 2014, https://www.w3.org/TR/rdf-schema/

[14] Dieter van Harmelen, Ian Horrocks, Deborah McGuinnedd, Peter Patel-Schneider, OIL:Na Ontology Infrastructure for the Semantic Web, 2001, http://www.few.vu.nl/~frankh/postscript/IEEE-IS01.pdf

[15] W3C, OWL Web Ontology Language Overview, 10 de Fevereiro de 2004, https://www.w3.org/TR/owl-features/

[16] Douglas K Harry, *Representational State Transfer (REST)*. http://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html [Acedido em: 12 – Fev – 2016]

[17] R. Fielding, UC. Irvine, J.Gettys, Hypertext Transfer Protocol, Junho de 1999. https://www.w3.org/Protocols/rfc2616/rfc2616.html [Acedido em: 12 – Fev – 2016]

[18] C. Lee Giles, Yang Sun, Isaac G. Councill, *Measuring the Web Crawler Ethics*, Raleigh, North Carolina, USA, ISBN: 978-1-60558-799-8

[19] Sotiris Batsakis, Euripides Petrakis, Evangelos Milios, *Improving the performance of focused web crawlers*, Technical University of Crete (TUC), Chania, Crete, Greece, 7 de Abril 2009, ISBN 0169-023X

[20] Carlos Castillo, *Effective Web Crawling,* Ph.D. thesis in Computer Science, University of Chile, 2004.

[21] Grume,Dick and Jacobs, Ceriel J.H., Introduction To Parsing*: Parsing Techniques: A Pratical Guide*, 2008.

[22] Aho, Sethi, Ullman, *Compilers: Principles, Techniques, and Tools,* Addison-Wesley, 1986. ISBN: 0-201-10088-6

[23] CodePlex, Project Hosting for Open Source Software, Html Agility Pack. http://htmlagilitypack.codeplex.com/ [Acedido em: 18 – Fev – 2016]

[24] jsoup, jsoup: Java HTML Parser, http://jsoup.org/ [Acedido em: 18 – Fev – 2016]

[25] Nokogiri, http://www.nokogiri.org/ [Acedido em: 18 – Fev – 2016]

[26] jQuery, write less, do more, http://jquery.com/[Acedido em: 18 – Fev – 2016]

[27] S.C. Chen, Jose Solorzano, Yousuke Kumakura, PHP Simple HTML DOM Parser, http://simplehtmldom.sourceforge.net/ [Acedido em: 18 – Fev – 2016]

[28] Android Developers, Develop, API Guides, App Components, Activity, https://developer.android.com/reference/android/app/Activity.html [Acedido em: 3 – Jun – 2016]

[29] Android Developers, Develop, API Guides, App Components, Activities, https://developer.android.com/guide/components/activities.html [Acedido em: 3 – Jun – 2016]

[30] W3C, Web Services Arquitecture Group, *Web Services Glossary*, 2004. https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice [Acedido em: 05 – Jan – 2016]

[31] Douglas K Harry, *Service Architecture.* http://www.service-architecture.com/articles/webservices/web_services_explained.html [Acedido em: 05 – Jan – 2016]

[32] Online Community for the Universal Description, Discovery, and Integration OASIS Standart. http://uddi.xml.org/uddi-101 [Acedido em: 06 – Jan – 2016]

[33] Papazoglou, Michael. *Web Services: Principles and Technology*. s.l.: Prentic Hall, 2008.

[34] Yuping Yang, Enterprice Computing, *SOA & Web Services*, University of Exeter, 2006.

[35] W3C, Web Services Arquitecture Group, *Web Services Characteristics*, 2003. https://www.w3.org/2003/Talks/0828-hh-wsidgjp/slide5-0.html [Acedido em: 09 – Jan – 2016]

[36] Erik W., Florian M., Stefan L., *Leveraging the Web Platform for the Web of Things: Position Paper for W3C Workshop on the Web of Things*, Abril 2014.

[37] W3C, Web Services Aequitecture Group, *Web Services Description Language*, 15 de Março 2001. https://www.w3.org/TR/wsdl [Acedido em: 24 – Jan – 2016]

[37] Oracle, Structure of a WSDL document, http://download.oracle.com/otn_hosted_doc/jdeveloper/1012/web_services/ws_wsdlstructure.html

[38] Erl, Thomas. *Service-Oriented Architecture, Concepts, Technology and Design*: Prentice Hall Indiana, Julho de 2005. ISBN 0-13-185858-0

[39] B. Upadhyaya, Y. Zou, H. Xiao., Ng. Joanna, Alex Lau, *Migration of SOAP – based Services to Restful Services*, Queen´s University Kingston, Ontario, Canada, 2011. ISBN 9781457706981

[40] W3C, World Wide Web Consortium, *SOAP Version 1.2: Primer*, 27 de Abril 2007. https://www.w3.org/TR/soap12-part0 [Acedido em: 11 – Fev – 2016]

[41] W3C, World Wide Web Consortium, *SOAP Version 1.2: Messaging Framework*, 27 de Abril 2007. https://www.w3.org/TR/soap12-part1 [Acedido em: 11 – Fev – 2016]

[42] P.A. Castillo, J.L. Bernier, M.G. Arenas, J.J. Merelo, P. García-Sánchez, *SOAP vs REST: Comparing a master-slave GA implementation*, 2011. arXiv:1105.4978v1

[43] Microsoft, Developer Network, Sample SOAP message: https://msdn.microsoft.com/en-us/library/cc485825.aspx

[44] Flylib: http://flylib.com/books/2/439/1/html/2/images/f02mp01.jpg

[45] Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

[Acedido em: 12 – Fev – 2016] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Anexos

1.1. Web Services

Um Web Service é um sistema de *software*, desenhado para suportar interações interoperacionais entre máquinas, através de uma rede [30]. Essencialmente podemos afirmar que é qualquer tipo de *software* que se esteja disponível pela internet e forneça os seus serviços utilizando qualquer tipo de mensagem para comunicar, sendo XML (eXtensible Markup Language) o tipo mais comum de linguagem utilizada em comunicação pela Web.

1.2. Tipo de Arquitetura

A arquitetura usada para a criação de um Web Service, possui três "atores" principais, o Service Provider, o Service Consumer e o Registo de Serviços:

- **Service Provider:** O Service Provider constrói o serviço e torna-o disponível na Internet para consumidores a partir do registo feito no Registo de Serviços. O Service Provider responde a pedidos feitos por parte do Service Consumer com uma mensagem do tipo XML.
- **Service Consumer:** Este representa qualquer consumidor do Web Service. O solicitante invoca um Web Service existente no registo abrindo uma ligação de rede com o prestador e enviando uma mensagem do tipo SOAP ou REST (ver no subcapítulo 2.2).

- **Registry:** Podemos considerar este "ator" como uma base de dados centralizada de serviços. O registo é usado como uma central por parte dos prestadores de serviço para publicarem novos serviços disponíveis para a rede.

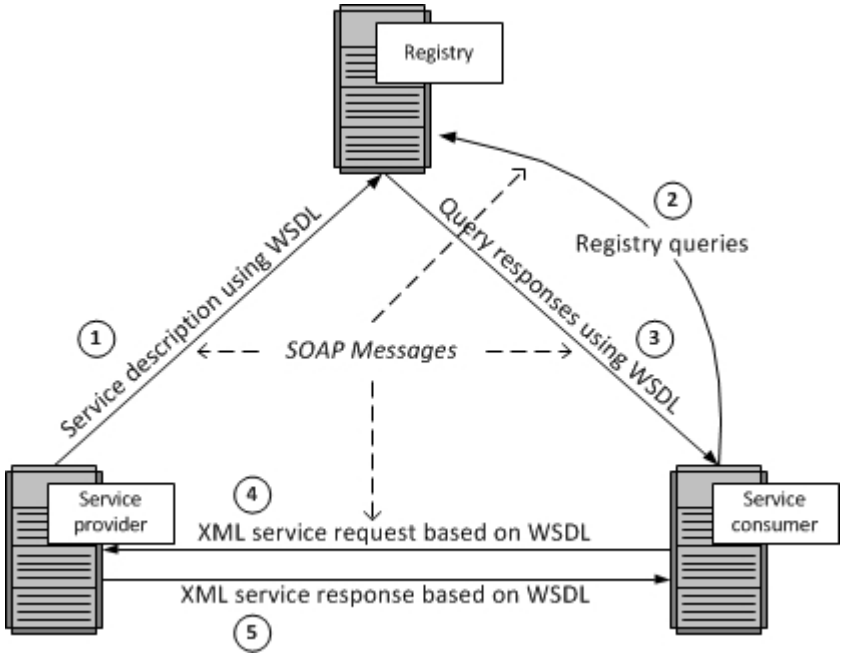

Figura 24 - Arquitetura de Web Services [31]

A Web Services Description Language (WSDL), forma a base para as especificações originais da arquitetura de Web Services. A figura 24 demonstra o uso de WSDL (padroniza o tipo de mensagens XML a ser utilizado na rede, de forma a possibilitar a comunicação). Os passos envolvidos em providenciar ao consumidor o servido requerido estão enumerados na figura 24 ordenadamente e podem ser descritos da seguinte forma: [31]

1. O Service Provider descreve o seu serviço utilizando WSDL. Esta descrição é publicada no Registo de Serviços.
2. O Service Consumer cria uma ou mais solicitações (queries) para o repositório de serviços de forma a localizar os serviços requeridos e determinar como se deve estabelecer a comunicação com esse serviço.
3. Parte do WSDL fornecido ao Registo por parte do Service Provider é passado ao consumidor. Isto diz ao Service Consumer quais são os pedidos e respostas disponíveis ao Service Provider.
4. O Service Consumer usa WSDL para enviar um pedido ao Service Provider.

5. O Service Provider envia a resposta esperada por parte do consumidor.

O tipo de Registo representado na figura 24 representa o protocolo Universal Description, Discovery, and Integration (UDDI). Este protocolo define o método padrão de publicar e encontrar componentes de uma arquitetura orientada a serviços (SOA) em rede. O padrão define um grupo de Web Services e interfaces de programação para publicação, obtenção, e gestão de informação sobre serviços [32].

1.3. Registo de Serviços

Como já foi mencionado no subcapítulo 2.1.1, o registo de serviços é utilizado como uma base de dados que fornece e armazena dados sobre serviços disponíveis, definindo o método de publicação e de comunicação para com o serviço.

1.4. Características do protocolo Universal Description, Discovery, and Integration UDDI

O UDDI especifica uma framework baseada em XML para a descrever Web Services e estabelecer o procedimento padrão para comunicar com serviços, procurando simular o que se encontra no registo de negócios, com nomes sofisticados e um bom serviço de diretorias.

Também define estruturas de dados e APIs para a publicação de descrições de serviços no Registo de Serviços e para pedidos feitos ao registo com o intuito de encontrar essas mesmas descrições. Por fim torna o próprio Registo de Serviços acessível da mesma forma que um Web Service.

Podemos então categorizar os tipos de informações guardadas no Registo UDDI [36]:

- Listar a organização, informação de contacto, e os serviços provenientes de cada organização.
- Separar os serviços por tipo de serviço prestado.
- A forma de invocar um dado Web Service.

1.5. Características de Web Services

De acordo com [34, 35, 36] podemos sumarizar as características dos Web Services como sendo:

- Baseado em XML:
  Uma mensagem é a unidade de comunicação com o Web Service. Estas mensagens são construídas com XML para transporte e representação de data. O uso de XML evita o acoplamento com qualquer rede, sistema operativo ou plataforma.
- Desacoplamento do Utilizador:
  Não providencia ao utilizador uma interface gráfica (GUI). Desta forma alterações á interface do Web Service não deterioram a capacidade do utilizador interagir com o serviço.
- Suporte para o Chamamento Remoto de Procedimentos (RPCs):
  Os Web Services permitem a clientes invocar métodos e operações a objetos remotos utilizando protocolos baseados em XML (como SOAP). Partilha a lógica de negócios, dados e processos a partir de uma interface programável através da rede. Parâmetros de Input e Output que um Web Service terá de suportar são estão disponíveis através de procedimentos remotos.
- Independente da Plataforma:
  Não estão dependentes de nenhum sistema operativo, linguagem de programação ou qualquer tipo de plataforma. Isto deve-se ao desacoplamento entre o servidor e o cliente. Também não necessitam obrigatoriamente do uso de browsers ou HTML.
- Habilidade de serem síncronos ou assíncronos:
  Interações entre o cliente e o serviço a ser executado são referidas como síncronas ou assíncronas. Em invocações síncronas, o cliente envia o seu pedido e espera por uma resposta sem ser capaz de executar outras operações até que a resposta seja recebida. Em contraste, invocações assíncronas permitem ao cliente pedir um serviço, e imediatamente executar outras operações sem ter de esperar por uma resposta vinda do servidor. A capacidade de Web Services

poderem ser assíncronos é um fator crucial para que o desacoplamento entre sistema e cliente seja possível.

- Interações entre Maquinas e Interoperabilidade:
  O sistema estabelecido por um Web Service, permite suportar interações entre máquinas, através de uma rede, e de uma forma autónoma sem a necessidade de um utilizador. Isto permite que operações sejam realizadas entre maquinas sem que sejam necessários inputs por parte do cliente.
- Aproveitamento da Arquitetura da Web:
  A arquitetura da Web fornece uma forma geral de estabelecer redes interligadas com recursos, que interagem entre si com trocas da representação dos seus estados. Os Web Services utilizam esta arquitetura já estabelecida aplicando-a á sua própria arquitetura.

1.6. Benefícios de Web Services

Também mencionados em [5, 6], temos os benefícios de utilizar Web Services:

- Capacidade de reutilização:
  Um Web Service é um componente ou processo que possam ser acedidos remotamente utilizando mensagens (no caso da World Wide Web, HTTP). Web Services fornecem formas de aceder a código previamente feito através da internet a partir de API´s (Application Programming Interface). Desta forma, as funcionalidades de outros programas podem ser invocadas por diferentes aplicações.
- Interoperabilidade:
  Outros dos benefícios deve-se ao facto dos Web Services permitirem a partilha de dados e da comunicação entre diferentes aplicações independentemente da plataforma em que se baseiam. Por exemplo: aplicações .NET podem interagir com Web Services Java e vice-versa.
- Pesquisa Automática (Registo de Serviços):
  O mecanismo presente nos Web Services de pesquisa automática permite a utilizadores e negócios de encontrarem facilmente o Prestados de Serviços e obter a descrição do serviço que tenha sido previamente publicado. O primei-

ro passo para aceder a qualquer Web Service é fazer a solicitação ao registo de serviços, que retorna ao cliente a descrição dos serviços e o Service Provider que estão de acordo com a descrição dada na solicitação do serviço. Depois desta pesquisa automática ser feita o cliente pode facilmente aceder ao Web Service desejado.

1.7. Comunicação em Web Services

Neste subcapítulo será feita uma descrição de vários conceitos usados para o funcionamento de Web Services, tecnologias como WSDL, SOAP e REST.

1.8. Web Services Description Language (WSDL)

De acordo com o W3C [37], a Web Service Description Language é um formato baseado em XML para descrever serviços de uma rede como um conjunto de pontos de saída que operam sobre mensagens que contenham informação orientada a documentos ou a procedimentos. As operações e mensagens são descritas de uma forma abstrata, e são ligadas a um protocolo de rede e mensagens concreto para definir os pontos de saída. WSDL é extensível para permitir a descrição dos pontos de saída e das suas mensagens independentemente do formato das mensagens ou protocolos de rede utilizados para comunicar.

Por outras palavras, um documento WSDL estabelece os pontos de contacto com o Service Provider. Também contem uma definição padrão sobre esses mesmos pontos de contacto de forma a dar ao Service Consumer a informação necessária para estabelecer comunicação com o Service Provider com o intuito de invocar o serviço desejado.

Num documento WSDL, Service (serviços) são definidos como um conjunto pontos de contacto. Este tipo de definição abstrata de conceitos é independente do seu uso real na criação de redes ou no seu uso em qualquer tipo de formato de dados. Isto permite a reutilização destes conceitos abstratos independentemente da plataforma onde WSDL é utilizado. Os principais elementos num documento WSDL para definir serviços de rede são os seguintes e podem ser observados na figura 25 que demonstra a estrutura de um documento WSDL [37]:

- **Types**: o conteúdo que informa o tipo de definições utilizadas num certo tipo de sistema.
- **Message**: definição abstrata para os dados a serem comunicados.
- **Operation**: descrição de uma ação suportada pelo serviço.
- **Port Type**: um conjunto abstrato de operações suportadas por um ou mais pontos de contacto.
- **Binding**: um protocolo concreto ou especificações de um formato de dados para um Port Type particular.
- **Port**: um ponto de contacto definido por uma combinação de um Binding e de um network address.
- **Service**: um conjunto de pontos de contacto relacionados.

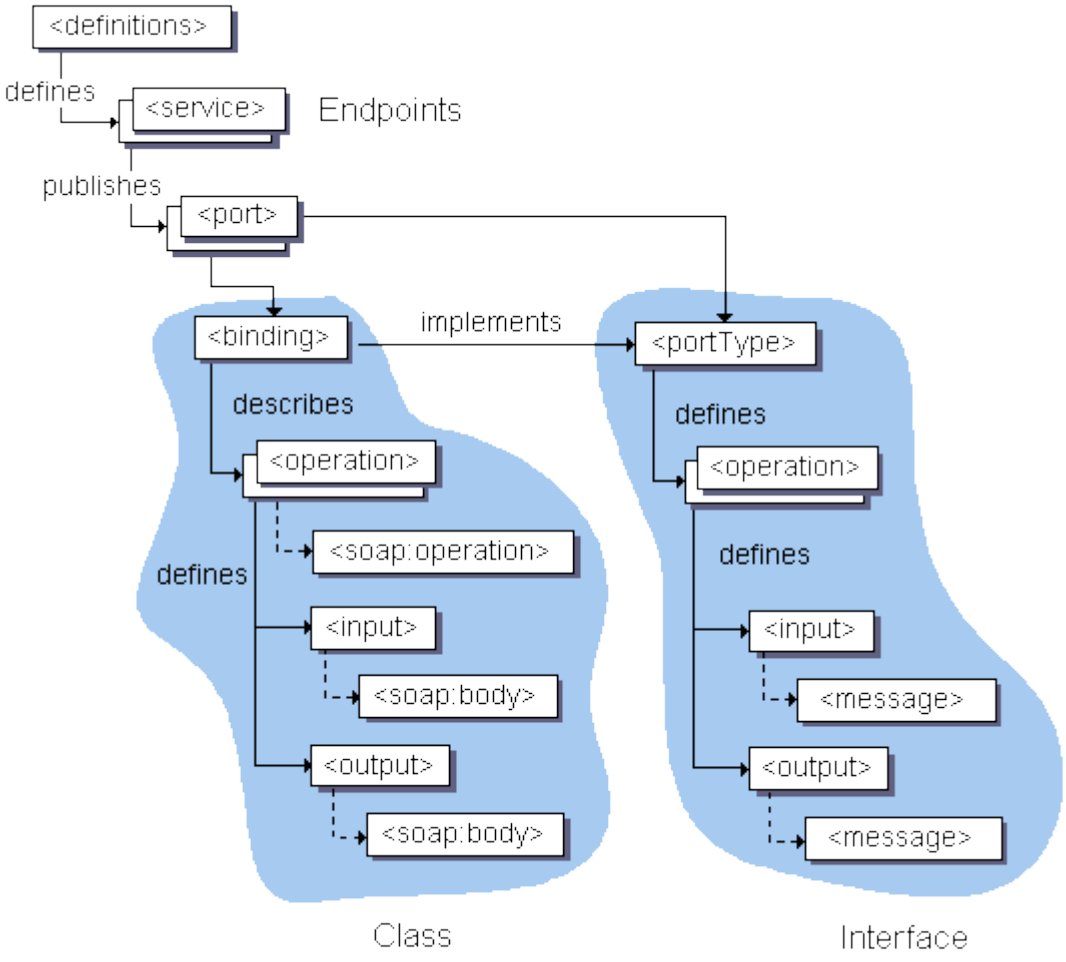

Figura 25 - Estrutura de um documento WSDL [37]

Cada um destes elementos presentes na figura 25, representam um elemento real presente na própria mensagem com WSDL. Na figura 26 temos um exemplo de uma mensagem com os elementos WSDL em que podem ser descritos da seguinte forma [38]:

- definitions: O elemento definitions encaplusa a todo o documento WSDL e normalmente contem várias definições de "namespaces". Como por exemplo:
  <definitions name="HelloService"
  targetNamespace="http://www.examples.com/wsdl/HellpService.wsdl">
  Os namespace contêm elementos ou atributos já definidos e que serão usados na mensagem.
- types: Contêm todas as formas de dados abstratos. Pode conter esquemas XML que definam outros tipos de dados.
- message: A message determina o conteúdo das mensagens a serem enviadas ou recebidas por um Web Service.
- portType: Este elemento descreve várias operações abstratas com mensagens enviadas ou recebidas.
- operation: Possui uma descrição abstrata das ações suportadas por parte do serviço.
- binding: O elemento binding determina o tipo de protocolo de comunicação a ser utilizado para aceder e interagir com WSDL. Ele define onde é que o serviço esta localizado ou para que adereço da rede as mensagens estão a ser enviadas.
- service: Este elemento contem o adereço físico do serviço. Também contem o elemento port que define a localização da informação.

```
<definitions>
   <types>
      definition of types........
   </types>

   <message>
      definition of a message....
   </message>

   <portType>
      <operation>
         definition of a operation.......
      </operation>
   </portType>

   <binding>
      definition of a binding....
   </binding>

   <service>
      definition of a service....
   </service>
</definitions>
```

Figura 26 - Exemplo de um documento WSDL [37]

1.9. SOAP (Simple Object Acess Protocol)

SOAP é um protocolo padrão de mensagens independente de quaisquer protocolos de transporte subjacentes, tais como HTTP, proposto pelo W3C para interagir com Web Services. SOAP também ignora a semântica usada em operações de protocolos subjacentes, usando uma interface baseada em XML. Desta forma, clientes SOAP podem aceder a objetos e métodos que residam em servidores remotos, utilizando um mecanismo padrão. Por exemplo, quando HTTP é usado para transferir uma mensagem em SOAP, esta é encaminhada através de uma operação POST [39].

SOAP envia e recebe mensagens utilizando XML, envolvido por um Header HTTP. Os métodos que podem ser acedidos utilizando serviços SOAP são especificados por uma WSDL. A WSDL de um determinado Web Service consiste numa descrição em XML da sua interface, por outras palavras, descreve os parâmetros, tipos de dados, e tipo de repostas que o Web Service pode devolver. Devido á utilização de um ficheiro WSDL, que é baseado numa linguagem neutra como XML, SOAP torna-se num protocolo de alto nível tornando fácil a distribuição de objetos por dife-

rentes servidores, independentemente da plataforma que em que estes servidores estão construídos [42].

1.10. Estrutura de mensagens SOAP

Neste subcapítulo será descrita a estrutura de uma mensagem SOAP juntamente com um exemplo de uma mensagem. Uma mensagem SOAP, contem dois subelementos dentro da estrutura principal chamada Envelope, nomeadamente o Header e o Body, tal como pode ser observado na figura 27. O conteúdo destes elementos são definidos por parte da aplicação e não como para das especificações SOAP. SOAP simplesmente estabelece o protocolo de como estes elementos devem atuar[40].

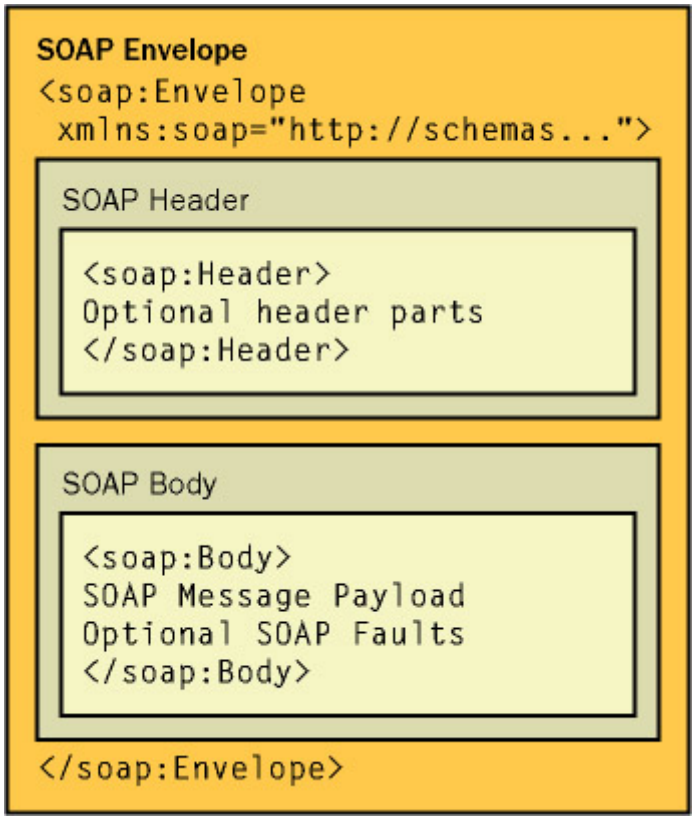

Figura 27 - Estrutura de uma mensagem SOAP [44]

- Header: é um elemento opcional que define um conjunto de atributos opcionais da mensagem. O seu propósito é o chamamento de extensões separadamente do conteúdo da mensagem sem alterar a estrutura fundamental do SOAP. Graças a esta separação, funcionalidades extras podem ser adiciona-

das, tais como segurança, transações, atributos de qualidade de serviço (QoS) sem modificar as especificações da mensagem [33].

- Body: é um elemento obrigatório que encapsula a mensagem a enviar num formato XML tal como pode ser visto na figura 28. Este elemento possui os dados requeridos (resposta) ou uma mensagem de erro (encapsulada dentro de um subelemento "fault"). Os dados requeridos representam a troca de dados especifica á aplicação com o Web Service. Estes dados podem estar num formato XML ou como parâmetros no chamamento de um método. Dentro do Body, o método de chamamento de dados tal como os argumentos relacionados com o método são definidos, a resposta ao método de chamamento é guardada tal como qualquer informação de erro [33].

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"
      xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">
      http://tempuri.org/IService/MyOperation
    </Action>
    <ActivityId CorrelationId="7224e2a9-8f9c-4acb-a924-17cb6af67b23"
xmlns="http://schemas.microsoft.com/2004/09/ServiceModel/Diagnostics">
      43ffa660-a0c6-4249-bb36-648b73a06213</ActivityId>
  </s:Header>
  <s:Body>
    <MyOperation xmlns="http://tempuri.org">
      <MyValue>Some Value</MyValue>
    </MyOperation>
  </s:Body>
</s:Envelope>
```

Figura 28 - Exemplo de uma mensagem SOAP [43]

Dois objetivos principais na construção do protocolo SOAP, são a simplicidade e extensibilidade. SOAP tenta realizar estes objetivos omitindo, por parte da framework das mensagens, funcionalidades que são normalmente encontradas em sistemas distribuídos. Tais funções incluem mas não estão limitas a "qualidade", "segurança", "correlação", "encaminhamento", e "Padrões de Troca de Mensagens". Outras funcionalidades estão abertas a ser definidas através de extensões [41].

1.11. REST (Representational State Transfer)

Representational State Transfer (REST) é um estilo de arquitetura baseados nos princípios que descrevem como os recursos de uma rede devem ser definidos e como se deve lidar com os mesmos recursos. Estes princípios foram descritos pela primeira vez em 2000 por Roy Fielding, um dos principais autores das especificações para Hypertext Transfer Protocol (HTTP) [17], como parte da sua dissertação de doctoramento, que pode ser encontrada em [16]. REST é uma alternativa a SOAP na implementação de um Web Service.

1.12. Arquitetura de REST

É importante ter em consideração que REST é um estilo de arquitetura ao contrario de um conjunto de padrões a seguir ou protocolos. Desta forma aplicações que utilizem uma arquitetura REST são denominadas por RESTful ou Rest-styled [16].

No caso de uma arquitetura RESTful, as interações servidor – cliente necessitam de ser stateless (sem estados), isto é, cada pedido do cliente feito ao servidor deve conter todas as informações necessárias para que o servidor perceba o pedido (figura 29). O pedido feito pelo cliente não pode invocar qualquer tipo de função guardada no servidor, mantendo o estado da sessão inteiramente no cliente [45].

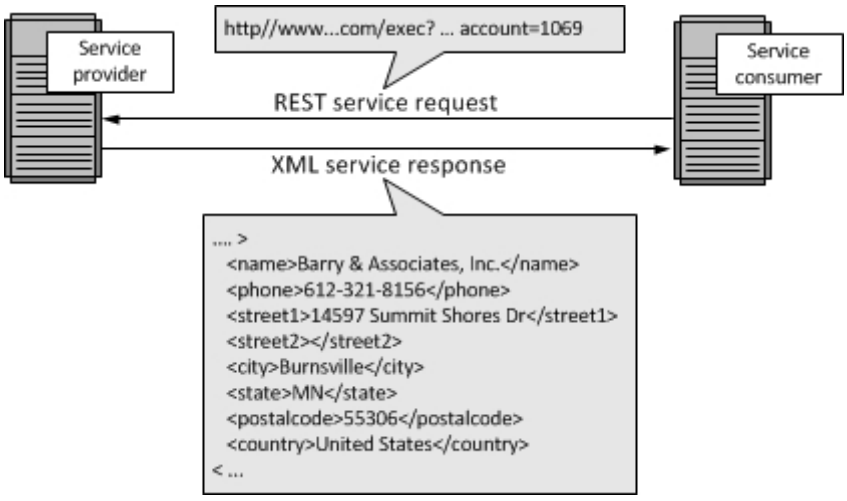

Figura 29 - Uso de REST para Web Services [16]

Tendo em conta que do lado servidor estados não necessitam de ser guardados entre pedidos, podemos rapidamente concluir que arquiteturas REST permitem ao servidor ficar com recursos livres, dado que o servidor não necessita que gerir recursos entre pedidos. Isto torna o sistema mais facilmente escalável visto que os recursos são geridos por cada cliente individual.

No entanto, o facto de o sistema ser stateless pode causar certas desvantagens. Diminuição da prestação da rede pode ser um problema no caso de numa serie de pedidos, o mesmo tipo de dados seja enviado [45]. Um exemplo disto pode ser observado também na figura 29, caso o cliente envie um pedido para saber o número de telefone da conta em questão, e de seguida envie outro pedido a requisitar a cidade da mesma conta, fazendo com que o cliente receba duas mensagens idênticas ambas com toda a informação desejada por parte do cliente. Para solucionar este possível problema na arquitetura REST, uma cache (como uma base de dados) é feita do lado do cliente que guarda qualquer tipo de dados marcados como cacheable. Esta marcação de cacheable ou não-cacheable é feita por parte do servidor. Desta forma o cliente reutiliza os dados para pedidos equivalentes, libertando recursos da rede e melhorando a latência para todos os clientes.

Para além de um problema possível na rede, devido aos estados da aplicação estarem localizados do lado do cliente, isto reduz o controlo por parte do servidor sobre o estado da aplicação [45].

1.13. Atributos de REST

O estilo de arquitetura REST pode ser caracterizado por:

- O protocolo é cliente – servidor, stateless, por camadas e capaz de suportar caching.
- Todos os recursos são unicamente acedidos e manipulados utilizando uma forma uniforme e mínima de comandos fixada a volta de operações create, read, update e delete (tipicamente comandos HTTP: GET, POST, PUT ou DELETE)
- A identificação dos recursos é feita através de um identificador uniforme de recursos (ex: URL).

- O conteúdo dos recursos pode ser apresentado em vários formatos (HTM, XML, texto normal, PDF ou JPEG).

Interação do estado através de hyperlinks. Como por exemplo manipulação de URL ou cookies