Pós-Graduação em Ciência da Computação

“Web service RESTful para Manipulação, Catalogação, Publicação na Web e Eventual Manutenção de Dados Abertos Governamentais”

Por

Bruno Iran Ferreira Maciel

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

RECIFE

2014

Universidade Federal de Pernambuco
Centro de Informática
Pós-graduação em Ciência da Computação

Bruno Iran Ferreira Maciel

"Web service RESTful para Manipulação, Catalogação, Publicação na Web e Eventual Manutenção de Dados Abertos Governamentais"

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: *Profa. Bernadette Farias Lóscio*

RECIFE

2014

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

M152w Maciel, Bruno Iran Ferreira
Web service RESTful para manipulação, catalogação, publicação na Web e eventual manutenção de dados abertos governamentais / Bruno Iran Ferreira Maciel – Recife: O Autor, 2014.
104 f.: il., fig., tab.

Orientadora: Bernadette Farias Lóscio.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2014.
Inclui referências e apêndice.

1. Banco de dados. 2. Web semântica. I. Lóscio, Bernadette Farias (orientadora). II. Título.

025.04 CDD (23. ed.) UFPE- MEI 2015-188

Dissertação de Mestrado apresentada por ***Bruno Iran Ferreira Maciel*** *à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "****Web service RESTful para manipulação, catalogação, publicação na Web e eventual manutenção de Dados Abertos Governamentais****" orientada pela Profa.* ***Bernadette Farias Lóscio*** *e aprovada pela Banca Examinadora formada pelos professores:*

Prof. Kiev Santos Gama
Centro de Informática/ UFPE

Profa. Damires Yluska de Souza Fernandes
Gerência Educacional de Informática / IFPB

Profa. Bernadette Farias Lóscio
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 2 de Setembro de 2014

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco.

“O mundo não é um arco-íris e um amanhecer, na verdade é um lugar ruim e asqueroso. E não importa o quão durão você seja, apanhará e ficará de joelhos, se assim permitir. Nem você, nem ninguém baterão tão forte quanto a vida. Porém, não se trata de quão forte pode se bater, se trata de quão forte se pode ser atingido e continuar seguindo em frente. É assim que a vitória é conquistada”.

Sylvester Stallone - Rocky Balboa

Agradecimentos

Agradeço primeiramente a Deus que está sempre do meu lado. Aos meus pais Márcia e Ronaldo, minhas irmãs Jackeline, Roberta e meu irmão Fábio, pessoas que me apoiaram e ajudaram a realizar meus sonhos e me ensinaram a lutar pelos meus objetivos. Sempre serei grato a todos vocês. Principalmente a minha mãe que tanto tem me apoiado em todos os momentos e minha irmã Jackeline juntamente com seu esposo Humberto Silva que sempre se mostraram solidários e dispostos a me ajudar em momentos recentes de dificuldades. A todos os meus amigos pelo companheirismo e pelas horas de descontração (raríssimas fora do CIn). Sempre me lembrarei de todos vocês. Obrigado a todos.

A todos do laboratório da pós-graduação "APG", em especial: *a*) Eunice Palmeira da Silva; *b*) Glaucia Melissa Medeiros Campos; *c*) Alixandre Thiago Ferreira de Santana; *d*) Myller Claudino de Freitas; *e*) Aline Chagas Rodrigues Marques; *f*) Sidartha Azevedo Lobo de Carvalho; e *g*) Carla Cristina Braz de Oliveira. Aos do grupo de pesquisa: *a*) Danusa Ribeiro Bezerra da Cunha; *b*) Walter Travassos Sarinho; *c*) Webber de Souza Fantini; e *d*) Márcio Angelo Bezerra de Lira, meus sinceros agradecimentos pela amizade, pelo agradável convívio, pelas discussões e troca de experiências enriquecedoras. Em especial, as minhas amigas Danusa Ribeiro Bezerra da Cunha e Aline Chagas Rodrigues Marques, assim como meu amigo Webber de Souza Fantini pelas valiosas contribuições durante o desenvolvimento e conclusão do trabalho. Ademais, sou eternamente grato a Emily, por seu tamanho apoio, amizade e atenção.

Aos professores, funcionários e estudantes da Universidade Federal de Pernambuco, Departamento de Pós-Graduação em Ciência da Computação, por terem me recebido com muita alegria e felicidade. Em especial os professores Roberto Souto Maior de Barros e Fernando da Fonseca de Souza.

A minha orientadora, Profa. Dra. Bernadette Farias Lóscio, por todo seu caráter, profissionalismo, amizade, paciência e dedicação comigo durante as diversas atividades de pesquisa e andamento deste trabalho.

Aos integrantes da banca Profa. Dra. Damires Yluska de Souza Fernandes e o Prof. PhD Kiev Santos Gama por se prestarem como avaliadores do trabalho. Meus sinceros agradecimentos aos dois.

Por fim, agradeço ao CIn-UFPE pela oportunidade e FACEPE pelo apoio financeiro que viabilizou o trabalho. Também sou muito grato a todos os colaboradores do NUTES-UFPE pelo carinho, amizade e oportunidade oferecida.

Resumo

Recentemente, diversas iniciativas surgiram com intuito de fortalecer a transparência da administração pública por meio da publicação de dados na *Web*, conhecida por Dados Abertos Governamentais. Esse tipo de iniciativa permite que os cidadãos permaneçam informados sobre a gestão do governo, permitindo acompanhar a administração pública, de modo a contribuir com o aumento do poder de fiscalização do cidadão.

A publicação de Dados Abertos é geralmente realizada por meio de sistemas para catalogação de dados, tais como: a Plataforma *Comprehensive Knowledge Archive Network* (CKAN). Além dos catálogos de dados, as *Application Programming Interface* (APIs) e *Web services* também podem ser utilizados para facilitar o acesso aos dados. Apesar do grande número de iniciativas que visam fornecer dados na *Web*, ainda não há um consenso sobre a melhor maneira de cumprir esta tarefa.

No entanto, independentemente da estratégia escolhida, é importante que os processos utilizados para publicar dados na *Web* e manutenção destes dados se realizem de forma automática e de modo que cada uma das suas atividades possam ser reproduzidas quando for necessário. Neste contexto, este trabalho propõe uma abordagem *Restful* API, com base em um conjunto de *Web services* para facilitar a publicação e o consumo de dados abertos.

Palavras-chave: SERVIÇO. DADOS ABERTOS. PUBLICAÇÃO

Abstract

Recently, several initiatives have emerged with the aim of strengthening the transparency of public administration by publishing data on the Web, known as Government Open Data. This type of initiative allows citizens to remain informed about the government's management, allowing monitoring the public administration in order to contribute to the increase of citizen oversight power.

Data Open Publication is generally carried out by means of systems for cataloging data such as: a Comprehensive Knowledge Archive Network Platform (CKAN). In addition to the data catalogs, Application Programming Interfaces (APIs) and Web services can also be used to facilitate access to the data. Despite the large number of initiatives aimed at providing data on the Web, there is still no consensus on the best way to accomplish this task.

However, regardless of the chosen strategy, it is important that the processes used to publish data to the web and maintenance of these data are carried out automatically and so that each of its activities can be reproduced when necessary. In this context, this paper proposes a Restful API approach, based on a set of Web services to facilitate the publication and consumption of open data.

Keywords: SERVICE. OPEN DATA. PUBLICATION

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AJAX *Asynchronous JavaScript and XML*

API *Application Programming Interface*

CKAN *Comprehensive Knowledge Archive Network*

CORBA *Common Object Request Broker Architecture*

CSV *Comma-Separated Values*

CSS *Cascading Style Sheets*

DAG Dados Abertos Governamentais

DCAT *Data Catalog Vocabulary*

DCOM *Distributed Component Object Model*

DTD *Document Type Definition*

ETL *Extract, Transformation and Loading*

HATEOAS *Hypertext As The Engine Of Application State*

IANA *Internet Assigned Numbers Authority*

IBM *International Business Machines*

IEEE *Institute of Electrical and Electronics Engineers*

HMAC *Keyed-Hashing for Message Authentication*

HTML *HyperText Markup Language*

HTTP *HyperText Transfer Protocol*

ISO *International Standardization Organization*

JSON *JavaScript Object Notation*

MIME *Multipurpose Internet Mail Extensions*

OKFN *Open Knowledge Foundation*

PDF *Portable Document Format*

OLTP *Online Transaction Processing*

PHP *Hypertext Preprocessor*

REST *Representational State Transfer*

RFC *Request for Comments*

RH Recursos Humanos

RUP *Rational Unified Process*

SEI *Software Engineering Institute*

SGBD Sistemas Gerenciadores de Bancos de Dados

SGBDOR Sistema Gerenciador de Banco de Dados Objeto Relacional

SHA1 *Secure Hash Algorithm 1*

SOA *Service-Oriented Architecture*

SOAP *Simple Object Access Protocol*

SQL *Structured Query Language*

SPARQL *Protocol and RDF Query Language*

URI *Uniform Resource Identifier*

URL *Uniform Resource Locator*

W3C *World Wide Web Consortium*

WSDL *Web Services Description Language*

XML *Extensible Markup Language*

WWW *World Wide Web*

Sumário

1 Introdução

Prefiro me arriscar fazendo algo que gosto perdidamente que se sentir perdido por não fazer o que amo.

—SYLVESTER STALLONE

Este capítulo contextualiza o foco desta dissertação e começa por apresentar a sua motivação, problema de pesquisa e seus objetivos. Em seguida, são tratados alguns aspectos relacionados, mas que não são diretamente abordados por esta pesquisa. Por fim, é apresentada a estrutura do restante deste trabalho.

1.1 Motivação

Em geral, governos possuem uma grande quantidade de dados para uso em suas operações e prestação de serviços. Historicamente, estes dados foram de caráter sigilosos e rigorosamente controlados apenas pela gestão pública. No entanto, com o passar do tempo, passaram a ser gradativamente disponibilizados para a sociedade. Entre as formas de distribuição estão documentos impressos em papel e, posteriormente por meio de formatos digitais.

Contudo, alguns destes formatos digitais adotados para disponibilização dos dados não permite que o mesmo seja livremente utilizado, reutilizado e redistribuído por qualquer pessoa ou agente de software. Logo, foi necessário propor características aos formatos digitais a fim de permitir sua livre utilização. Neste sentido, iniciativas governamentais que buscavam fortalecer a transparência da administração pública passaram a adotar formatos de Dados Abertos para publicação de dados.

De acordo com The Open Definition (2014), os Dados Abertos podem ser entendidos como dados que possuem disponibilidade de acesso, permitem ser reutilizados, redistribuídos por qualquer pessoa ou agente de software e sem restringir licença. As características contribuem para uma participação universal e novos conjuntos de dados. Dentre as diferentes notações (para escrita dos dados) capazes de atender às particularidades (dos Dados Abertos), estão os documentos em notação *Comma-Separated Values* (CSV), *JavaScript Object Notation* (JSON) e *Extensible Markup Language* (XML) que comumente são utilizados para transferência de dados por meio de *softwares*.

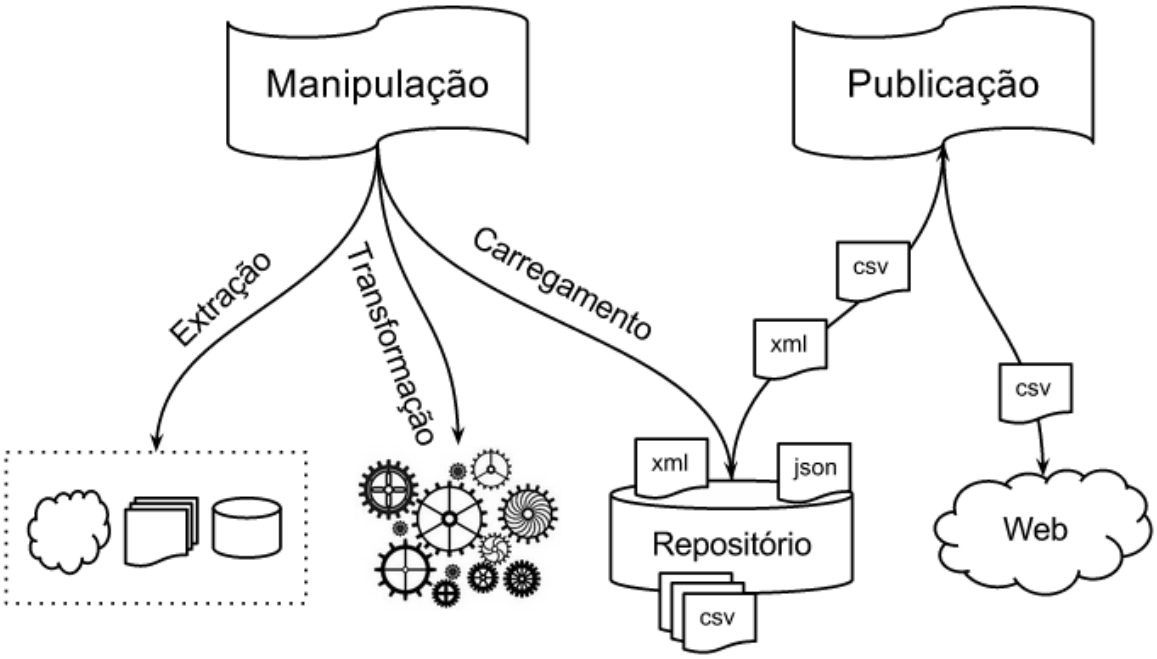
Quando a população em geral, tem acesso aos dados em formato aberto, dentre outras

vantagens, os cidadãos passam a ter um novo recurso para acompanhar a gestão pública. Contribuindo diretamente para o fortalecimento do poder de fiscalização dos cidadãos. Além disso, novos conjuntos de dados podem ser construídos por meio da combinação de diferentes fonte de dados. Estes Dados Abertos, quando disponibilizados pelos governos, também são conhecidos como Dados Abertos Governamentais (DAG).

Acredita-se que, a partir de iniciativas que colaborem com a publicação de DAGs, seja possível ampliar a participação dos cidadãos na gestão pública. Os DAGs permitem sua leitura, reutilização, análises e visualizações, assim com a combinação e cruzamento com outros dados (respeitam os princípios dos Dados Abertos). Contudo, sua criação e publicação na *Web*, exigem a execução de diferentes atividades.

A Figura 1, apresenta as duas principais atividades dentro do processo para publicação dos dados na *Web*, são elas: manipulação e publicação na *Web*. Fazem parte da atividade de manipulação as tarefas de extração, transformação e carga. Na atividade de publicação é possível mencionar a tarefa de catalogação, dentre outras.

Figura 1 Atividades de publicação de dados



Fonte: o autor

O processo de criação dos conjuntos de dados (extração das fontes de dados e disponibilização na *Web*) não é tarefa trivial. Alguns fatores pertinentes nesta atividade dificultam a criação dos conjuntos de dados, assim como a reutilização do mesmo processo (atividades realizadas).

Os dados, rotineiramente são extraídos em bases de dados gerenciadas por **SGDB!**s (**SGDB!**s). Logo, depois da seleção dos dados, consultas do tipo *Structured Query Language* (SQL) são construídas e executadas, a fim de criar os arquivos contendo os conjuntos de dados. Após o processo de criação do arquivo é necessário disponibilizar o conjunto de dados na *Web* devidamente anotado e descrito. Geralmente são utilizados sistemas independentes para "criação do conjunto de dados" e "publicação na *Web*", tornando o processo mais oneroso (demorado).

A publicação dos conjuntos de DAGs geralmente é feita por meio de sistemas que permitem a catalogação dos dados, como por exemplo o *software* de código aberto *Comprehensive Knowledge Archive Network* (CKAN). Durante o processo de catalogação, os conjuntos de dados são importados e hospedados de forma estática, tornando a manutenção mais custosa devido às diferentes atividades envolvidas.

Nesse contexto (independência de atividades), garantir que o conteúdo publicado esteja de acordo com os dados de origem, mesmo quando estes sofrerem atualizações, exige retrabalho. Nos casos em que a manutenção é realizada de forma manual, isto implicará na alocação de recurso humano e mais trabalho para produzir os DAGs.

Os DAGs estão disponíveis para os diferentes integrantes da sociedade, dentre eles estão os desenvolvedores de *softwares*. Estes por sua vez, são responsáveis também por produzir soluções conhecidas como *Web Mashups*, algumas das característica apresentadas por estas aplicações são: possuem finalidade específica, apresentam como conteúdo a combinação de diferentes dados, podendo ter como origem diferentes fontes de dados provenientes de *Application Programming Interface* (API), *Web services*, documentos e outras.

O uso de APIs ou *Web services* contribuem com a integração entre aplicações de forma interoperável. Desenvolvedores e cidadãos podem se beneficiar destas formas de distribuição dos dados. Estas maneiras de acesso aos dados permite interações processáveis por máquina de maneira automática.

Apesar da grande quantidade de iniciativas que buscam disponibilizar dados abertos, ainda não existe um consenso sobre a melhor forma de realização desta tarefa. Este trabalho tem como motivação automatizar a execução das atividades que envolvem o processo de publicação e facilitar o uso dos dados abertos.

Desta forma, será apresentada, na seção seguinte, a pergunta de pesquisa do presente trabalho, que trata exatamente do problema relacionado à abordagem utilizada para publicação e uso de dados abertos governamentais.

1.2 Problema de Pesquisa

Diante do exposto, torna-se perceptível a necessidade de explorar melhores abordagens e processos de software capazes de facilitar e automatizar atividades de publicação e uso de dados abertos governamentais.

Sendo assim, o seguinte problema de pesquisa foi formulado: **Como fornecer uma abordagem capaz de facilitar as atividades de publicação e uso dados abertos governamentais?**

1.3 Objetivos

Para responder à pergunta de pesquisa, foram definidos os seguintes objetivos, divididos em geral e específicos:

1.3.1 Geral

Propor uma abordagem baseada em serviços *Web* para integrar as atividades do processo de publicação e uso de dados abertos, com o intuito de automatizá-las e facilitar sua execução. Desta forma, acredita-se que será possível a construção de novas aplicações baseadas nestes serviços *Web*, além de ser possível integrar esta abordagem a outras estratégias de publicação e uso de dados abertos.

1.3.2 Específicos

Para alcançar o objetivo geral, foram estabelecidos alguns objetivos específicos para esta dissertação. São eles:

– Análise do processo de publicação e uso de dados aberto.
– Definição de uma abordagem baseada em serviços *Web* para a publicação e o uso de dados abertos.
– Especificação de uma arquitetura baseada em *Representational State Transfer* (REST) para prover os serviços *Web*.

A principal *contribuição* deste trabalho foi a definição de uma abordagem proposta baseada em serviços *Web*, além da implementação do protótipo para a abordagem sendo possível fornecer os serviços descritos. Este protótipo pode ser utilizado como componente em sistemas que proveem a integração de fontes de dados na *Web*.

1.4 Organização da Dissertação

Esta dissertação é composta por seis capítulos, como descrito a seguir. A sequência dos capítulos e estruturas dos conteúdos de cada capítulo procura refletir o processo adotado durante o andamento deste trabalho.

Este capítulo introduziu a dissertação e sua finalidade. Ainda neste capítulo, o problema de pesquisa foi apresentado, bem como os objetivos gerais e específicos para responder à pergunta de pesquisa. Em resumo, este capítulo identifica o problema de pesquisa, o raciocínio por trás da mesma e a estrutura do trabalho.

O restante do trabalho está organizado da seguinte forma: O Capítulo 2 apresenta os conceitos fundamentais para o entendimento deste trabalho; O Capítulo 3 descreve a abordagem proposta para publicação e uso de dados abertos; O Capítulo 4 apresenta detalhes técnicos da implementação da abordagem como tecnologias, padrões, escolhas e definições adotadas para o desenvolvimento da abordagem; O Capítulo 5 descreve a estratégia prática por meio de protótipos (lado do cliente e do lado do servidor) para simular e avaliar o uso da abordagem e, finalmente, o Capítulo 6 apresenta as conclusões obtidas a partir da fundamentação teórica, desenvolvimento da proposta e cenário estudado. Também são apresentadas sugestões e recomendações para trabalhos futuros.

2

Fundamentação Teórica

Ninguém vai bater mais forte que a vida. Não importa como você bate e sim o quanto aguenta apanhar e continuar lutando, o quanto pode suportar e seguir em frente. É assim que se ganha.

—SYLVESTER STALLONE - ROCKY BALBOA

A seguir, serão abordados, com apoio da literatura, os principais temas que formam a base teórica deste trabalho, a saber: Dados Abertos, *Web Service*, processo de *Extract, Transformation and Loading* (ETL). Por fim, serão apresentados os trabalhos relacionados e as conclusões deste capítulo.

2.1 Dados Abertos

Mpinda, Bungama e Maschietto (2015)

Do ponto de vista conceitual, o termo "dado" representa o mais baixo nível de abstração a partir do qual as informações e, em seguida, o conhecimento são derivados. Segundo Pipino, Lee e Wang (2002, p. 2), muitas vezes o termo é utilizado como sinônimo de informação, mas dado pode ser entendido como fatos sobre entidades que podem ter ou não significado para o manipulador. O termo "Dados Abertos", por sua vez, possui definição de dados que possam ser livremente usados, reutilizados e redistribuídos por qualquer pessoa ou agente de *software* (The Open Definition, 2014).

O conceito de Dados Abertos também foi introduzido no setor público, dando início ao chamado Dados Abertos Governamentais, descrito posteriormente neste documento. Em geral, a definição de "**Dados Abertos**" é aplicado tanto aos dados em sua forma bruta (diretamente da fonte de dados de origem) como processada (após realizar operações com os dados para agregar valor), mesmo eles não oferecendo uma visão completa sobre os dados. Isso significa dizer que o mais importante está em suas possibilidades de abertura e reutilização. As principais características que definem os Dados Abertos são apresentadas na Tabela 1 com detalhes sobre elas The Open Definition (2014).

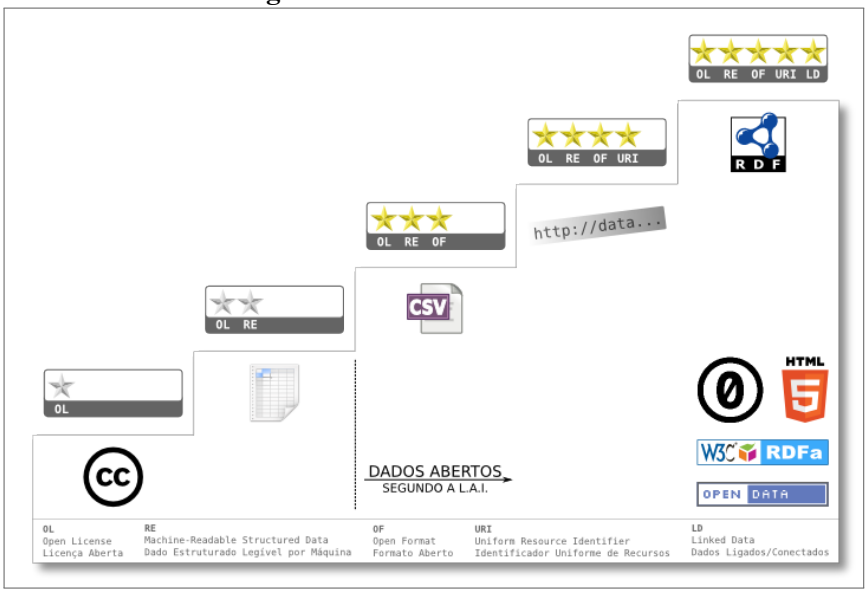
Tabela 1 Principais Características dos Dados Abertos

Característica	Detalhe
Disponibilidade e Acesso	Os dados devem estar disponíveis como um todo e, de preferência, por meio de cópia do documento na *Internet* (*download*). Os dados também devem estar disponíveis em uma forma conveniente e modificável.
Reuso e Redistribuição	Os dados devem ser fornecidos de maneira que permitam a sua reutilização e redistribuição, incluindo a combinação com outros conjuntos de dados.
Participação Universal	Todos devem ser capazes de usar, reutilizar e redistribuir. Não deve haver discriminação contra os campos de atuação ou contra pessoas ou grupos. Por exemplo, as restrições "não-comerciais" que impediriam o uso "comercial", ou restrições de utilização para determinados fins (por exemplo, só na educação) não são permitidos.

Fonte: *The Open Definition (2014)*

O modelo cinco estrelas elaborado por Berners-Lee (2006) atribui características aos conjuntos de dados quanto à intenção de realizar comparações ou integração dos dados. São definidos cinco níveis de intenções, e a cada nível é atribuído uma estrela. A Figura 2 apresenta adaptação realizada por The Open Definition (2014) da versão original de *Tim Berners-Lee*[1]. Quanto mais estrelas, mais poderosa a publicação fica e mais fácil de as pessoas utilizarem e integrarem com outros conjuntos de dados.

Figura 2 Modelo Cinco Estrelas



Fonte: (OKFN.BR, 2013)

De acordo com Berners-Lee (2006), no modelo cinco estrelas, os dados abertos são avaliados na visão (intenção) da sua evolução em relação ao uso correto dos padrões e sua relevância em relação à integração com *Linked Data*[2]. O modelo não é exclusivo para os Dados Abertos.

[1]<www.w3.org/People/Berners-Lee/>. Acesso: 01/10/2014
[2]<http://linkeddata.org/>. 28/11/2014

De acordo com OKFN.BR (2013), a Figura 2 pode ser explicada levando em consideração o número de estrelas. A Tabela 2 apresenta um detalhamento em relação à quantidade de estrelas e características sobre os níveis "Estrelas".

Tabela 2 Detalhes sobre o modelo cinco estrelas

Número de Estrela(s)	Detalhe
1	Os dados estão disponíveis na *Web*, independente de formato, sob uma a licença aberta (Por exemplo um documento *Portable Document Format* (PDF) (Formato de Documento Portátil) sob uma licença aberta).
2	Além da condição anterior, consiste também a disponibilização como dado estruturado legível por máquina (Por exemplo um arquivo *Excel* ao invés de uma imagem escaneada de uma tabela).
3	Todas as anteriores, juntamente com a possibilidade de utilizar um formato não proprietário (Por exemplo: um arquivo CSV ao invés de um *Excel*).
4	Todas as anteriores, juntamente com a possibilidade de utilizar *Uniform Resource Identifier* (URI) bem desenhadas para identificar os dados, então as pessoas podem referenciá-los.
5	Todas as anteriores e ainda a vinculação de seus dados com o de outras pessoas para prover contexto.

Fonte: OKFN.BR (2013)

As subseções a seguir fornecem definições sobre Dados Governamentais e Dados Abertos Governamentais.

2.2 Dados Abertos Governamentais

Os governos, em geral, produzem grandes quantidades de dados, sejam estes dados administrativos (internos) ou de serviços públicos prestados à sociedade. Historicamente, estes dados eram rigorosamente controlados apenas pela gestão pública e com o passar do tempo passaram a ser gradativamente liberados. Dentre as formas de liberação, estão os documentos impressos em papel.

Segundo SILVA (2010, p. 8-9), no dia 21 de Janeiro de 2009, o presidente dos Estados Unidos da América *Barack Obama* direcionou um memorando intitulado *Transparency and Open Government* (Transparência e Governo Aberto) a todos os chefes de departamentos executivos e agências. O memorando trata do comprometimento da atual gestão do governo em criar níveis sem precedentes de abertura dos dados governamentais. Ele define três dimensões de abertura a serem exploradas: transparência, participação e colaboração. A Tabela 3 descreve estas características.

Tabela 3 Três Dimensões da Abertura a Serem Exploradas

Dimensão	Detalhe
Transparência	A transparência promove a responsabilização e fornece informações para os cidadãos sobre o que o governo está fazendo.
Participação	A participação fornece engajamento público e reforça a eficácia do governo em melhorar a qualidade das suas decisões. O conhecimento é disperso na sociedade, e as autoridades públicas se beneficiam de ter acesso a esse conhecimento disperso.
Colaboração	A colaboração deve envolver ativamente os cidadãos na gestão do governo.

Fonte: adaptado de SILVA (2010, p. 10-11)

Um conjunto de políticas foi criado em resposta ao memorando (SILVA, 2010, p. 8-9). Estas políticas (iniciativas) foram chamadas de *Open Government Initiative*[3] (Iniciativas de Governo Aberto).

Segundo Araujo *et al.* (2012), é comum encontrar o termo Dados Abertos Governamentais e Governo Aberto como sinônimos, mas Governo Aberto é um conceito mais amplo. O termo governo aberto faz referência à disponibilização de todas as informações (em qualquer formato), que sejam da responsabilidade de um governo. Não implicando, para isso, o uso de tecnologias ou formatos predefinidos.

Os passos a serem seguidos para publicar dados abertos governamentais podem ser agrupados em três (DINIZ, 2009). São eles:

– Selecionar os dados que serão disponibilizados e identificar quem os controla;
– Representar esses dados de uma maneira que as pessoas possam reutilizá-los;
– Publicar os dados e divulgar. A escolha dos conjuntos de dados a serem disponibilizados é uma questão política.

Publicar dados na *Web* sempre foi possível, desde o início da *Internet*, seja por meio de uma página *Web* ou arquivo disponível para cópia (*download*). No entanto, a publicação de Dados Abertos pressupõe que determinadas características sejam respeitadas para garantir que eles possam ser acessados, reutilizados e redistribuídos.

A seguir, são apresentadas as características que definem os Dados Abertos Governamentais e os passos para a disponibilização (publicação) de dados, de modo a preservar essas características. Esses atributos também são válidos para definir e especificar os Dados Abertos. Estas características podem ser vistas na Tabela 4.

[3]<http://www.whitehouse.gov/Open/>. Acesso: 01/10/2014

Tabela 4 Características dos Dados Abertos Governamentais

Característica	**Detalhe**
Interoperabilidade	Ser independente de plataformas tecnológicas e basear-se em formatos padronizados.
Uso de Padrões	A garantia de evolução e contínua melhoria da representação dos dados está nas tecnologias sustentadas por organismos internacionais de padrões como *World Wide Web Consortium* (W3C) e *International Standardization Organization* (ISO).
Independente de Aplicação	Os dados devem estar desvinculados das ferramentas, relatórios ou páginas *Web* que os originaram.
Notação	O formato utilizado para representação dos dados deve, preferencialmente, permitir a manipulação destes por máquinas. No entanto, os dados deverão estar estruturados. Uma boa estruturação permite que terceiros possam fazer uso automatizado dos dados. Formatos que somente podem ser vistos, e não extraídos, não são úteis nesse contexto e devem ser evitados.
Metadados	Cada conjunto de dados deverá possuir uma descrição externa de si próprio (*metadados*) de tal forma que seja identificada a sua natureza, conheça-se a sua origem e seja possível uma análise dos dados por meio de um conjunto de instruções legíveis por máquina que descreve os dados e suas relações.
Semântica	Sempre que possível, insira conteúdos semânticos no código da página *web* onde os dados estão disponíveis. Além de facilitar a leitura dos dados por outras máquinas, os mecanismos de buscas, como por exemplo *Google* ou *Yahoo*, encontrarão os dados mais facilmente.
Organização do Conteúdo	Se a opção de disponibilização de dados for por meio de uma interface de programação de aplicativos, separe os dados da interface.
URI Amigável	Sempre que possível, crie URI para cada objeto dos seus dados (por exemplo, equipamentos públicos, autoridades, órgãos). A URI é um padrão de codificação para fornecer uma representação alfanumérica universal e sem ambiguidade para cada objeto, de maneira independente da plataforma de *software* e do idioma.

Fonte: adaptado de DINIZ (2009, p. 6)

A publicação e divulgação dos dados deverá ser feita no ambiente da rede mundial de computadores (*Internet*). Deve-se informar claramente o uso de catálogo de dados, bem como os seus respectivos *metadados*. Deve-se dar preferência aos conjuntos de dados em diferentes formatos para publicação. Sendo assim, o alcance e as alternativas para os usuários terem acesso aos dados são maiores. Indicações de como publicar estes dados na *Web* podem ser encontradas em DINIZ (2009, p. 10).

Os dados abertos também são pautados pelas três leis e oito princípios[4]. As três leis são apresentadas na Tabela 5 e os oito princípios na Tabela 6. As duas tabelas são complementares.

Tabela 5 As Três Leis dos Dados Abertos Governamentais

Lei	Detalhe
1	Se o dado não pode ser encontrado e indexado na *Web*, ele não existe.
2	Se não estiver aberto e disponível em formato compreensível por máquina, ele não pode ser reaproveitado.
3	Se algum dispositivo legal não permitir sua replicação, ele não é útil.

Fonte: adaptado de PORTAL BRASILEIRO DE DADOS ABERTOS (2014)

Tabela 6 Oito Princípios dos Dados Abertos Governamentais

Princípio	Detalhe
Completos	Todos os dados públicos são disponibilizados. Dados são informações eletronicamente gravadas, incluindo, mas não se limitando a documentos, bancos de dados, transcrições e gravações audiovisuais. Dados públicos são dados que não estão sujeitos a limitações válidas de privacidade, segurança ou controle de acesso, reguladas por estatutos.
Primários	Os dados são publicados na forma coletada na fonte, com a mais fina granularidade possível, e não de forma agregada ou transformada.
Atuais	Os dados são disponibilizados o quão rapidamente seja necessário para preservar o seu valor.
Acessíveis	Os dados são disponibilizados para o público mais amplo possível e para os propósitos mais variados possíveis.
Processáveis por Máquina	Os dados são razoavelmente estruturados para possibilitar o seu processamento automatizado.
Acesso não Discriminatório	Os dados estão disponíveis a todos, sem que seja necessária identificação ou registro.
Formatos não Proprietários	Os dados estão disponíveis em um formato sobre o qual nenhum ente tenha controle exclusivo.
Livres de Licenças	Os dados não estão sujeitos a regulações de direitos autorais, marcas, patentes ou segredo industrial. Restrições razoáveis de privacidade, segurança e controle de acesso podem ser permitidas na forma regulada por estatutos.

Fonte: adaptado de PORTAL BRASILEIRO DE DADOS ABERTOS (2014)

[4]<http://www.opengovdata.org/home/8principles>. Acesso: 01/10/2014

Em Tim Davies (2012), um conjunto de diretrizes foi organizado no sentido apoiar dados abertos. Elas apresentam opiniões e uma série de pontos de vista teóricos e práticos no desenvolvimento de dados abertos e intervenções organizacionais, econômicas e sociais necessárias para apoiar a iniciativa DAGs. A Tabela 7 apresenta uma síntese da proposta de engajamento sobe a qual o publicador de dados deve se questionar antes de liberar seus dados.

Tabela 7 Engajamento de Dados Abertos

Número de Estrela(s)	Detalhe
1	Por demanda: (i) Disponibilizar ferramentas de apoio ao uso dos dados; (ii) Fornecer um canal de comunicação para os clientes contribuírem.
2	Colocar dados em contexto: (i) Disponibilizar informações claras para descrever os dados, incluindo informações sobre a frequência de atualizações, formatos de dados; (ii) Incluir informações qualitativas ao lado de conjuntos de dados, tais como detalhes de como os dados foram criados, ou manuais para trabalhar com os dados; (iii) Disponibilizar análises realizadas acerca dos dados e ferramentas para trabalhar com os dados (podendo ser de terceiros).
3	Suporte a discussão em torno dos dados: (i) Permitir que as pessoas possam comentar sobre conjuntos de dados; (ii) Analisar *feedbacks* das pessoas; (iii) Disponibilizar um canal para perguntas e sugestões com o administrador dos dados; (iv) Disponibilizar conversas *offlines* sobre os dados.
4	Desenvolver capacidade, competências e redes: (i) Fornecer ou disponibilizar *link* para ferramentas para as pessoas que trabalham com os seus conjuntos de dados (ferramentas para visualização); (ii) Fornecer ou disponibilizar *link* para com orientação sobre o uso de ferramentas de análise de dados abertos, para que as pessoas possam construir sua capacidade e habilidades para interpretar e utilizar dados nos caminhos que eles querem; (iii) Procurar a comunidade para executar sessões de capacitação sobre o uso de dados de forma particular, ou utilizar determinados conjuntos de dados; (iv) Patrocinar ou colaborar com a capacitação para ajudar o trabalho comunitário com dados abertos.
5	Colaborar em dados como um recurso comum: (i) Manter interações de *feedback* com as pessoas para que possam ajuda a melhorar as suas bases de dados; (ii) Colaborar com a comunidade para criar novos recursos de dados (por exemplo, conjuntos de dados derivados); (iii) Prestar apoio a pessoas para construir ferramentas úteis e serviços que trabalhem com os seus dados; (iv) Trabalhar com outras organizações para conectar-se às fontes de dados.

Fonte: adaptado de Tim Davies (2012)

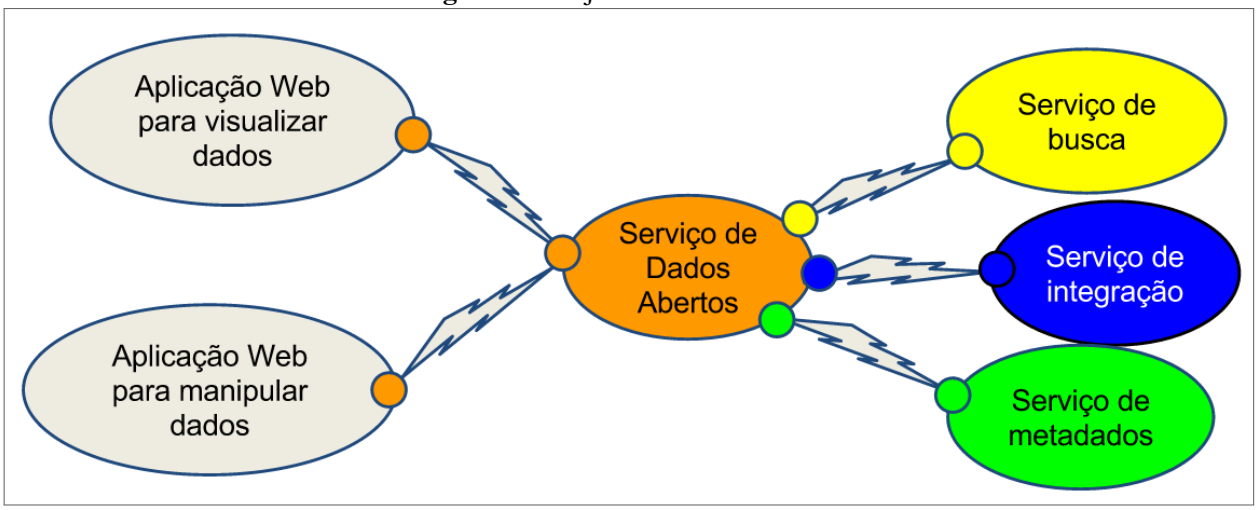
De acordo com Araujo *et al.* (2012, p. 5), a disponibilização de bases de dados governamentais aos cidadãos permite que os interessados possam manipular, agrupar e gerar novos conhecimentos e aplicações. Por meio destes novos conjuntos de dados, uma série de benefícios podem ser desenvolvidos por meio de série de áreas e atividades. Outros benefícios podem ser encontrados no Manual de Dados Abertos (MANUAL, 2011).

2.3 *Web Services*

Na literatura não existe um consenso para uma definição padrão do termo "*Web service*" (Serviço *Web*)[5]. Pode-se entender por *Web service* uma abstração de *software*, implementada em sua forma concreta para ser usada em ambientes distribuídos, sendo ele, sozinho, capaz de fornecer uma maneira para que aplicações heterogêneas interajam por meio de mensagens. Elas podem ser escritas, por exemplo, utilizando notação XML.

Para o W3C (2014), a definição para "*Web Service*" dá-se como um sistema de *software* projetado para suportar interação de forma interoperável e processável automaticamente por máquina em uma rede. Ele tem uma interface descrita em um formato processável por máquina (especificamente *Web Services Description Language* (WSDL)). Outros sistemas interagem com o *Web service* de uma maneira prescrita por sua descrição, usando mensagens do *Simple Object Access Protocol* (SOAP).

Figura 3 Conjunto de *Web Services*



Fonte: o autor

A Figura 3 ilustra um cenário com vários *Web services*, onde cada balão representa um *Web service* responsável por uma atividade específica. Ao centro, em "Serviço de Dados Abertos", é possível observar o *Web service* que concentra um conjunto de serviços/recursos e outros *Web services* que interagem com o mesmo. A comunicação destes *Web services* ocorre por meio de protocolos. Algumas das tecnologias envolvidas na comunicação são descritas na Tabela 8.

[5]Será adotado o termo em sua forma original em inglês

Tabela 8 Tecnologias utilizadas em *Web services*

Tecnologia	Detalhe
XML	Surgiu como padrão para a estruturação e troca de mensagem por meio da Internet.
XML *Schema*	Descreve a estrutura de um documento XML. O XML *Schema* é um sucessor do *Document Type Definition* (DTD) que e permite ter tipos de dados.
SOAP	É um protocolo padrão de troca de mensagens estruturado em XML que possibilita a comunicação entre *Web services*.
WSDL	É um formato XML para e descrever *Web services*, possibilitando a separação entre funcionalidades oferecidas e suas descrições abstratas.

Fonte: adaptado de PORTAL BRASILEIRO DE DADOS ABERTOS (2014)

Os *Web services* especificados pelo W3C (SOAP + WSDL) podem ser categorizados em duas gerações. A primeira especificada possuía várias limitações para seu funcionamento, desde limitações na linguagem para descrição dos serviços a seu funcionamento. A segunda geração, também conhecida como "***WS-****", representa um complemento, fornecendo novos recursos e funcionalidades para suprir as limitações da geração anterior. Dentre essas especificações e padrões, destacam-se o *WS-security framework*[6], *WS-Addressing*[7], *WS-Transaction*, *WS-Attachements* e *WS-ReliableMessaging*[8], considerados fundamentais para o desenvolvimento de aplicações distribuídas que utilizam *Web services* Erl (2004, p. 90-129). Esta coleção de tecnologias é conhecida também como "*Big Web services*" (RICHARDSON; RUBY, 2007, p. xv). Outros detalhes sobre *Web services* podem ser encontrados em W3C (2014) e especificações da segunda geração em Erl (2004).

De acordo com Pautasso, Zimmermann e Leymann (2008, p. 805), recentes tendências tecnológicas em relação ao domínio de *Web services* indicam que a solução capaz de eliminar a complexidade existente nos padrões ***WS-**** é vista no REST. De acordo com Fielding (2000, p. 13), o REST é um estilo arquitetônico baseado em princípios para uso, estes princípios são simples e aplicáveis perfeitamente na *Web*, permitindo resolver problemas como, por exemplo, relacionado à integração entre aplicativos ou mesmo simplificar o mecanismo necessário para construir soluções baseadas em *Service-Oriented Architecture* (SOA).

Pelo fato de SOAP ser um modelo baseado no *HyperText Transfer Protocol* (HTTP), ele funciona bem com *firewalls*[9] (não necessitando de configuração especial de portas para seu uso). Esta talvez essa seja uma das principais vantagens do SOAP sobre outras tecnologias, por exemplo: *Distributed Component Object Model* (DCOM) ou *Common Object Request Broker Architecture* (CORBA). Uma das desvantagens do SOAP, porém, é seu formato XML, que o torna demorado, tornando-o lento em comparação com formatos binários (ou baseado em textos "mais leves") para a transmissão (influenciando no quesito largura de banda), assim como também para analisar as mensagens.

Em Sumaray e Makki (2012), são apresentadas comparações com diferentes formatos de

[6]<http://www.w3.org/TR/ws-policy-primer/>. Acesso: 01/10/2014
[7]<http://www.w3.org/Submission/ws-addressing/>. Acesso: 01/10/2014
[8]<http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf>. Acesso: 01/10/2014
[9]<http://www.oxforddictionaries.com/us/definition/american_english/firewall>. Acesso: 01/10/2014

dados serializados. Dentre os formatos, estão o XML e JSON. Entre as comparações, estão a medida da velocidade necessária para serialização (codificar) e desserialização (decodificar). Os experimentos do trabalho apresentado por Sumaray e Makki (2012) mostraram que o formato de dados XML é largamente inferior aos outros formatos de serialização. O XML também apresentou o maior tamanho serializado e velocidade mais lenta para as operações.

Para Sumaray e Makki (2012), o XML deve ser evitado, a menos que necessário, o JSON tem se mostrado uma alternativa superior. Também é terminantemente recomendado que *Web services* utilizem JSON sempre que possível, uma vez que é um formato baseado em texto ("mais leve" em comparação ao XML) e pode ser escrito e analisado em qualquer plataforma.

Uma das principais diferenças entre SOAP e REST diz respeito à manutenção do estado no lado do servidor. Enquanto nas operações em SOAP o servidor pode manter o controle de conversas individuais do cliente, no REST essa manutenção de estado no lado do servidor viola um dos seus princípios. Isto significa que cada pedido de um recurso usando REST deve fornecer todas as informações necessárias para o processamento do pedido.

Segundo Daigneau e Robinson (2011), diversas tecnologias como CORBA e DCOM, assim como REST e SOAP/WSDL, podem ser usadas para criar *Web services*. Esta dissertação adotou o REST para especificação da proposta. A seguir, serão vistos outros detalhes sobre o REST.

2.3.1 REST

De acordo com Fielding (2000, p. xvi), a *Web* é bem sucedida devido ao projeto da arquitetura de software ser capaz de atender às necessidades de um sistema hipermídia distribuído[10] em escala de *Internet*. A *Web* teve sua evolução de forma interativa e passou por várias intervenções nos padrões que definem sua arquitetura.

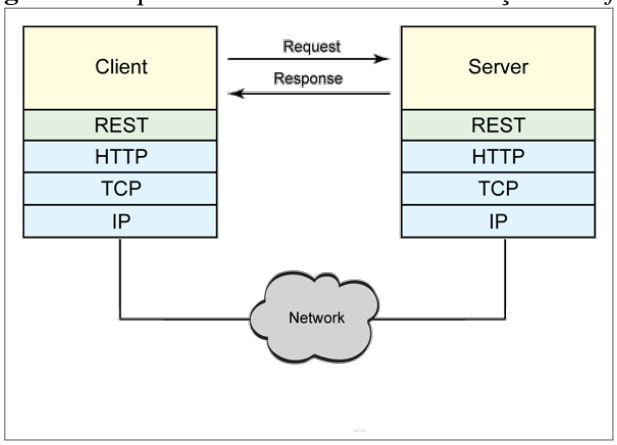
Segundo Fielding (2000), o REST pode ser entendido como um estilo arquitetônico para ser usado como orientação a concepção e desenvolvimento de sistemas de hipermídia distribuídos em larga escala.

De acordo com Garlan *et al.* (2010, p. 492), um estilo arquitetônico pode ser entendido como uma especialização de elementos com tipos e relações, juntamente com um conjunto de restrições sobre como eles podem ser utilizados. Esta definição também está presente no glossário da página oficial do *Software Engineering Institute* (SEI)[11]. Em relação ao estilo arquitetônico REST, Pautasso, Zimmermann e Leymann (2008, p. 807) conclui que ele pode ser entendido como uma entidade abstrata, cujos princípios têm sido usados para explicar a excelente escalabilidade do protocolo HTTP.

O termo ***Restful***, comumente encontrado na literatura, é utilizado para referenciar aplicações que implementam os princípios do REST. De acordo com Richardson e Ruby (2007, p. xvi), ele tem significado semelhante ao utilizado para designar "orientação a objetos". Uma aplicação pode ser desenvolvida usando linguagem orientada a objetos, mas isso não faz com que sua arquitetura seja arquitetura orientada a objetos. O termo *Restful* segue a mesma lógica.

[10] <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-29012001-160200/pt-br.php>. Acesso: 01/10/2014
[11] <http://www.sei.cmu.edu/architecture/start/glossary/>. Acesso: 01/10/2014

Figura 4 Arquitetura em Camadas de Interações *Restful*



Fonte: (JONES, 2012)

A Figura 4 apresenta uma ilustração básica de interações do tipo cliente e servidor em uma aplicação *Restful*. O cliente realiza o pedido, que é traduzido em uma solicitação HTTP. Este pedido é iniciado como qualquer outra transação HTTP entre cliente e servidor. O servidor *Web*, por sua vez, recebe a solicitação e responde após processar o pedido. Também é apresentada na Figura 4 uma abstração de sistema baseado em camadas[12].

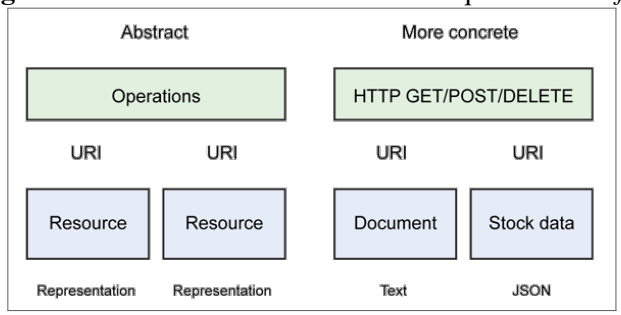
Em relação ao REST, Pautasso, Zimmermann e Leymann (2008, p. 807) afirma que o termo REST é frequentemente utilizado em conjunto com o HTTP. De acordo com Fielding (2000, p. 100), o REST pode ser implementado utilizando outro protocolo, não apenas com o HTTP. Nesta seção, serão apresentadas as principais características do REST com foco na interpretação atual, usada para definir "*Restful Web services*". Outras informações sobre o REST podem ser encontradas também em Fielding (2000).

2.3.2 Princípios do REST

Os princípios são características determinantes estabelecidas pelo estilo REST (algumas essenciais e outras não). As obrigatórias são entendidas como as restrições, ou seja, condição restritiva, essencial. As demais características são determinantes, mas não essenciais (sugestões de uso). O conjunto coordenado destes princípios arquiteturais formam o estilo arquitetural REST (FIELDING, 2000).

[12]O conceito de camadas contribui facilitando a estruturação e divisão das responsabilidades por meio de níveis de abstrações.

Figura 5 Visão de Alto Nível de uma Arquitetura *Restful*



Fonte: (JONES, 2012)

A Figura 5 ilustra uma arquitetura *Restful* por dois pontos de vista. Os pontos de vista abstrato e concreto representam os princípios do REST. Para mapear a forma abstrata em concreta, foi utilizado o HTTP. A base de uma arquitetura REST são os recursos e estes estão representados na ilustração. Os recursos são identificados por URIs e uma representação interna. Finalmente, há um conjunto de operações por meio do qual é possível manipular os recursos. O mapeamento das operações na ilustração pode ser feito por meio dos método do HTTP *POST*, *GET*, *PUT*, *DELETE* e outros.

O REST define quatro princípios (restrições) de interface (FIELDING, 2000, p. 82). São eles: *a*) identificação dos recursos; *b*) manipulação de recursos por meio de representações; *c*) mensagens autodescritivas; e *d*) Conceito de *Hypertext As The Engine Of Application State* (HATEOAS)[13]. Estas restrições são a base do REST.

Quando o REST é implementado utilizando o protocolo HTTP, a identificação dos recursos é feita por meio do URI. Um *Restful Web Service* apresenta um conjunto de recursos e eles são utilizados pelos clientes para interagir. Em "*Cool URIs don't change*", datado em 1998 por *Tim Berners-Lee*, isso demonstra sua preocupação quanto ao uso dos URIs e sua manutenção de modo que eles permaneçam fiéis à sua criação, independente da quantidade de tempo que devam existir. O mesmo documento é uma prova desta preocupação e comprometimento, permanecendo até nos dias de hoje com a mesma URI.

Os recursos são basicamente manipulados, usando um conjunto fixo de quatro operações definidas pelos métodos do HTTP. Os métodos *POST*, *GET*, *PUT* e *DELETE* compõem estas operações. O *POST* é utilizado para criar um novo estado ao recurso. Usando *DELETE*, é possível remover um estado existente em um recurso. O *GET* recupera um estado ou uma coleção de estados em um recurso representados com alguma notação. O *PUT* atualiza um estado pertencente ao recurso. Este mapeamento cumpre a meta "restrição" da interface uniforme do REST.

Para Pautasso, Zimmermann e Leymann (2008, p. 807), "mensagem autodescritiva" pode ser entendida como a capacidade da mensagem em possuir toda informação necessária. Para realizar uma transação com o recurso, todo o conteúdo necessário para ser entendido pelo servidor deve ser enviado na mensagem. Por exemplo, deve conter informações sobre o formato de representação solicitado pelo cliente, realizar autenticação no sistema e etc.

[13]<http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven>. Acesso: 01/10/2014

Segundo Medjahed e Bouguettaya (2011, p. 41), as interações sem estado (*Stateless*) (não necessitam manter informações do lado do servidor), por meio de *hiperlinks*, podem ser realizadas por meio de várias técnicas para troca de estado (por exemplo, reescrita de URI, *cookies* e campos de formulário ocultos). O estado deve ser incorporado nas mensagens de resposta para apontar para futuros estados válidos da interação. Pautasso, Zimmermann e Leymann (2008, p. 807) complementa que cada interação com um recurso deve ser sem estado, ou seja, as mensagens de solicitação precisam ser autossuficientes. As interações (sem estado) são baseadas no conceito de estado explícito na transferência. Mais informações sobre as relações dos recursos na *Web* são apresentadas na *Request for Comments* (RFC) 5988, especificada em Nottingham (2010).

Existem outros cincos princípios presentes no REST. São eles: (i) *Stateless Interactions* (Interações sem estado); (ii) *Cacheable* (Armazenáveis); (iii) *Client-Server* (Modelo para comunicação em um sistema baseado em aplicação distribuída que atribui fluxo de processamento entre consumidores (cliente) e fornecedores (servidor)); (iv) *Layered System*(Sistema baseado em camadas); (v) *Code on Demand*(Código por demanda) . Cada uma das restrições apresentadas serão descritas a seguir de forma detalhada:

- **Interações Sem Estado** - foi brevemente mencionada e diz respeito ao fato que nenhuma informação do cliente deve ser armazenada no lado do servidor (como por exemplo por meio do uso de sessão ou *cookie*). Toda informação necessária para processar o seu pedido deve ser enviado junto da requisição, podendo estar contida na própria URI ou inclusa em parâmetros de consulta da *Uniform Resource Locator* (URL) (Localizador Universal de Recurso) ou por meio do uso de atributos no cabeçalho da mensagem HTTP. Em interações *stateful*s (com estado), são armazenadas informações do cliente no lado do servidor, mantendo o controle do estado da conexão. As interações *stateful*s são mais custosas para o servidor.

- **Armazenáveis** - faz referência à possibilidade de armazenamento, por parte dos clientes, em ambiente local, as requisições processadas. A resposta do servidor que implementa REST deve permitir que o cliente armazene cópias locais dos dados para que não seja preciso enviar novo pedido para um já processado.

- **Cliente e Servidor** - essa restrição diz respeito a separação entre o cliente e servidor. Com isso, as responsabilidades são fragmentadas, tornando as separadas também. Estas preocupações vão desde portabilidade do código e possibilidades de dimensionamento do lado do servidor.

- **Sistema em Camadas** - torna as responsabilidades melhor definidas e aumenta a possibilidade de os clientes conectarem-se ao servidor por meio destas abstrações de camadas. A separação das camadas contribui também como um reforço as políticas de segurança incorporadas do lado do servidor. Também pode se beneficiar para melhorar a escalabilidade.

- **Código por Demanda** - sendo esta a única opcional (não restrição), diz respeito ao servidor poder atender a funcionalidades de um cliente.

2.4 Processo de ETL

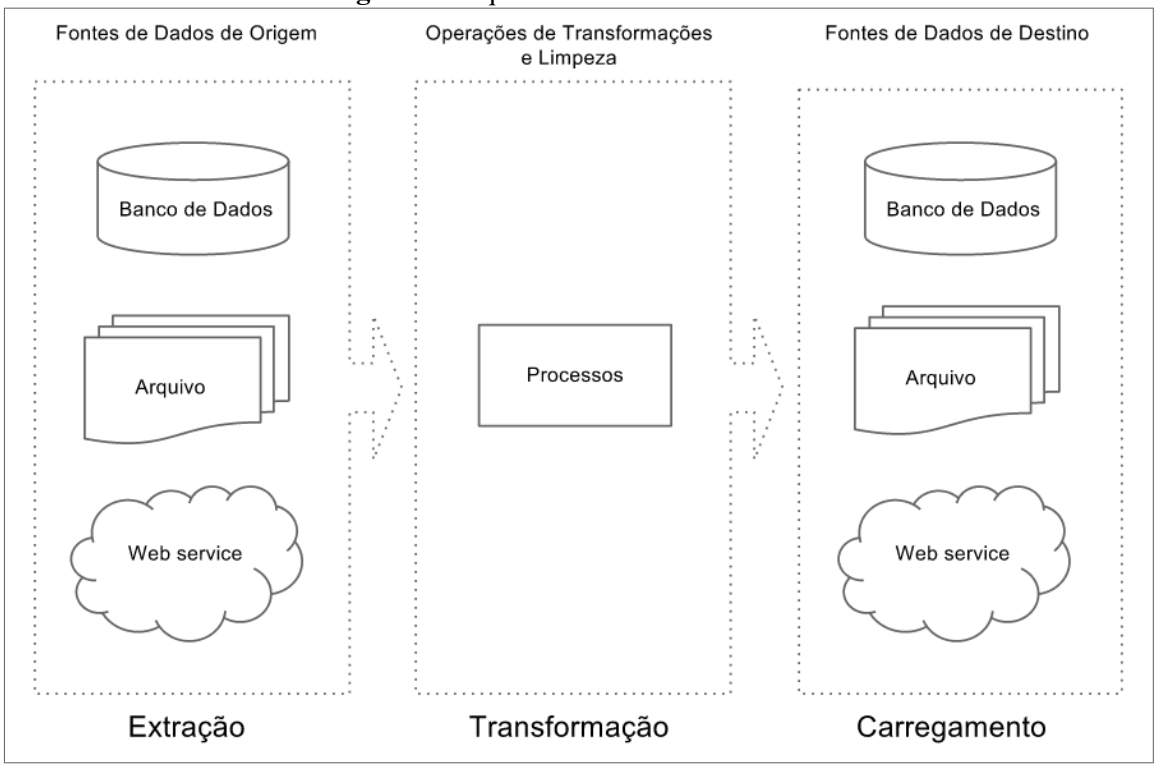
Segundo Casters, Bouman e Dongen (2010, p. 5), o processo de ETL (Extração, Transformação e Carregamento) pode ser definido como um conjunto de processos para obter dados dos sistemas de *Online Transaction Processing* (OLTP) em um *Data Warehouse*[14] (ou armazém de dados). Mas esta definição não abrange a modernidade das ferramentas de ETL. Os dados não são apenas provenientes de sistemas OLTP, mas também de sites, arquivos simples, bancos de dados, *e-mail*, planilhas e bancos de dados pessoais.

Para Vassiliadis (2009, p. 1-2), os processos de *software* que facilitam o preenchimento e atualização periódica do conteúdo no armazenamento de dados são comumente conhecidos como processos de ETL. A intenção desta seção é apresentar e descrever estas atividades do processo de ETL: *a*) extração dos dados apropriados a partir das fontes; *b*) transformação dos dados de origem e o cálculo para novos valores e, possivelmente, registros, a fim de obedecer a estrutura de destino; *c*) o carregamento dos dados transformados para a relação apropriada no armazém de dados, juntamente com a atualização de seus índices que acompanham e visualizações materializadas (representação do dado em seu destino) .

As ferramentas de ETL são utilizadas com diferentes objetivos (CASTERS; BOUMAN; DONGEN, 2010, p. 9). Elas oferecem suporte a um conjunto variado de opções de conectividade e transformações nos dados. Elas também colaboram na migração de dados, por exemplo, migrar dados do banco B para uma planilha de dados em notação CSV.

[14]De acordo com Inmon (1996), um Data Warehouse pode ser entendido como uma coleção de dados interligados, orientados por categorias, não voláteis e sensíveis com relação ao tempo e auxiliam às tomadas de decisão.

Figura 6 Arquitetura do Processo de ETL



Fonte: adaptado (KIMBALL; CASERTA, 2004, p. 43)

Na Figura 6, é possível observar uma arquitetura clássica de um processo de ETL. Ele não é apenas usado para carregar uma base de dados única, mas podem ter muitas outras atribuições, gerando carga em planilhas, arquivos e outras operações. As principais etapas do processo de ETL, no entanto, podem ser agrupadas em três grupos: extração, transformação e carregamento. Na próxima seção, serão apresentados mais detalhes sobre a etapa de extração.

2.4.1 Extração

A extração é a primeira etapa do processo de ETL, sendo ela responsável por extrair o conteúdo de fontes de dados que podem ser heterogêneas (LIU; ZSU, 2009, p. 677). De acordo com Casters, Bouman e Dongen (2010, p. 5), esta etapa também é responsável por todo o processamento necessário para conectar as várias fontes de dados, extrair os dados e tornar estes dados disponíveis para as etapas de processamento subsequentes. Os dados são extraídos das fontes de dados e são armazenados em uma estrutura física e lógica temporária para que outras operações possam ser realizadas.

Para Kimball e Caserta (2004, p. 116), o conteúdo da extração pode sofrer alterações quanto à sua forma original. Por exemplo, informações de clientes, provenientes de diferentes fontes, podem ser apresentadas em uma visão tabular (tabela), arquivo de dados ou estrutura de dados na memória, cuja configuração é a mesma, independentemente da fonte de dados de origem. A estrutura é definida para receber os dados coletados das diferentes fontes de dados de origem, os quais serão armazenados nesta estrutura para que seja possível a execução das demais atividades do processo de

ETL.

2.4.2 Transformação

O segundo marco no processo de ETL está nas operações de transformações dos dados (KIMBALL; CASERTA, 2004, p. 116). Para Liu e Zsu (2009, p. 677), esta atividade também consiste em analisar e integrar os dados. Ao término das transformações (inclui limpeza), o conteúdo está apto a ser carregado em seu formato de destino final.

Também é responsabilidade desta etapa analisar a qualidade das operações realizadas. Em essência, são executadas verificações para garantia da veracidade das operações. As operações de transformação contribuem diretamente para agregar valor aos dados gerados devido à natureza das operações realizadas (KIMBALL; CASERTA, 2004, p. 113).

A lógica por trás do mecanismo para limpar os dados também inclui em suas atividades confrontá-los com os requisitos de negócios. Cada sistema armazena os dados de uma forma que é única para aquele sistema em particular. Por exemplo, isto quer dizer que no sistema de RH (a), o atributo que identifica o sexo do cliente é armazenado como F (para o sexo feminino), M (para o homem), e U (por desconhecido), enquanto que o sistema de Recursos Humanos (RH) (b) teria a mesma informação codificada como 0, 1, e nulo.

Segundo Casters, Bouman e Dongen (2010, p. 20), boa parte do trabalho envolvido no processo de ETL tem relação com a transformação de dados. Entre coleta e entrega, os dados precisam ser validados, agregados, divididos, combinados, transportados, classificados, mesclados, clonados, duplicados, filtrados, excluídos, substituídos além de outras operações possíveis. É difícil dizer qual é o conjunto mínimo de transformações disponíveis que precisam ser feitas, pois as transformações nos dados diferem de acordo com os dados de origem e objetivos a serem alcançados.

2.4.3 Carga

Na fase final do processo de ETL, os dados encontram-se extraídos e transformados, prontos para serem carregados para uma representação final, por isso, ela é a última fase (LIU; ZSU, 2009, p. 677). Os dados podem ter sido extraídos de uma ou várias fontes, e nesta última fase eles são carregados em um ou vários repositórios.

Dependendo da natureza dos dados, eles periodicamente podem exigir algum tipo de atualização. Dentre as possíveis estratégias utilizadas para atualizar os dados, está a de carregá-los em sua totalidade na primeira vez (em sua criação) e depois de forma incremental, durante manutenção. A frequência de atualização pode ser em horas, dias ou meses. Cada programação depende da necessidade do negócio em questão.

2.5 Trabalhos Relacionados

Nesta seção serão apresentadas algumas das soluções adotadas processo de publicação e suo de dados abertos. Dentre elas, uma das mais expressivas é a plataforma CKAN que se tornou padrão

mundial. Ela é coordenada pela *Open Knowledge Foundation* (OKFN), a qual é uma organização que promove o conhecimento aberto, incluindo dados e conteúdos abertos em todas as suas formas como: publicação de documentos *online*, divulgação de dados governamentais entre outros.

Dentre as diversas abordagens para divulgação de dados abertos, têm-se soluções que é possível destacar desde os editores de planilhas até as ferramentas mais poderosas de processamento de dados e as aplicações online, nas quais é possível encontrar uma vasta variedade de ferramentas disponíveis. Serão descritas a seguir algumas soluções que contribuem com as atividades de publicação de dados:

- *Scraperwiki*[15]: é uma plataforma *Web* de extração e análise de dados. Foi lançada para auxiliar os jornalistas a analisar e utilizar mais dados para reportagens. Em 2011, conquistou o Desafio Jornalístico *Knight* no Reino Unido.

  Essa ferramenta oferece novas opções para conseguir dados de sites em um formato legível para humanos. Ela permite que os usuários realizem a extração de dados de PDFs, CSVs, Excel e *HyperText Markup Language* (HTML); pesquisar no *Flirck* por imagens contendo dados geográficos, entre outros atributos disponíveis na ferramenta.

  Depois de realizar a tarefa de extração dos dados e torná-los portáteis, os usuários têm várias possibilidades para trabalhar com dados, como: podem visualizá-los em um formato de tabela; criar um gráfico ou mapa do conjunto de dados utilizando consultas SQL; fazer *download* ou compartilhar os dados, ou resumi-los com uma série de ferramentas de visualização.

- *Open Refine*[16] (*ex-Google Refine*): é uma solução para tratamento de dados. Ela contribui na limpeza e transformação entre diversos formatos, além de estendê-los através de *Web services* e integrar a outras bases.

- *CKAN*[17]: é um sistema de gerenciamento de dados abertos *open source*, o qual oferece ferramentas para agilizar a publicação, compartilhamento, recuperação de dados, além do que favorece a utilização de dados publicados. A sua utilização consiste em países que publicam dados abertos ao redor do mundo para tornar suas informações disponíveis.

  Na plataforma existem diversas características como: publicar e permitir a pesquisa, tanto com utilização de filtros como de *tags* em dados através de uma interface web; armazenar os dados brutos e também seus metadados; visualizar os dados em formatos estruturados como tabelas, gráficos e mapas; ter integração direta como gerenciadores de conteúdo como *Drupal* e *Joomla*; criar um sistema de *harvester* para interoperabilidade com outros portais de dados abertos; tem uma licença *open source* e por isso permite seu uso por qualquer pessoa e também a customização quando necessária; suporte a fluxo de trabalho permitindo que departamentos ou grupos possam gerir a sua própria publicação de dados e por fim, um sistema de catálogo completo com interface *Web* e uma API robusta;

[15]<https://scraperwiki.com>. Acesso: 28/11/2014
[16]<http://openrefine.org>. Acesso: 01/10/2014
[17]<http://ckan.org/>. Acesso: 28/11/2014

- *Socrata*[18]: é um portal de dados abertos, que oferece aos usuários uma melhor maneira de acessar e usar as informações públicas. Ao invés de passar por um processo formal para solicitar informações, eles podem avaliar, comparar, visualizar e analisar os dados - e compartilhar suas descobertas - em tempo real.

  A plataforma provê ferramentas de fácil utilização para publicação e atualização de dados de planilhas, sistemas de arquivos, bancos de dados transacionais, e fontes de dados. Além disso, ela automatiza a publicação e agenda atualizações, disponibiliza bibliotecas do cliente com base em API e ferramentas de terceiros como o *Pentaho* e *Software* Seguro.

  Ela também pode definir um vocabulário padrão de *metadados* para organizações. Permitindo que seja possível criar e manter um inventário de dados da empresa que pode ser exposto via APIs, como por exemplo, em um arquivo *data.json*.

- *Junar*[19]: é uma plataforma Web de dados abertos, pois permite que organizações e governos disponibilizem seus dados para gerar novas oportunidades de colaboração e transparência. Foi construída para gerenciar grande volumes de dados e dar suporte para organizações locais e globais. O usuário pode facilmente decidir quais dados serão coletados, como apresentá-los, e quando disponibilizá-los, podendo também adicionar *metadados* às informações que irá ser publicada.

  Essa plataforma contém várias funcionalidades dentre elas, pode-se destacar:

  Coletar: a plataforma pode ajudar a coletar e organizar as informações que desejam compartilhar, em um único lugar. Tornar fácil a escolha dos dados a serem coletados, independente do formato em que se encontrem - planilha, PDF ou *Web*.

  Melhorar: transformar os dados em informações produtivas, que possam ser acessadas e compartilhadas por seus públicos de forma interessante, por meio de mapas, tabelas e gráficos.

  Publicar: permite a criação de um *workflow* para ajudar o gerenciamento da publicação dos dados, tendo sempre em vista que tem processos internos específicos de trabalho. Ela permite especificar o que cada membro da equipe pode fazer, desde coletar, contextualizar e publicar dados. Uma vez publicados os dados, os usuários podem compartilhar o que acham interessantes, e até mesmo consumir através de uma API, permitindo aos desenvolvedores acessar os dados de seus aplicativos de programação.

  Analisar: relatórios são gerados em formatos de tabelas, visualizações e planilhas, inclusive apresenta quais dados estão sendo consumidos por meio da API.

  Kettle[20]: é uma solução utilizada para integração de dados. Faz parte da suíte do *Pentaho*, tem licença *open source*, suporta um vasto conjunto de formatos de entrada e saída de dados, abrange os processos de Extração, Transformação e Carga ETL.

[18]<http://socrata.com>. Acesso: 28/11/2014
[19]<http://www.junar.com>. Acesso: 28/11/2014
[20]<http://community.pentaho.com/projects/data-integration/>. Acesso: 28/11/2014

Tabela 9 Comparação entre as diferentes ferramentas

Característica	Scraperwiki	Open Refine	CKAN	Socrata	Junar	Kettle
Extração					x	x
Transformação/Limpeza	x	x		x		x
Exportação	x	x		x	x	x
Catalogação na *Web*			x	x		
Publicação	x	x	x	x	x	
Atualização						
API	x	x	x	x	x	x

Fonte: o autor

Segue abaixo alguns critérios de avaliação das ferramentas:

- Extração: capacidade de extrair dados.
- Transformação/Limpeza: capacidade de executar operações de transformação.
- Exportação: capacidade de exportar o dado por meio de uma representação de dados.
- Catalogação: capacidade de catalogar os dados na *Web*.
- Publicação: capacidade de disponibilização dos conjuntos de dados na *Web*.
- Atualização: capacidade de atualização dos dados de maneira automática e republicar na *Web*.
- API: possui API para ter acesso as funcionalidades

Este trabalho propõe uma abordagem baseada em serviços Web para publicação e uso de dados abertos que contempla todas as características analisas nesta seção. Na próxima seção, serão apresentadas as conclusões ao capítulo.

2.6 Conclusões

Este capítulo apresentou uma revisão da literatura, a fim de fundamentar o presente trabalho e ressaltar a importância do tema e o modo como ele tem sido abordado sob a ótica de diferentes autores, bem como a necessidade de estudar métodos com vistas a desenvolver novas abordagens que possam ser principalmente mais duradouras. Foram também discutidos os principais assuntos relacionados a abordagem proposta. Para isso, foram descritos os principais pontos referentes a Dados Abertos, bem como Dados Abertos Governamentais, com enfoque nos princípios e sua utilização. Em seguida, foi apresentado um embasamento sobre *Web service*, que consiste como proposta de solução para o problema de pesquisa desta dissertação. Seguidamente, foi descrito

brevemente o processo de ETL utilizado neste trabalho, bem como detalhes que são mencionados no decorrer desta dissertação. Por fim, foram apresentadas as ferramentas relacionadas e realizado uma análise comparativa utilizando algumas das características encontradas no processo de publicação e uso de dados abertos.

O próximo capítulo irá descrever, em detalhes, a abordagem proposta para esta dissertação, assim como, fornecer uma visão geral da arquitetura a ser implementada pelo *web service*.

3

Descrição da Abordagem

Não importa o quanto você bate, mas sim o quanto aguenta apanhar e continuar. O quanto pode suportar e seguir em frente. É assim que se ganha.

—SYLVESTER STALLONE - ROCKY BALBOA

Este capítulo apresenta a abordagem para automatizar o processo de publicação de dados abertos, com o objetivo de facilitar o seu uso. São apresentadas algumas características e um conjunto de requisitos considerados essenciais para a construção da abordagem. Por fim, é apresentada a visão geral da arquitetura, descrevendo cada uma das etapas realizadas pela abordagem proposta.

3.1 Requisitos Desejáveis

Esta seção apresenta os elementos essenciais para elaboração desta abordagem. Estes elementos são tratados como requisitos desejáveis. O termo desejável é empregado no sentido de ser importante, ser necessário, ter um valor significativo para o desenvolvimento deste trabalho de pesquisa. Os requisitos são apresentados sem muito formalismo, objetivando apenas sua identificação.

Estas características foram analisadas dentro do processo de publicação e uso de dados abertos e são adotadas por esta abordagem. São elas:

Fontes de Dados:

- podem estar geograficamente distribuídas.
- podem ter um único proprietário ou vários.
- pode não haver controle da evolução.
- são heterogêneas.
- as principais são: Sistemas Gerenciadores de Bancos de Dados (SGBD) *PostgreSQL*, *MySQL* e documento de notação CSV.

Catalogação dos dados:

- deve fornecer uma maneira para catalogar os dados por conjunto de dados e diferentes representações.

- deve manter estas informações para catalogação de forma expressiva e processáveis por máquina.

Significado dos dados:

- a descrição sobre os dados deve ser feita também por meio da inclusão de *metadados*.

Notações para os dados e *metadados*:

- deve fornecer conteúdo em notações: HTML, XML, CSV e JSON.
- o cliente deve ser capaz de solicitar sua representação (notação).

Periodicidade de atualização dos dados nas fontes de dados:

- deve ser capaz de atualizar os dados e *metadados* automaticamente.

Operações de manipulação dos dados:

- deve fornecer funcionalidade que permita a conversão de um dado valor para outro no conteúdo dos dados extraídos.
- deve permitir operações de conversão, seleção e ordenação.

Acesso:

- deve permitir interação com outros *softwares*.

Manutenção do sistema:

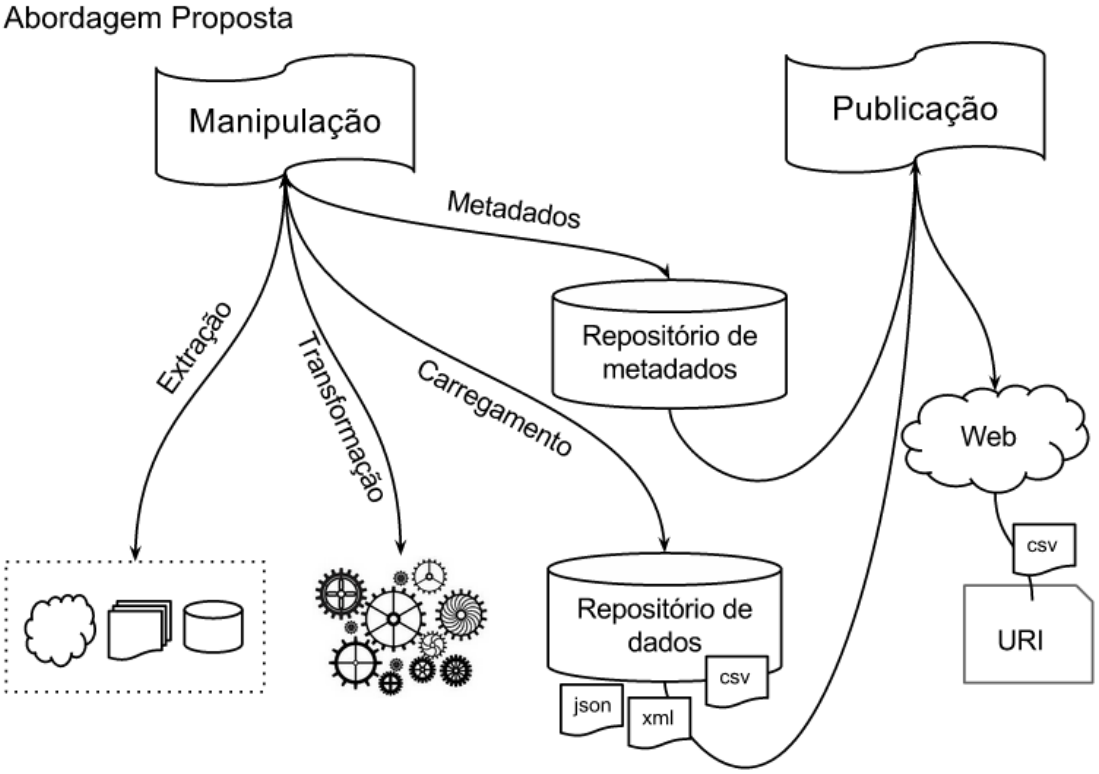
- deve ser organizado de forma a facilitar futuras atualizações.

Na próxima seção serão explicadas cada atividade que compõem o processo proposto pela abordagem deste trabalho.

3.2 Abordagem Proposta

Nesta seção, serão descritas as diferentes atividades que são realizadas pela abordagem desta dissertação.

Figura 7 Atividades do Processo de Publicação e Uso de Dados Abertos da Abordagem



Fonte: o autor

Estas atividades são divididas em dois grupos: Manipulação e Publicação, como podem ser visualizadas na Figura 7. Estas atividades são descritas a seguir:

- **Atividade Manipulação**:
  - Etapa de Extração - Neste momento, ocorre a seleção dos dados a serem extraídos das fontes de dados de origem e armazenados em uma estrutura de dados que permita a execução da próxima etapa.
  - Etapa de Transformação - Os dados encontram-se transformados. Esta etapa também é conhecida como etapa de tratamento e higienização dos dados. Operações de substituição de dados são realizadas nesta etapa.
  - Etapa de Carregamento - Neste momento, os dados encontram-se no repositório de destino. Os dados são carregados (exportados) para uma representação de dados.
  - Etapa de Catalogação - Diz respeito à descrição dos dados. Neste momento os dados são descritos de forma a facilitar sua identificação dentro do conjunto de dados.
  - Etapa de Metadados - Corresponde ao repositório de *metadados*.
- **Atividade Publicação**:
  - Etapa de Publicação - Torna os dados das etapas anteriores disponíveis na *Web*.

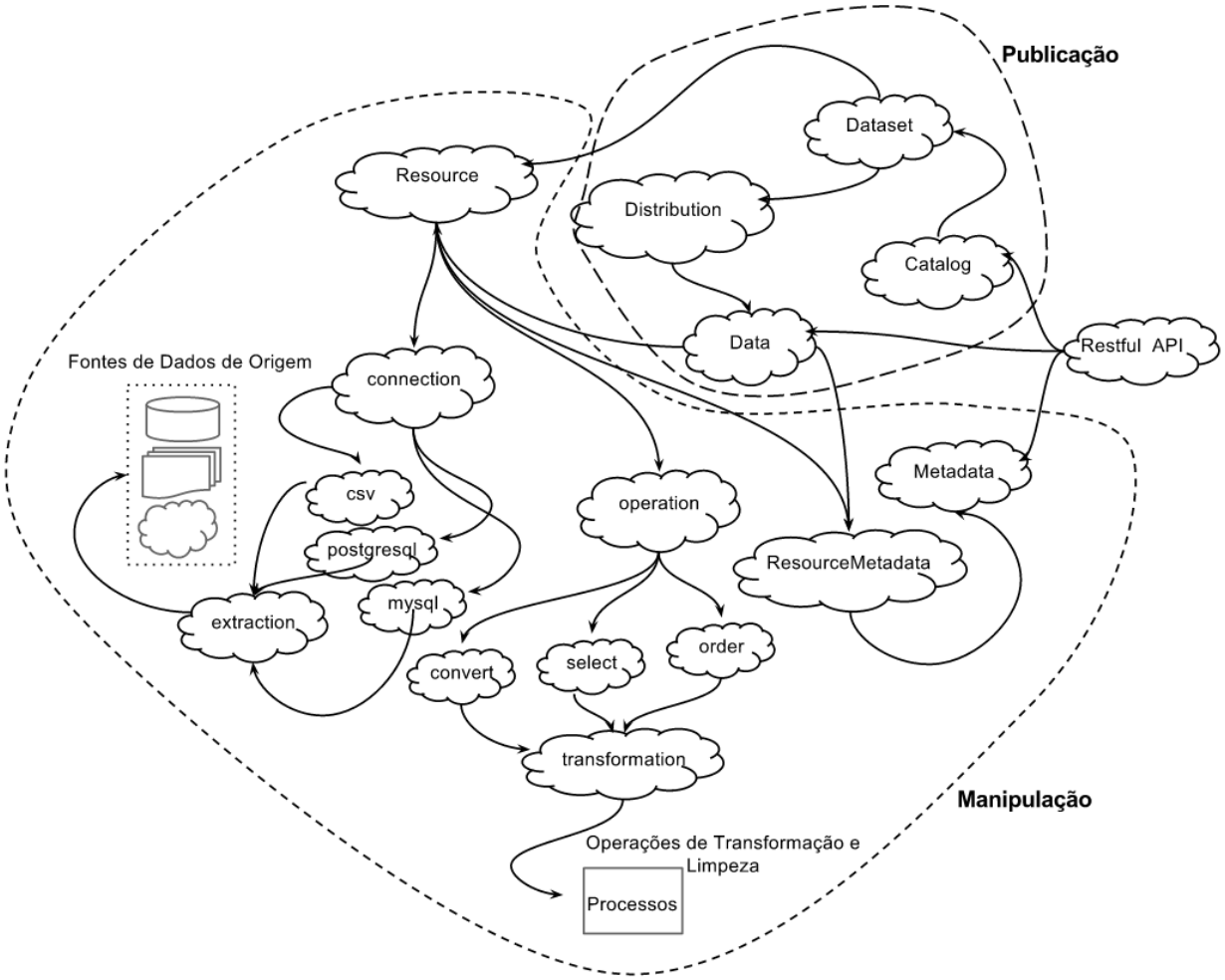
A estratégia adotada para a concepção das atividades acima foi por meio da utilização de serviços Web. O conjunto das atividades executadas dentro do processo de manipulação e

publicação apresentados é fornecido de maneira interoperável e executa as operações de maneira a se complementarem. Neste sentido, as futuras atualizações se beneficiam e podem ser processadas de forma automática.

Para a concepção das atividades acima como serviços, foi adotado o termo "recurso" utilizado na abordagem REST. Em REST os serviços são tratados como recursos. Dizer que a abordagem é baseada em serviços ou recursos tem sentidos semelhantes, os dois podem ser tratados como um ponto endereçável dentro de uma rede, objetivando a realização de trabalho. O primeiro é mais utilizado para denotar uma funcionalidade, o segundo possui por definição um sentido mais amplo, como visto no capítulo 2 na seção que aborda o REST.

A Figura 8 ilustra os recursos que são utilizados, bem como destaca a interação da API e serviços: os clientes da API realizam suas solicitações por meio da nuvem "*Restful* API". Em seguida, a API captura essas transações (solicitações), prepara o formato da mensagem e as envia ao recurso solicitado. Este, por sua vez, processa o recurso e retorna a solicitação ao cliente.

Figura 8 Recursos utilizados pela abordagem



Fonte: o autor

Os serviços estão representados de forma hierárquica na Figura 8 e, simbolicamente, cada nuvem da imagem representa um recurso. Os recursos estão conectados entre si através de setas de acordo com suas relações, podendo existir ou não, a relação de dependência dos serviços.

Nesta abordagem, primeiramente será tratada a parte da catalogação dos dados, sendo responsável por fornecer a descrição para os dados disponibilizados. Em primeiro lugar, será criado um catálogo, em seguida um *dataset* e, por último, uma distribuição de dados. Cada uma destas atividades serão melhor explanadas.

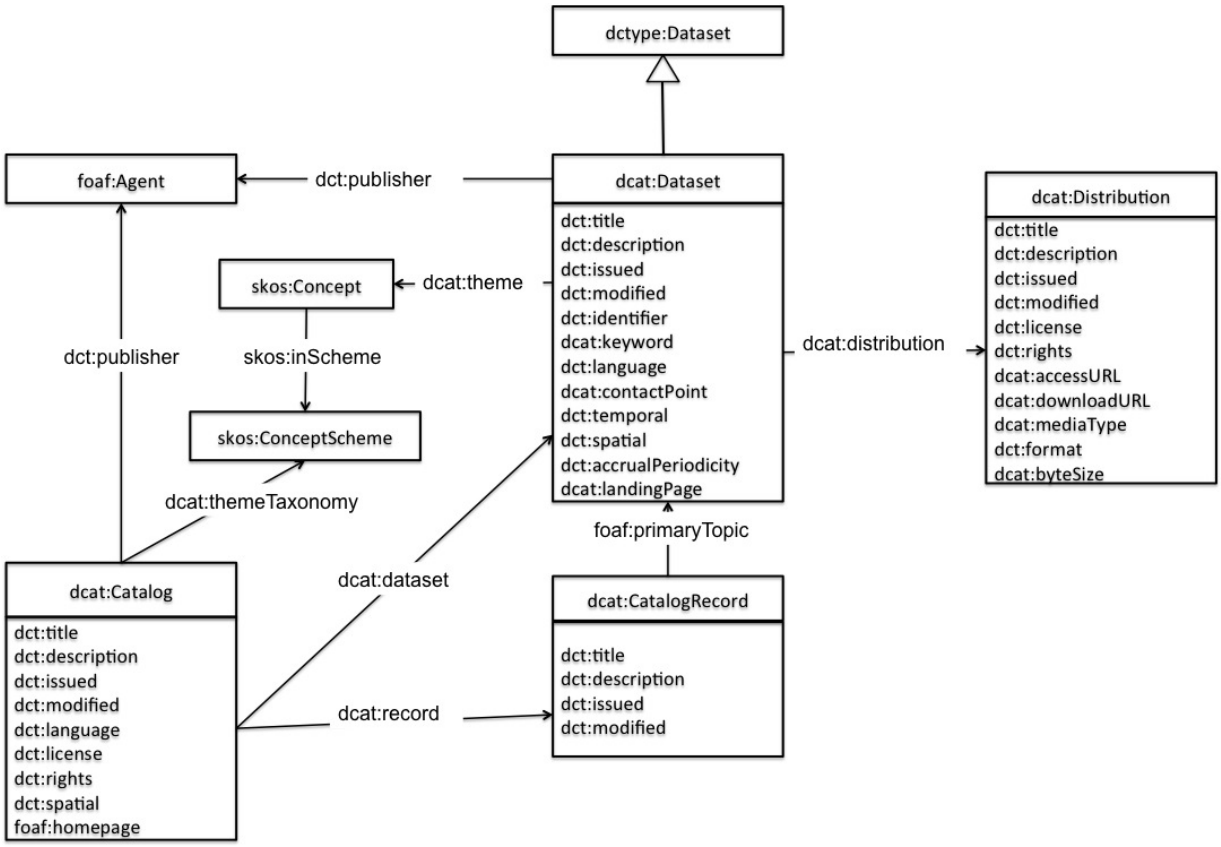
Para catalogação dos dados, será adotado o *Data Catalog Vocabulary* (DCAT), o qual é um vocabulário[1] para representação de catálogos homologado pelo W3C. O DCAT foi projetado para facilitar a interoperabilidade entre catálogos de dados na *Web*, pois seu vocabulário fornece expressividade para integrar dados com outros conjuntos de dados de maneira interoperável.

Adicionalmente, o vocabulário DCAT especifica quais entidades são representadas, como elas estarão agrupadas, e quais relacionamentos as mantém interligadas e pode ser visto como uma relação direta entre quem fornece os dados e quem os utiliza. A Figura 9 apresenta o diagrama de classes e propriedades do DCAT. Este, por sua vez, faz reúso das especificações de outros vocabulários. O vocabulário *Dublin Core*[2] possui o maior conjunto desse reaproveitamento das propriedades feito pelo DCAT. Todas as propriedades que começam com o prefixo "*dct:*" pertencem ao *Dublin Core*.

Os catálogos, *datasets* (conjunto de dados) e distribuições fornecerão a base para futuras integrações com outros conjuntos de dados. Cada uma destas três entidades (classes) mencionadas possui propriedades que podem ser vistas na Figura 9.

[1] <http://www.w3.org/TR/vocab-org/>. Acesso 28/11/2014
[2] <http://dublincore.org/>. Acesso 28/11/2014

Figura 9 Diagrama que ilustra as classes e propriedades do vocabulário DCAT



Fonte: (MAALI; ERICKSON, 2014)

As classes da Figura 9 utilizadas nesta abordagem são descritas a seguir:

- *dcat:Catalog* (Catálogo) - Corresponde a um catálogo de dados.
- *dcat:Dataset* (Conjunto de dados) - Corresponde a uma coleção de conjuntos de dados pertencente a um catálogo. Em um *dataset* pode existir nenhuma ou várias distribuições de dados.
- *dcat:Distribution* (Distribuição) - Corresponde à referência dos dados propriamente ditos. Na distribuição, são encontradas propriedades como URL, representação do dado, data de publicação, descrição e outras que contribuem para sua identificação.

As propriedades dessas classes *dcat:Catalog*, *dcat:Dataset* e *dcat:Distribution* podem ser vistas na Tabela 10. Cada propriedade possui um breve comentário que permite descrever a propriedade utilizada nessas classes.

Tabela 10 Propriedades das classes *dcat:Catalog*, *dcat:Dataset* e *dcat:Distribution*

Propriedade	**Detalhe**
dct:identifier	Referência não ambígua ao recurso dentro de um dado contexto. *Token* usado para criar URI.
dct:title	Descreve o título de um dado contexto.
dct:description	Descreve detalhes sobre um dado contexto.
dct:issued	Data da publicação
dct:modifield	Última data de alteração
dct:creator	Descreve a entidade principal responsável.
dct:language	Idioma dos dados.
dct:license	Licença sobre os conjuntos de dados.
dct:rights	Direitos de utilização e reutilizado.
dct:spatial	Utilizado para denotar características espaciais.
foaf:homepage	Página da Web sobre o assunto.
dcat:keyword	Palavras chaves ou tags que descrevem o recurso.
dcat:contactPoint	Vincular um conjunto de dados para informações de contato relevantes que é fornecido usando *VCard*.
dct:temporal	Período de validade dos dados.
dct:spatial	Denota características espaciais.
dct:accrualPeriodicity	Frequência com que o dado é publicado.
dcat:landingPage	Página da *Web* com informações adicionais.
dcat:accessURL	Página da *Web*, terminal *Protocol and RDF Query Language* (SPARQL) ou outro tipo de recurso que dá acesso à distribuição do conjunto de dados.
dcat:downloadURL	Um arquivo que contém a distribuição do conjunto de dados em um determinado formato.
dcat:mediaType	O tipo de mídia da distribuição, conforme definido pela *Internet Assigned Numbers Authority* (IANA).
dcat:format	O formato do arquivo.
dcat:byteSize	O tamanho do arquivo em *bytes*.

Fonte: o autor

A utilização deste vocabulário para catalogar os dados na abordagem é constituído da seguinte maneira: a classe *dcat:Catalog* pode ser utilizada para agrupar um domínio de conhecimento, corporação, departamento ou outra grande área que associe uma coleção de conjuntos de dados. A classe *dcat:Dataset*, por sua vez, é utilizada para descrever um conjunto de distribuições. No DCAT, distribuição é uma referência para um arquivo ou ponto de acesso na rede capaz de fornecer recursos de dados. Por fim, a classe *dcat:Distribution* possui as propriedades necessárias para catalogar um arquivo ou ponto de acesso na rede. Todas essas classes mencionadas possuem suas propriedades na Tabela 10. A Figura 10, apresenta um exemplo de utilização do DCAT para criar um catálogo de dados de nome "*CIn*". A notação utilizada para criar o documento foi RDF/XML. Na prática, este documento também poderá ser utilizado para responder consultas SPARQL.

Figura 10 Exemplos de utilização do vocabulário *DCAT*

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
 xmlns:time="http://www.w3.org/2006/time#"
 xmlns:dct="http://purl.org/dc/terms/"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:dcat="http://www.w3.org/ns/dcat#"
 xmlns:foaf="http://xmlns.com/foaf/0.1/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<dcat:Catalog rdf:about="http://localhost:8080/api/v4/catalog/cin">
  <dct:identifier>cin</dct:identifier>
  <dct:title>Catlogo de dados sobre o CIn</dct:title>
  <dct:description>Este catalogo foi criado como exemplo para utilizacao do DCAT.</dct:↩
    description>
  <dct:issued rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2014-11-29T03↩
    :02:33Z</dct:issued>
  <dct:modified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2014-11-29T03↩
    :02:33Z</dct:modified>
 <dct:creator>bruno</dct:creator>
  <dc:language>BR</dc:language>
  <foaf:homepage rdf:resource="http://dados.recife.pe.gov.br/"/>
  <dct:license rdf:resource="http://www.opendatacommons.org/licenses/pddl/1.0/"/>
</dcat:Catalog>
</rdf:RDF>
```

Fonte: o autor

É possível observar na Figura 9 e na Tabela 10 alguns prefixos nas propriedades e nomes de classes. Estes prefixos são utilizados para denotar *namespaces* e por sua vez a especificação de um vocabulário. Nesta abordagem proposta o prefixo adotado para especificar as propriedades particulares é nomeado como "*b*". Entretanto, são utilizados outros termos que pertencem a outros prefixos, mas a maioria encontram-se no *namespace* do DCAT e *Dublin Core*. O conjunto completo de prefixos e *namespaces* utilizados nesta abordagem proposta é encontrado na Tabela 11.

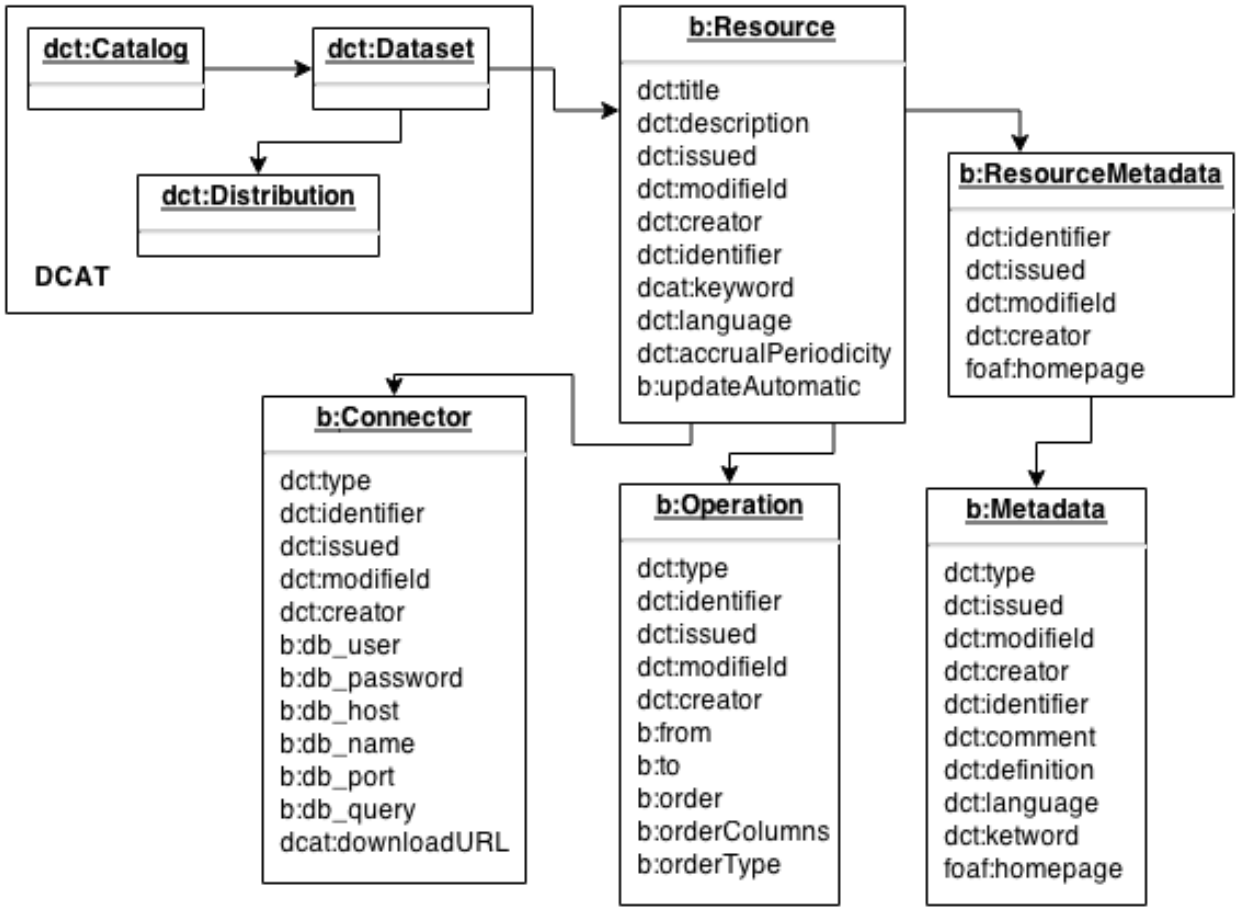
Tabela 11 Prefixos e *Namespaces*

Prefixo	**Namespace**
dct	*http://purl.org/dc/terms/*
dcat	*http://www.w3.org/ns/dcat#*
b	*http://cin.ufpe.br/ bifm/pubs/master/dissertation/ns/terms#*
foaf	*http://xmlns.com/foaf/0.1/*

Fonte: o autor

A Figura 11 apresenta o diagrama de classes e propriedades do vocabulário **b** criado para ser utilizado nesta abordagem. Este, por sua vez faz reuso das especificações de outros vocabulários. O vocabulário *Dublin Core* possui o maior conjunto desse reaproveitamento das propriedades. Todas as propriedades que começam com o prefixo "*dct:*" pertencem ao *Dublin Core*. O vocabulário **b** tem por objetivo fornecer uma melhor expressividade na descrição das principais propriedades, classes e seus respectivos relacionamentos adotados para descrever os recursos em alto nível.

Figura 11 Diagrama que ilustra as classes e propriedades do vocabulário b



Fonte: o autor

As principais classes do vocabulário **b** serão apresentadas a seguir:

- ***b:Resource*** (recurso) - Diz respeito às propriedades que descrevem o conjunto de dados. Em especial as propriedades *b:updateAutomatic* e *dct:accrualPeriodicity* que fazem referência a permitir ou não o processo de atualização automática e a segunda propriedade corresponde ao tempo para o conjunto de dados ser atualizado.
- ***b:Connector*** (conector) - Diz respeito à configuração de um conector de dados. Três conectores foram adotados por esta abordagem. São eles:
- ***b:Operation*** (operação) - Esta classe possui as propriedades adotadas para realizar as operações de transformação.
- ***b:ResourceMetadata*** (*metadados* do recurso) - Diz respeito aos *metadados* sobre o conjunto de dados extraídos. Esta classe possui as propriedades que são utilizadas para referenciar um *metadado* que possua uma URI que o descreva.
- ***b:Metadata*** (repositório de *metadados*) - Esta classe corresponde ao repositório central de *metadados*. A abordagem propõe que esse repositório seja utilizado para enriquecer os *metadados* dos conjuntos de dados. Cada *metadado* possuirá uma URI que o descreve de maneira detalhada. As propriedades desta classe podem ser vistas na Figura fig:cap3:vocabulariob.

Com esse repositório central de *metadados*, é possível reaproveitar suas descrições nos futuros conjuntos de dados, contribuindo também com futuras integrações e processos capazes de automatizar a escolha dos *metadados* dos conjuntos de dados.

As propriedades das classes *dcat:Resource*, *b:Connector*, *b:Operation*, *b:ResourceMetadata* e *b:Metadata* podem ser vistas na Tabela 12. As definições apresentadas para as propriedades das classes *dcat:Catalog*, *dcat:Dataset* e *dcat:Distribution* foram omitidas desta Tabela 12.

Tabela 12 Propriedades das classes do vocabulário *b*

Propriedade	**Detalhe**
b:db_user	Nome de usuário para acessar o SGBD.
b:db_password	Senha do usuário para acessar o SGBD.
b:db_host	*Host/Ip* para acessar o SGBD.
b:db_name	Nome da base de dados existente no SGBD.
b:db_port	Porta de acesso ao SGBD.
b:db_query	*Query* para ser executada no SGBD.
b:from	Denota origem.
b:to	Denota destino.
b:order	Ordem dos dados.
b:orderColumns	Ordem das colunas.
b:orderType	Tipo de ordem a ser considerado (crescente ou decrescente).
b:updateAutomatic	Processo de atualização automático. O valor padrão é desativado.
dcat:definition	Define o dado.
dcat:comment	Define um comentário para o dado.

Fonte: o autor

O vocabulário *b* foi criado para contribuir com a descrição da abordagem proposta. Este vocabulário *b* incorpora as principais classes e propriedades necessárias para a descrição desta abordagem proposta. Estas classes do vocabulário *b* são a base para serem transformadas em serviços *Web* capazes de oferecer as funcionalidades necessárias para a implementação da abordagem proposta.

Após o processo de catalogação de dados, agora é realizado o processo de manipulação de dados descrita a seguir.

A estratégia adotada por esta abordagem para manipular os dados corresponde às atividades de extração, transformação, carga e criação dos *metadados*. Primeiramente, é realizada a extração dos dados de acordo com a fonte de dados de origem. Consequentemente, estes dados precisam ser armazenados em uma estrutura de dados com o objetivo de serem utilizados posteriormente. Após isto, caso necessário, são realizadas as operações de transformações nos dados extraídos. Por fim, com o intuito de disponibilizar os dados extraídos e transformados, são utilizadas representações de dados.

- Extração - Esta atividade é responsável por extrair os dados da fonte de dados de origem. À

medida em que ocorre o progresso da extração dos dados, são salvos em uma estrutura de dados que permita a execução das demais atividade do processo de manipulação.

- Transformação/Limpeza - Corresponde à atividade capaz de modificar a estrutura e os dados extraídos. Estas modificações podem ser por exemplo de conversão, seleção ou ordenação. A primeira realizar substituições de valores, substituindo um dado valor em outro, dar-se da seguinte maneira: todos os dados pertencentes a uma determinada coluna serão consultados. Caso seja encontrado o valor definido como "valor original", o valor será substituído por "novo valor". No segundo tipo de operação "seleção", são selecionadas as colunas dos dados extraídos. Desse modo, é possível retirar o excesso de colunas, limpando o conjunto de dados extraídos. Em último lugar, a operação de "ordenação", com ela é possível optar por duas estratégia de ordenação com base uma coluna: crescente e decrescente.
- Carregamento - Esta atividade é responsável por fornecer os dados extraídos e transformados por meio de diferentes representações de dados. Independente do tipo da fonte de dados de origem, os dados serão carregados em diferentes representações de dados. Por fim, cada conjunto de dados extraído será acessível por meio de uma URI amigável.
- *Metadados* - Esta atividade se divide em duas: criação e uso dos *metadados*. Primeiramente, são criados os *metadados* no repositório central de dados. Cada metadado possui uma URI amigável, desse modo, eles poderão ser reutilizados. Em seguida, são utilizadas as URIs dos *metadados* para enriquecer os dados dos conjuntos de dados. Esse processo acontece associando cada coluna dos dados extraídos para uma URI que descreva os dados da coluna.

Após a etapa de manipulação, são disponibilizados os dados e *metadados* que compõem o processo de publicação de dados. A estratégia adotada para disponibilizar os dados é baseada em URI, cada conjunto de dados possui a sua. Com esta URI, é possível fornecer uma descrição prévia do conjunto de dados em sua especificação (URI amigável).

Em síntese, o conjunto de dados e *metadados* são disponibilizados na Web por meio da URI. Através desta, os consumidores de dados especificam qual o formato de representação dos dados desejado, além de servir também para os *metadados* a pluralidade das representações de dados. Esta abordagem propõe a utilização do padrão para escrita da URI como segue:

- *URI/metadata*: Por exemplo, a URI é *http://cin.ufpe.br/cin/professora/bernadette-farias-loscio*. Essa URI retorna o conjunto de dados, para solicitar os *metadados* dessa URI basta acrescentar "/metadata", resultando em "*http://cin.ufpe.br/cin/professora/bernadette-farias-loscio/metadata*".

A estratégia de incluir na URI parte do pedido é adotada para definir as representações de dados. Por exemplo: para recuperar o conjunto de dados em notação csv na URI *http://cin.ufpe.br/cin/professora/be*
farias-loscio, basta acrescentar o "/csv" ao final da URI *http://cin.ufpe.br/cin/professa/bernadette-farias-loscio/csv*. Para solicitar os *metadados* em notação csv ficará *http://cin.ufpe.br/cin/professora/bernadette-farias-loscio/metadata/csv*.

Analisando a URI base adotada, "*http://cin.ufpe.br/cin/professora/bernadette-farias-loscio*", será apresentado o catálogo, dataset e descrição do conjunto de dados. O primeiro "diretório"

representado por “cin”, corresponde ao catálogo de dados criado para organizar o restante dos dados. Em seguida, “*/professor/*”, corresponde ao *dataset* criado para organizar as distribuições de dados, no caso a distribuição é “*bernadette-farias-loscio*”.

A próxima atividade é a de atualização dos dados. Esta é possível devido à integração de todas as atividades mencionadas anteriormente. Especificamente, são as atividades de extração e transformação que são executadas e, ao final desta execução, os conjuntos de dados passam por todo o processo de manipulação e publicação de dados novamente e são atualizados sem a intervenção do usuário.

Como visto até o presente momento, o processo de publicação proposto possui várias atividades. Desse modo, ao final de todas as etapas, as mesmas são compiladas e integradas. O resultado desta integração é um conjunto de dados que seguem os padrões para publicação de dados abertos o qual poderá ser utilizado por desenvolvedores de *software* para a criação de novas aplicações que utilizam os dados abertos.

Salienta-se ainda que, para o usuário final, todo o processo realizado por esta abordagem é transparente e possibilita uma redução considerável do esforço empregado para a publicação de dados abertos, permitindo também que os dados possuam uma frequência de atualização. Para isso, é preciso definir uma periodicidade do conjunto de dados e, então, todas as atividades de manipulação e publicação serão executadas de forma automática.

3.3 Conclusões

Este capítulo apresentou a abordagem proposta. Primeiramente, apresentando um conjunto de características consideradas essenciais para esta abordagem proposta e, posteriormente, foram analisadas quais das ferramentas apresentadas no capítulo 2 possuem características relevantes para a abordagem proposta. Posteriormente, foi apresentado um conjunto de requisitos desejáveis a serem incorporados na abordagem proposta. Estes requisitos tratam aspectos desejáveis para o processo de publicação e uso de dados abertos. Em seguida, foi apresenta a arquitetura da abordagem proposta por meio de uma visão geral. Por fim, foram apresentados os detalhes e conclusões da proposta dessa dissertação.

4 Implementação

Não importa o quanto você bate, mas sim o quanto aguenta apanhar e continuar. O quanto pode suportar e seguir em frente. É assim que se ganha.

—SYLVESTER STALLONE - ROCKY BALBOA

Este capítulo tem por objetivo descrever os detalhes técnicos da implementação desta abordagem a qual foi apresentada no capítulo 3 desta dissertação. São apresentados os detalhes sobre tecnologias, padrões, decisões e definições adotadas para implementação da abordagem proposta. Em seguida, são apresentados os recursos disponibilizados por esta abordagem para publicação e uso dos dados abertos. Por fim, são vistas as conclusões deste capítulo.

4.1 Convenções (conformidade)

Algumas das principais decisões pertinentes à abordagem serão apresentadas nesta seção, dentre elas estão: *a*) Uso do estilo arquitetural REST (FIELDING, 2000) com o protocolo HTTP/1.1 especificado na RFC 2616[1]; *b*) Notações JSON especificada na RFC 4627[2] e XML[3] para comunicação (pedidos realizados do cliente para o servidor); *c*) Credenciais de usuários por meio de chave pública, privada, *timestamp*[4] e *hash* da mensagem, utilizando algoritmo *Keyed-Hashing for Message Authentication* (HMAC) especificado na RFC 2104[5] com criptografia *Secure Hash Algorithm 1* (SHA1) especificado na RFC 3174[6]; *d*) Comunicações síncronas entre cliente e servidor; e *e*) Múltiplas representações para recursos de dados. Serão explicitados cada uma das escolhas a seguir.

A escolha do estilo arquitetural REST dá-se pela simplicidade e atual interesse pela comunidade para uso na *Web*. O padrão SOAP-WSDL foi objeto de estudo e apresentado na seção 2.3, apesar deste padrão ser uma recomendação do W3C para comunicação de mensagem (cliente-servidor) na *Web*, optou-se por usar o REST. Alguns dos ponto fortes considerados para adotação do

[1] <http://tools.ietf.org/html/rfc2616>. Acesso: 01/10/2014
[2] <http://tools.ietf.org/html/rfc4627>. Acesso: 01/10/2014
[3] <http://www.w3.org/TR/REC-xml/>. Acesso: 01/10/2014
[4] <http://www.unixtimestamp.com/>. Acesso: 01/10/2014
[5] <http://tools.ietf.org/html/rfc2104>. Acesso: 01/10/2014
[6] <http://tools.ietf.org/html/rfc3174>. Acesso: 01/10/2014

REST nesta abordagem são: *a*) Ele foi concebido logo após a especificação dos protocolos da *World Wide Web* (WWW); *b*) É um estilo arquitetural de *software* para sistemas de hipermídia distribuídos, como a WWW; *c*) Ele fornece uma abordagem centralizada sobre os recursos em vez de funções. Isso quer dizer que em REST as operações são executadas em um recurso por meio de interface uniforme, não sendo necessário criar vários nomes que representem as funcionalidades. Como exemplo, é possível citar o REST com HTTP, em que os métodos do HTTP operam os recursos; *d*) O sucesso das *RestfulL* também deve ser considerado. De acordo com FERNANDEZ (2009, p. 2), este sucesso pode ser observado na proliferação de *Mashups*[7]. Estas aplicações são produzidas pela combinação de funcionalidade remota que podem ser disponíveis por meio de *Restful* API; *e*) Por ser menos burocrático e exigir menos formalismo para implementação; *f*) Pela relevância do interesse do termo na *Web*; *g*) Os *Web services* com REST são mais manuteníveis do lado do servidor em comparação aos *Web services* SOAP-WSDL Oliveira (2012); *h*) Possuem identificação dos recursos por URIs; e *i*) Interface uniforme, padronização das operações de criação, leitura, atualização, exclusão por meio dos métodos (*POST*, *GET*, etc...) do protocolo HTTP.

Esta abordagem prevê suporte para comunicação de mensagem por XML e JSON. O padrão adotado para troca de mensagem é JSON, sendo flexível ao uso da notação XML.

A serialização (ou também conhecida como *Marshalling*) com JSON tem por característica conseguir reduzir o tamanho da mensagem em comparação às mensagens serializadas com XML. Esta particularidade é refletida na carga das mensagens trafegadas na rede, tendo impacto direto no processamento das mensagens.

A validação das credenciais de usuário dá-se por meio do algoritmo HMAC com criptografia SHA1. Este método utiliza conjunto com chaves que são enviados ao servidor junto com a mensagem. Ele é melhor explicado na seção 4.2.3. Levando em consideração a restrição do estilo arquitetural REST de não manter estado do lado do servidor, outros meios que utilizam sessão ou *Cookie* (mecanismo de gestão de estado com HTTP)[8] foram descartados.

Padrões como *OpenID*[9] e oAuth[10] foram estudados, prevalecendo o método HMAC com criptografia SHA1. O *OpenID*, assim como o *oAuth*, possue como característica o compartilhamento de credenciais por meio de *token* (chave). A particularidade do *oAuth* é a capacidade de validar credenciais de autorização a diferentes recursos. Não sendo característica do projeto desta dissertação o compartilhamento de credenciais, desta forma optou-se pelo método HMAC com criptografia SHA1.

Por natureza da especificação do HTTP, ele tem um modelo de solicitação e resposta baseado em operações síncronas. O cliente estabelece uma conexão com o servidor de aplicação Web, submete seu pedido, por meio de mensagem HTTP e mantém aberto o canal de comunicação até que o servidor envie uma resposta. Se o cliente não estiver interessado em obter resposta do servidor, essa comunicação pode ser encerrada pelo cliente a qualquer momento.

As múltiplas representações dos dados fornecem maior interoperabilidade sintática aos

[7] Mashups são aplicações construídas por vários componentes de diferentes tipos: *Web services*, fontes de dados da *Web* e *Web widgets* (AGHAEE; PAUTASSO, 2011, p. 311).

[8] <http://tools.ietf.org/html/rfc6265>. Acesso: 01/10/2014

[9] <http://openid.net/developers/specs/>. Acesso: 01/10/2014

[10] <http://tools.ietf.org/html/rfc6749>. Acesso: 01/10/2014

clientes. As representações das estruturas de dados fornecidas (HTML, XML, JSON e CSV) para serializar os documentos em arquivos com diferentes notações contribuem também para o processamento por máquinas automático.

4.2 Mensagem HTTP

É adotada a versão 1.1 do protocolo HTTP, portanto, qualquer uso deve estar em conformidade com os requisitos do mesmo. De acordo com Fielding *et al.* (1999), o HTTP é um protocolo de nível de aplicação para sistemas distribuídos, de colaboração e baseados em hipermídia.

Por definição, ele é um protocolo genérico, sem estado e possui diversas aplicações. Uma destas aplicações largamente utilizada é seu uso para hipertexto. O HTTP possui extensão de seus métodos de solicitação (comuns são *POST* e *GET*), códigos de erro e cabeçalhos para troca de mensagens. Outra característica do HTTP está na facilidade da representação de dados. Estas características apresentadas são pertinentes e essenciais para a proposta desta dissertação.

4.2.1 Formato das Mensagens

São oferecidos suporte a várias representações de dados. Elas possibilitam que dados sejam transmitidos em um pedido ou recebido como resposta. Além disso, são aceitas as apresentações em XML e JSON para comunicação do cliente para o servidor.

A seleção da representação de dados para obter como resposta acontece por solicitação do cliente, por duas maneiras. A primeira é por meio do cabeçalho da mensagem, através da propriedade "*Content-Type*". A segunda acontece por meio da inserção direta do formato no final da URL.

Cada uma destas representações de dados é identificada por um tipo de mídia específica. Os tipos de mídia suportados são descritos na Tabela 13 a seguir e seguem conformidade do Freed, Baker e Hoehrmann (2014).

Tabela 13 Representações de dados suportados

Representação	Detalhe
HTML	*text/html*
XML	*application/xml*
JSON	*application/json*
CSV	*text/csv*

Fonte: o autor

4.2.2 Corpo de Mensagem

Com base no tipo de solicitação que está sendo executada, o corpo da mensagem é alterado. O conteúdo da mensagem do pedido é descrito e informado no cabeçalho por "*Content-Type*", que deve ser um dos tipos de mídia apropriadas para este tipo de solicitação (JSON ou XML).

Caso a solicitação (requisição HTTP) deva retornar um corpo de mensagem contendo representação diferente da padrão (*application/json*) fornecida pelo *Web service*, ele deve ser explicitado. Na requisição solicitada pelo cliente, deve ser incluído o parâmetro de cabeçalho HTTP "*Accept*", em que será descrito o tipo de mídia ou tipos de mídia aceitas como resposta ao pedido. O cabeçalho de resposta deve retornar com a propriedade "*Content-Type*" e formato retornado. Por exemplo, parâmetros de cabeçalho: "*Accept: text/html, text/csv*". Neste caso, o *Web service* respeita a ordem estabelecida no parâmetro (primeiro "*text/html*" e depois "*text/csv*") e responde ao cliente de acordo com seu suporte. Caso não haja suporte, será retornado apenas um código de estado HTTP.

4.2.3 Autenticação de Mensagem

A autenticação está atrelada à confirmação da identidade. A procedência do usuário será verificada por meio da combinação de variáveis que identificam a veracidade da mensagem e usuário.

A autorização acontece por meio de chaves criptográficas associadas à conta do usuário. Cada usuário com permissão de acesso privado aos recursos do *Web service* possui chave privada e pública. Para cada pedido que exija credencial, devem ser enviadas no cabeçalho HTTP a chave pública (chamada de "*x-public-id*" internamente), *timestamp* (chamada de "*x-microtime*" internamente) e *hash* da mensagem (chamada de "*x-message*" internamente) com inclusão de *timestamp* ao final. Esse conjunto de parâmetros contribui para validação da credencial do usuário e segurança do sistema.

A chave[11] da mensagem é gerada usando o algoritmo HMAC[12] e criptografia SHA1. Para gerar o *hash* da mensagem o algoritmo tem como entrada o conteúdo da mensagem mais o valor de *timestamp* (o mesmo enviado na requisição) e a chave privada. A mensagem enviada ao servidor será validada realizando a mesma criptografia do lado do servidor, a fim de confirmar o *hash* da mensagem (*x-message*) e as outras informações.

A próxima seção apresenta a implementação dos serviços da abordagem proposta. Por fim, outros detalhes sobre mensagem do HTTP foram incluídas no Apêndice A.

4.3 Visões da Arquitetura

Em computação, a palavra Arquitetura está intimamente ligada a tudo o que envolve o *software* ou sistema e seu ambiente. Em um *software*, existem componentes de competências internas e externas. Na primeira, destacam-se a linguagem de programação utilizada e quais padrões de projeto foram adotados para o desenvolvimento. Na última, a plataforma em que ele é executado e quais partes necessitam se comunicar com outro *software*.

Para Kruchten (2003, p. 288), o termo arquitetura pode ser entendida como um conjunto de decisões significativas sobre a organização de um sistema de *software*.

Existem várias formas de representar uma arquitetura. Entre elas, é comum o uso de

[11] Baseada na estrutura de dados "*tabela hash*", a estrutura possui um mapeamento contendo chave e valor para cada entrada. Página *Web* <https://www.cs.auckland.ac.nz/software/AlgAnim/hash_tables.html>

[12] <http://tools.ietf.org/html/rfc2104>. Acesso: 01/10/2014

apenas um diagrama ou documento. Entretanto, estas representações não expressam com detalhes os componentes envolvidos e não são capazes de transmitir todas as preocupações necessárias para mapear os requisitos em um modelo de padrão de projeto e desenvolvimento de *software*. É preciso levar em conta e descrever com detalhes todos os componentes envolvidos para definição de uma arquitetura.

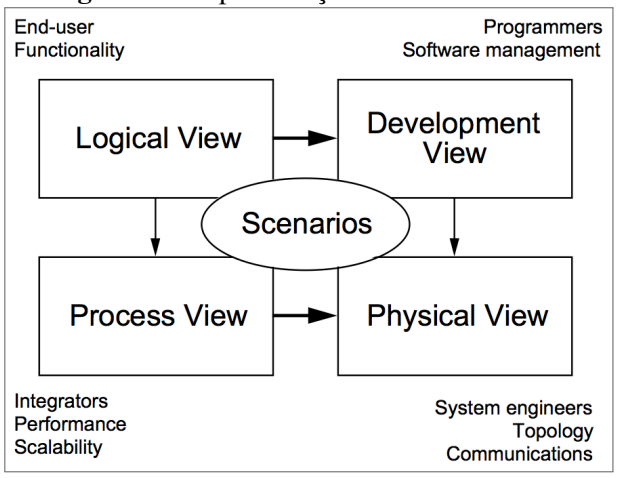
Uma forte motivação para a especificação de uma arquitetura é o desejo de obter o sucesso, seja reduzindo problemas conceituais e ou melhorando o entendimento do sistema para os envolvidos. Na elaboração da arquitetura, várias atividades estão incluídas. Dentre elas, destaca-se a atividade de identificação dos requisitos do sistema. Ainda nesta fase, é possível corrigir possíveis falhas provenientes do entendimento do sistema, e o arcabouço conceitual, definido na arquitetura, organiza os elementos do mesmo.

Esta seção aborda a representação da arquitetura da abordagem proposta por meio de perspectivas de alto nível, por meio de visões. Nesta fase arquitetural, é importante considerar características do sistema em partes menores, separadamente, mas também combinadas. Ao considerar o sistema em partes menores, esta separação contribui na abstração da ideia, simplificando também o entendimento da combinação com outras partes. Diferentes cenários podem ser testados, através da combinação destas partes, permitindo novas visões.

Nesta dissertação é adotado o modelo de visão "4+1" Kruchten (2003, p. 288) para descrição da arquitetura. O modelo 4+1 propõe a descrição da arquitetura do *software* por cinco diferentes pontos de vista: *Logical View*(Visão Lógica), *Process View*(Visão de Processo), *Physical View* (Visão Física), *Development View* (Visão de Desenvolvimento) e *Scenarios* (Cenários de caso de uso). Ele propõe cortar o sistema por meio destes diferentes pontos de vista para fornecer diferente representações da arquitetura do sistema.

A Figura 12 apresenta ilustração que representa simbolicamente os diferentes pontos de vista do modelo 4+1.

Figura 12 Representação do modelo "4+1"



Fonte: Kruchten (1995)

Boa parte dos conceitos empregados no modelo "4+1" foram incluídos em processos de desenvolvimento, como por exemplo, *International Business Machines* (IBM) *Rational Unified*

Process (RUP)[13] ou OpenUP[14] (Processo de código aberto que aplica abordagens iterativo e incremental dentro de um ciclo de vida estruturado). Existe também uma padronização elaborada pelo *Institute of Electrical and Electronics Engineers* (IEEE) 1471[15] que define o objetivo de abordar as preocupações das várias pessoas envolvidas, a fim de definir uma arquitetura de *software* (ISO/IEC, 2011, p. 10–12).

Esta seção apresentou detalhes sobre as visões arquiteturais e visão geral da proposta. Nas próximas subseções, serão apresentados por dois pontos de vista para representação da arquitetura da abordagem. Foram priorizados para apresentação dos pontos de vista as visões de cenário e lógica.

4.3.1 Visão baseada em Cenários

De acordo com Kruchten (2003, p. 288), um cenário pode ser entendido como a instância de um caso de uso ou um subconjunto de um caso de uso. Os cenários são representados por casos de uso (de forma textual ou por diagramas). Eles descrevem as transações e contribuem para facilitar o entendimento das funcionalidades. Possuem maior nível de abstração quando comparados às tradicionais listas de requisitos.

Os casos de uso utilizados neste trabalho estão representados de duas formas: textual e gráfica. Na primeira, foram organizados na Tabela 14, em particular são agrupados por nome e detalhe. A segunda pode ser vista na Figura 13. A última possui inclusão dos atores, sendo esta uma das principais vantagens em representar de forma gráfica. Ela contribui também para um entendimento com alto nível de abstração.

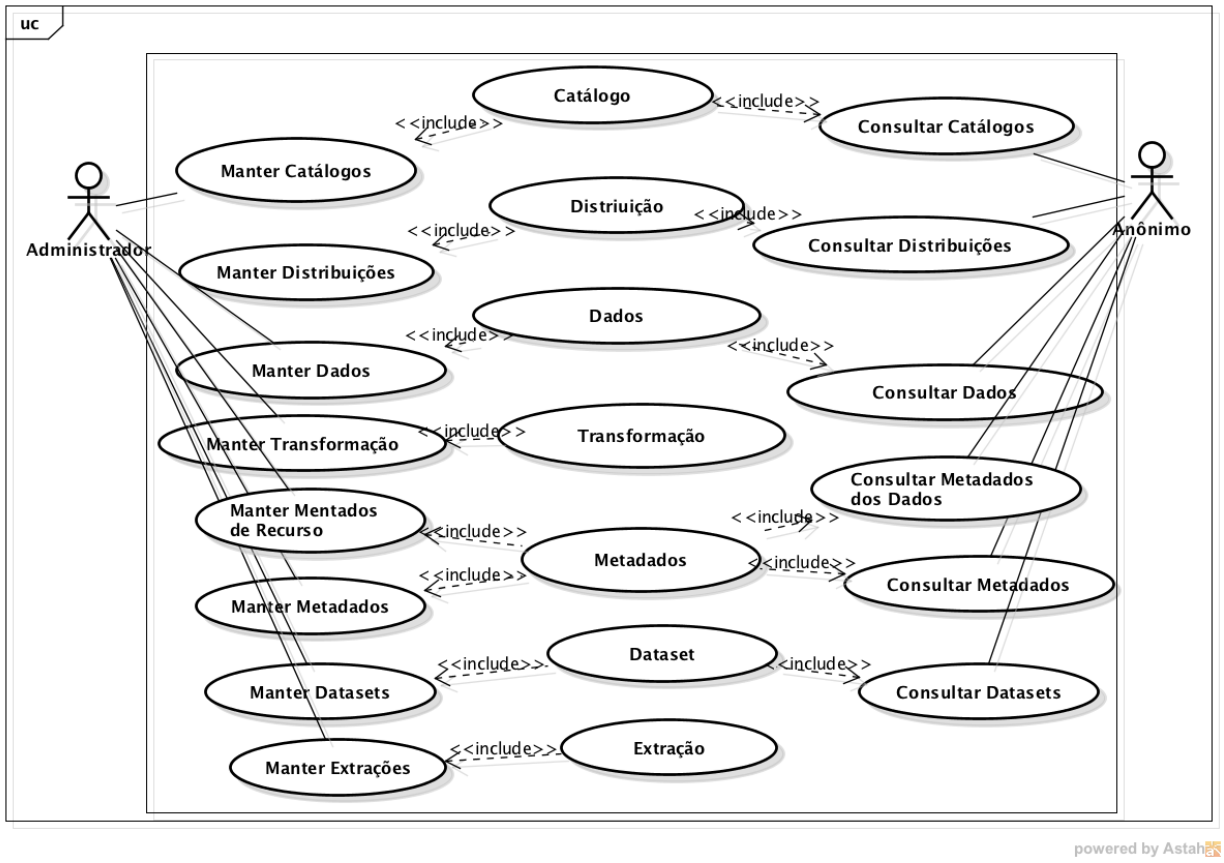
Tabela 14 Caso de Uso das principais funcionalidades do sistema

Nome	**Detalhe**
Manter Catálogos	Responsável por criar, ler, atualizar e excluir catálogos.
Manter *Dataset*	Responsável por criar, ler, atualizar e excluir *datasets*.
Manter Dados	Responsável por criar, ler, atualizar e excluir dados.
Manter Distribuições	Responsável por criar, ler, atualizar e excluir distribuições.
Manter Extrações	Responsável por criar, ler, atualizar, excluir e executar os conectores de extração de dados.
Manter Transformações	Responsável por criar, ler, atualizar e excluir as operações de transformação.
Manter *Metadados* dos Dados	Responsável por criar, ler, atualizar e excluir *Metadados* dos dados.
Manter *Metadados*	Responsável por criar, ler, atualizar e excluir *metadados*.
Consultar *Metadados*	Responsável por consultar e ler *metadados*.
Consultar *Metadados* dos dados	Responsável por consultar e ler *metadados* dos dados.
Consultar Catálogos	Responsável por consultar e ler catálogos.
Consultar *Datasets*	Responsável por consultar e ler *datasets*.
Consultar Distribuições	Responsável por consultar e ler distribuições.
Consultar Carregamentos	Responsável por consultar e ler carregamentos.

Fonte: o autor

[13] <http://www-01.ibm.com/software/rational/rup/>. Acesso: 01/10/2014
[14] <http://epf.eclipse.org/wikis/openup/>. Acesso: 01/10/2014
[15] <http://www.iso-architecture.org/ieee-1471/>. Acesso: 01/10/2014

Figura 13 Caso de Uso da Abordagem



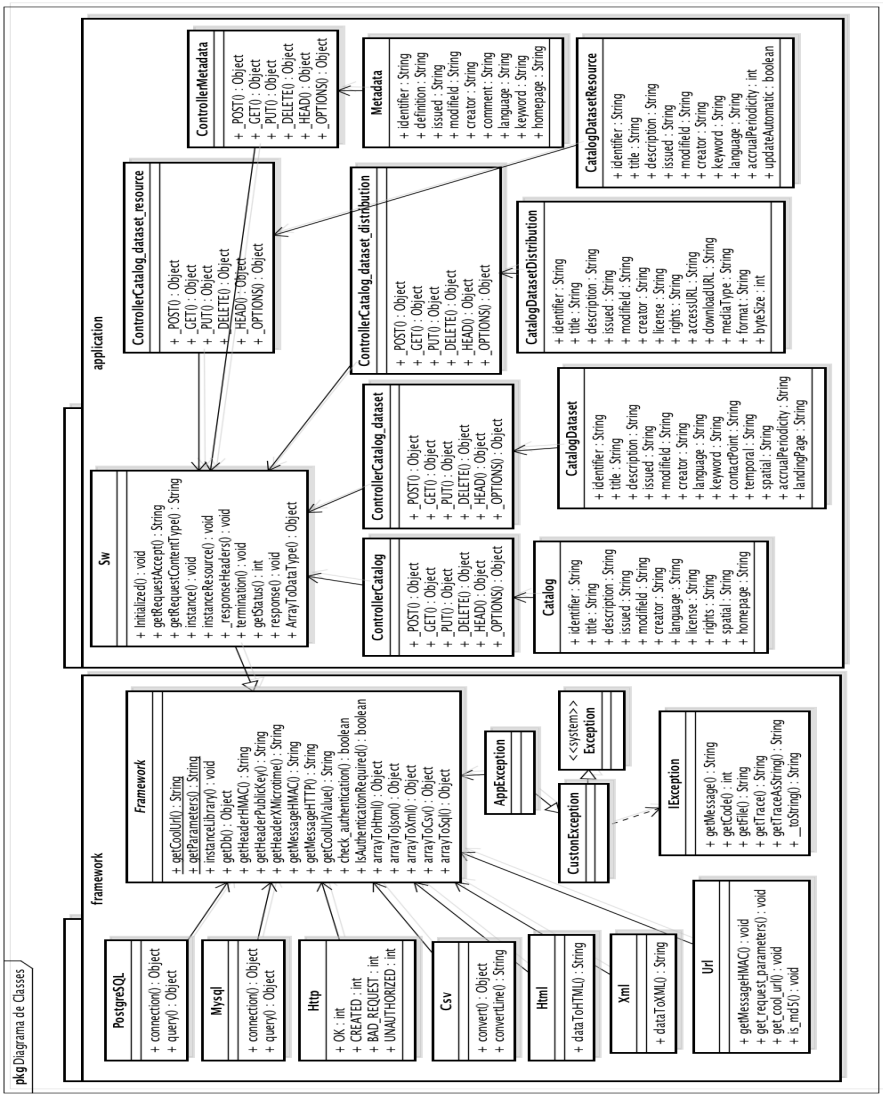
Fonte: o autor

A Tabela 14, apresentou as principais funcionalidades pertinentes a abordagem. Esse fato de abstrair detalhes da forma implementada tem por objetivo ocultar detalhes técnicos e tornar o caso de uso de fácil leitura e interpretação. A seguir, será apresentado o ponto de vista lógico.

4.3.2 Visão Lógica

De acordo com Kruchten (1995), na visão lógica, o sistema pode ser representado por diagramas de classes e sequência, requisitos funcionais e não funcionais. As principais funcionalidades estão organizadas no diagrama de classes da Figura 14. Para elaboração das classes deste diagrama, algumas informações foram omitidas ou removidas, tendo como objetivo fornecer o entendimento geral e seus relacionamentos.

Figura 14 Diagrama de Classes



Fonte: o autor

Na Figura 14, são apresentadas as principais classes que implementam a abordagem proposta, deste modo foram priorizadas as classes que permitem representar uma visão lógica da estrutura da implementação do código. Foram omitidos alguns detalhes técnicos como atributos, métodos e outras bibliotecas, com o intuito de abstrair o código desenvolvido. Serão descritas brevemente as classe *Framework*, *Sw*, *ControllerCatalog* e *Catalog* a seguir:

- *Framework*: responsável por coordenar um conjunto de funcionalidades, reutilizando a estrutura criada. Permite também estender as funções existentes por meio de novas incorporações de bibliotecas. Possui o controle de autenticidade das mensagens.
- *Sw*: implementa a estrutura responsável por instanciar os serviços Web. Possui várias atribuições, dentre elas, a verificação das mensagens recebidos, a conversão dos dados para representação de destino.
- *ControllerCatalog*: responsável por processar as operações relacionadas ao catálogo de dados, são elas: criação, leitura, atualização, exclusão, metadado e consulta de métodos do HTTP

suportados pelo recurso.

- *Catalog*: possui como atributos de classe as propriedades da classe *dcat:Catalog* do vocabulário DCAT. Ela é utilizado para fazer o mapeamento das propriedades do vocabulário DCAT para a implementação.

A próxima seção, apresenta a implementação da abordagem por meio de URIs e métodos do HTTP.

4.4 Implementação dos Recursos

Esta seção apresenta a forma concreta da abordagem proposta no capítulo 3. Todos os serviços apresentados na abordagem foram implementados seguindo os princípios do REST. Nesta seção serão apresentados os serviços que permitem a catalogação dos dados, os demais serão incluídos no apêndice B.

Os serviços conceituados no capítulo 3, contribuem para descrição da abordagem proposta. Em REST, os serviços são tratados como recursos, deste modo os recursos apresentados podem ser entendido com um objeto ou serviço disponível em rede que pode ser identificado por uma URI (FIELDING *et al.*, 1999, p. 8).

Para Garlan *et al.* (2010, p. 494), o termo recurso é expresso como uma função, método de fluxo de dados, variável global, ponto final da mensagem, disparador do evento, ou qualquer instalação com endereço dentro de uma interface.

Tabela 15 Métodos do HTTP de suporte na abordagem

Abreviação	Método	Detalhe
C	*POST*	Utilizado para criar um novo estado no recurso.
R	*GET*	Utilizado para recuperar um único estado ou coleção de estados do recurso.
U	*PUT*	Utilizado para alterar informações em um estado de recurso.
D	*DELETE*	Utilizado para remover um estado do recurso.
H	*HEAD*	Utilizado para obter os detalhes sobre o recurso.
O	*OPTIONS*	Utilizado para obter a relação dos métodos do HTTP suportados pelo recurso.

Fonte: o autor

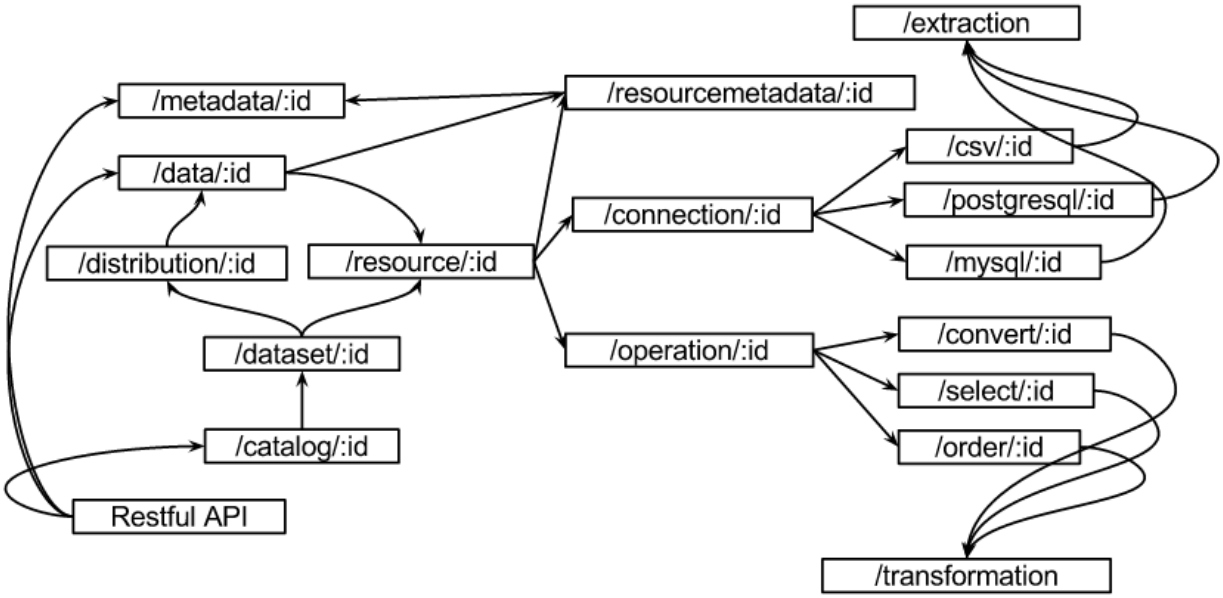
Todos os recursos da abordagem suportam os métodos HTTP: *OPTIONS* e *HEAD*. A Tabela 15, apresenta relação completa dos métodos do HTTP com suporte nesta abordagem. Com eles, os desenvolvedores de aplicações e agentes de softwares podem se beneficiar com a descoberta dos métodos suportados por cada recurso e por obter conjunto de *metadados* que auxiliam na descrição do recurso.

O método *HEAD* do HTTP não retorna corpo da mensagem[16]. Essa particularidade é explicitada na RFC 2616 Fielding *et al.* (1999, p. 8), sendo relevante o destaque dessa explicação,

[16]<http://www.pragmaticapi.com/2013/02/14/restful-patterns-for-the-head-verb/>. Acesso: 01/10/2014

por se tratar de um método pouco explorado pelo desenvolvedores e essa observação tem por interesse minimizar futuras dificuldades no desenvolvimento das aplicações e clareza da especificação deste capítulo.

Figura 15 Mapa da ligação dos recursos



Fonte: o autor

Os recursos estão representados de forma hierárquica na Figura 15 e simbolicamente, cada retângulo da imagem representa um recurso. Os recursos estão conectados entre si através de setas de acordo com suas relações e dependência dos recursos. A notação ":id", faz referência ao identificador único do estado do recurso. A seguir, serão apresentados os recursos para catalogação dos conjuntos de dados, os demais são apresentados no Apêndice B, como mencionado no início desta seção, deste modo foram omitidos desta seção com o intuito de tornar a leitura menos cansativa.

4.4.1 Recurso "*catalog*"

O recurso *catalog* é utilizado para catalogar uma coleção de *datasets* (coleção de conjunto de Dados). As propriedades (atributos) que descrevem este recurso seguem especificação das propriedades do vocabulário DCAT, a classe de referência para este recurso é "*dcat:Catalog*". Na Tabela 16, são apresentadas as URIs e métodos do HTTP suportados no recurso *catalog*.

Tabela 16 URIs do recurso *catalog* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog*	x	x		
2	*/catalog/:id*	x	x	x	

Fonte: o autor

Tabela 17 Propriedades do recurso *catalog*

Propriedade	**Restrição**	**Detalhe**
dct:identifier	obrigatório	Identifica o catálogo.
dct:title	obrigatório	Descreve o título do catálogo.
dct:description	obrigatório	Descreve detalhes sobre o catálogo.
dct:issued	automático	Data de publicação.
dct:modifield	automático	Última data de alteração.
dct:creator	automático	Descreve entidade principal responsável pelo catálogo.
dct:language	opcional	A linguagem do catálogo.
dct:license	opcional	Isso nos leva ao documento de licença sob a qual o catálogo é disponibilizado e não os conjuntos de dados. Mesmo se a licença do catálogo aplica-se a todos os seus conjuntos de dados e distribuição, deve ser replicado em cada distribuição.
dct:rights	opcional	Este descreve os direitos em que o catálogo pode ser utilizado / reutilizado e não os conjuntos de dados. Mesmo se os direitos desses se aplicarem a todos os conjuntos de dados de catálogo e distribuição, deve ser replicado em cada distribuição.
dct:spatial	opcional	A área geográfica abrangida pelo catálogo.
foaf:homepage	opcional	A página inicial do catálogo.

Fonte: o autor

As propriedades do recurso *catalog* são apresentadas na Tabela 17. Em particular as demais propriedades, a propriedade *dct:identifier* é utilizada para identificar unicamente um estado do recurso, deste modo, ela define a URI. Por exemplo, o catálogo com a propriedade *dct:identifier* = "*cin*", terá como URI *http://exemplo.com/catalog/cin*.

A próxima seção apresenta o recurso *dataset*, ele permite que coleções de conjuntos de dados sejam catalogadas.

4.4.2 Recurso "*dataset*"

O recurso *dataset* é utilizado para catalogar uma coleção de *distributions* (conjunto de Dados). As propriedades (atributos) que descrevem este recurso seguem especificação das propriedades do vocabulário DCAT, a classe de referência para este recurso é "*dcat:Dataset*". Na Tabela 16, são apresentadas as URIs e métodos do HTTP suportados no recurso *datset*.

Tabela 18 URIs do recurso *dataset* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset*	x	x		
2	*/catalog/:id/dataset/:id*	x	x	x	x

Fonte: o autor

Tabela 19 Propriedades do recurso *dataset*

Propriedade	**Restrição**	**Detalhe**
dct:identifier	obrigatório	Referência não ambígua ao recurso dentro de um dado contexto.
dct:title	obrigatório	Descreve o título do *dataset*.
dct:description	obrigatório	Descreve detalhes sobre o *dataset*.
dct:issued	automático	Data de publicação.
dct:modifield	automático	Última data de alteração.
dct:creator	automático	Descreve entidade principal responsável pelo *dataset*.
dcat:keyword	opcional	Palavras chaves ou *tags* que descrevem o recurso.
dct:language	opcional	O idioma do recurso.
dcat:contactPoint	opcional	Vincular um conjunto de dados para informações de contato relevantes que é fornecido usando *VCard*.
dct:temporal	obrigatório	O período temporal que abrange o conjunto de dados.
dct:spatial	opcional	A área geográfica abrangida pelo catálogo.
dct:accrualPeriodicity	opcional	A frequência com que conjunto de dados é publicado.
dcat:landingPage	opcional	A Uma página da *Web* que pode ser navegada para em um navegador da *Web* para ter acesso ao conjunto de dados, suas distribuições e / ou informações adicionais.
dct:publisher	opcional	informações sobre quem publicou os dados.

Fonte: o autor

As propriedades do recurso *catalog* são apresentadas na Tabela 17. Em particular as demais propriedades, a propriedade *dct:identifier* é utilizada para identificar unicamente um estado do recurso, deste modo, ela define a URI. Por exemplo, o catálogo e *dataset* com as respectivas propriedades *dct:identifier* = "*cin*" e *dct:identifier* = "*professor*", terá como URI *http://exemplo.com/catalog/cin/dataset/professor*.

A próxima seção apresenta o recurso *distribution*, ele permite que um conjunto de dados seja catalogado.

4.4.3 Recurso "*distribution*"

O recurso *distribution* é utilizado para catalogar conjunto de Dados. As propriedades (atributos) que descrevem este recurso seguem especificação das propriedades do vocabulário DCAT, a classe de referência para este recurso é "*dcat:distribution*". Na Tabela 20, são apresentadas as URIs e métodos do HTTP suportados no recurso *distribution*.

Tabela 20 URIs do recurso *distribution* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/distribution*	x	x		
2	*/catalog/:id/dataset/:id/distribution/:id*	x	x	x	x

Fonte: o autor

As propriedades do recurso *distribution* são apresentadas na Tabela 21. Em particular as

demais, a propriedade *dct:identifier* é utilizada para identificar unicamente um estado do recurso, deste modo, ela define a URI. Por exemplo, o catálogo, *dataset* e *distribution* com as respectivas propriedades *dct:identifier* = "*cin*" e *dct:identifier* = "*professor*", *dct:identifier* = "*bernadette-farias-loscio*", terá como URI *http://exemplo.com/catalog/cin/dataset/professor/bernadette-farias-loscio*.

Tabela 21 Propriedades do recurso *distribution*

Propriedade	**Restrição**	**Detalhe**
dct:identifier	obrigatório	Referência não ambígua ao recurso dentro de um dado contexto. Identifica unicamente o estado do recurso.
dct:title	obrigatório	Descreve o título da distribuição.
dct:description	obrigatório	Descreve detalhes sobre a distribuição.
dct:issued	automático	Data de publicação.
dct:modifield	automático	Última data de alteração.
dct:license	opcional	Isso nos leva ao documento de licença sob a qual o catálogo é disponibilizado e não os conjuntos de dados. Mesmo se a licença do catálogo aplica-se a todos os seus conjuntos de dados e distribuição, deve ser replicado em cada distribuição.
dct:rights	opcional	Este descreve os direitos em que o catálogo pode ser utilizado / reutilizado e não os conjuntos de dados. Mesmo se os direitos teses se aplicam a todos os conjuntos de dados de catálogo e distribuição, deve ser replicado em cada distribuição.
dct:creator	automático	Descreve entidade principal responsável pela distribuição.
dcat:accessURL	opcional	A página de destino, alimentação, terminal SPARQL ou outro tipo de recurso que dá acesso à distribuição do conjunto de dados.
dcat:downloadURL	obrigatório	Um arquivo que contém a distribuição do conjunto de dados em um determinado formato.
dcat:mediaType	obrigatório	O tipo de mídia da distribuição (FREED; BAKER; HOEHRMANN, 2014).
dcat:format	obrigatório	O formato do arquivo da distribuição.
dcat:byteSize	opcional	O tamanho de uma distribuição em *bytes*.

Fonte: o autor

Os demais recursos e detalhes podem ser encontrados no Apêndice B.

4.5 Conclusões

Este capítulo apresentou alguns dos aspectos relacionados a implementação da abordagem. Foram explanados detalhes técnicos sobre o HTTP, parâmetros de utilização empregados e tipos de representações de dados adotados na implementação da abordagem. Além disso, foi apresentada uma visão geral da arquitetura e uma breve descrição dos recursos que implementam os serviços definidos na abordagem. O capítulo seguinte deste trabalho apresenta a avaliação experimental.

5

Avaliação Experimental

...Agora, se você sabe o que vale a pena, então vá e consiga o vale a pena para você! Mas você tem que estar disposto a receber os golpes e não apontando dedos dizendo que você não está onde quer por causa "daquele", "daquela" ou de qualquer um! Só covardes fazem isso e você não é um! Você é melhor do que isto!

—SYLVESTER STALLONE - ROCKY BALBOA

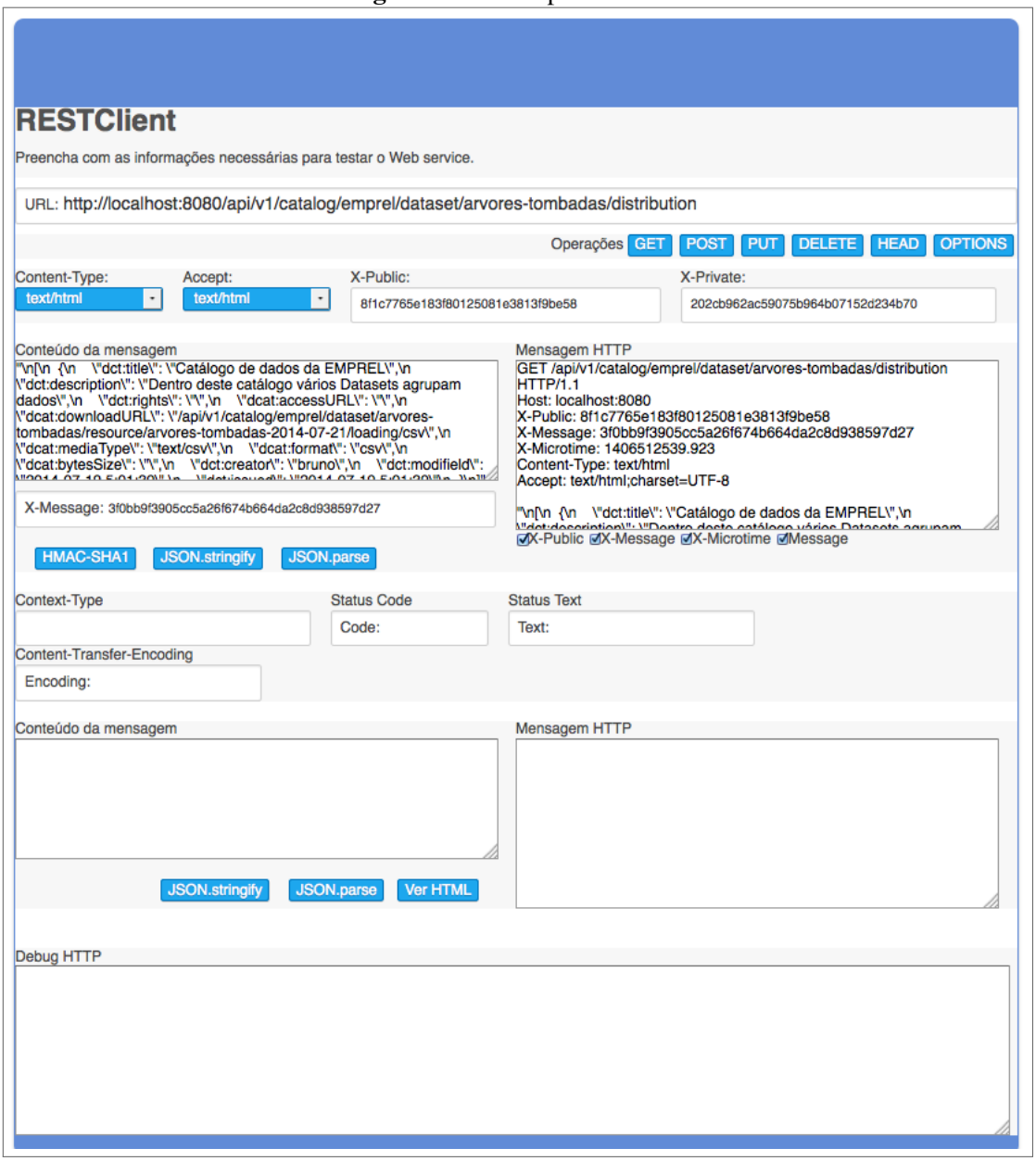
Neste capítulo serão descritos a configuração do ambiente utilizado para execução do experimento, a descrição dos protótipos: abordagem e cliente, a estratégia adotada para execução do experimento e por fim, serão apresentadas as conclusões.

5.1 Descrição do Ambiente

O ambiente controlado foi preparado fazendo uso de 1 (uma) máquina com sistema operacional *OS X* 10.9.4 (13E28), *kernel Darwin* 13.3.0, com processador 2.8 *GHz Intel Core 2 Duo* e memória de 8 GB e para criação do banco de dados, foi utilizado o Sistema Gerenciador de Banco de Dados Objeto Relacional (SGBDOR) *PostgreSQL* versão 9.3.

Foi utilizado o navegador (*browsers*) *Chrome* (36.0.1985.125) para negociar as requisições e respostas. O protótipo do cliente (apresentado na Figura 16) executa no navegador, sendo ele capaz de facilitar a exploração do protótipo da abordagem que responde às requisições. O cliente foi elaborado em HTML com *JavaScript* e *Cascading Style Sheets* (CSS).

Figura 16 Protótipo do cliente



Fonte: o autor

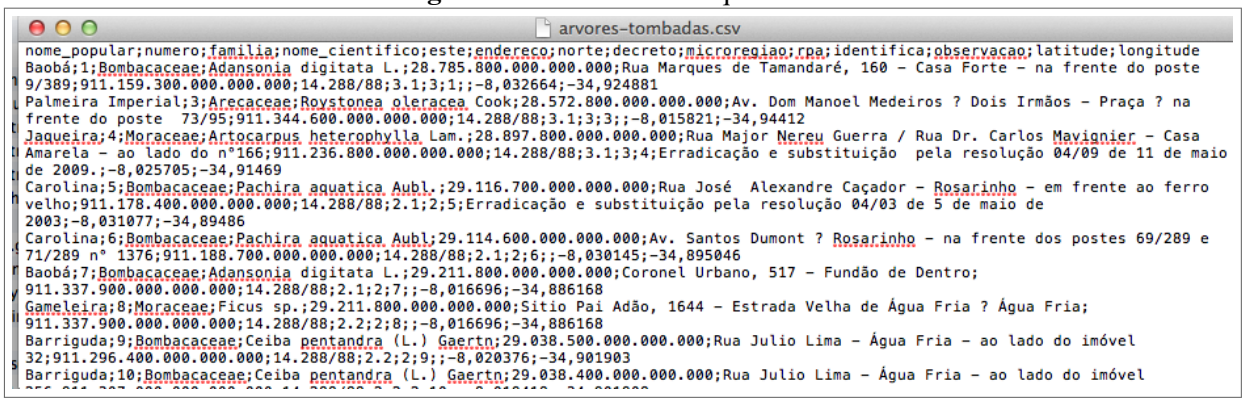
O servidor *Web* utilizado pelo protótipo que implementa a abordagem foi o *Apache*/2.4.6 (*Unix*) e executado com *Hypertext Preprocessor* (PHP)/5.4.22. Após uso do protótipo do cliente (Figura 16), foram necessárias pequenas intervenções na implementação do protótipo da abordagem. Estas alterações forneceram um melhor resultado. Na próxima seção, será apresentado o cenário para execução do experimento.

5.2 Descrição do Cenário

Será apresentado um cenário publicação e uso de dados abertos. Diante disso, um problema será descrito e uma estratégia será traçada para resolução. Primeiramente será importado um conjunto de dados para uma base de dados do tipo PostgreSQL, como origem dos dados é adotado o portal de

dados abertos da prefeitura do Recife[1].

Figura 17 Conteúdo do arquivo CSV



Fonte: o autor

O problema consiste em extrair os dados relacionados a árvores tomadas e publicar como dados abertos. Para o experimento é considerado que os dados já estejam em banco de dados PostgreSQL, para isso foi realizada uma cópia do arquivo em representação CSV[2] que possui o conjunto de árvores tombadas do Recife. Os dados desse arquivo podem ser vistos na Figura 17.

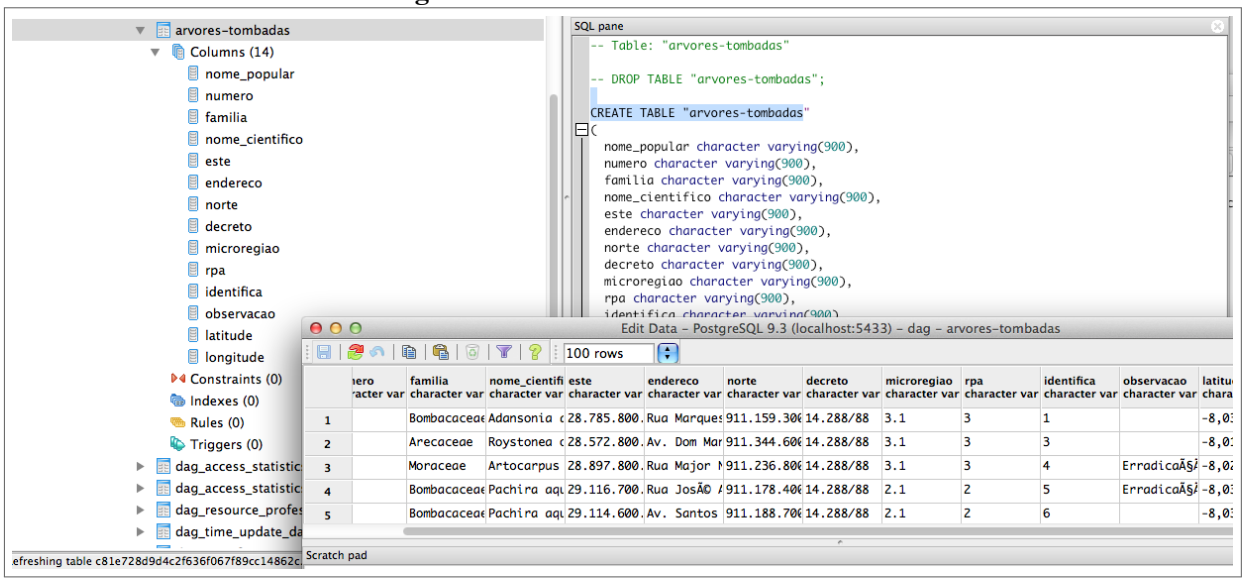
Em seguida os dados do arquivo copiados foram armazenados em banco de dados PostgreSQL. A estratégia adotada para criar o conteúdo dos dados do arquivo na representação em tabela do banco de dados, foi criar um campo na tabela para cada coluna do arquivo.

O próximo passo para a construção do conjunto de dados utilizado no cenário é criar uma tabela (banco de dados) em SGBD *PostgreSQL*. Será criada uma base de dado de nome "DAG", e, dentro dessa base, a tabela "*arvores-tombadas*". As colunas da tabela correspondem às colunas do arquivo CSV copiado anteriormente, podendo ser visto na Figura 18. Na Figura 18, é observa-se a estrutura das instruções DML para criar a tabela. Por fim, os dados foram inseridos na tabela criada.

[1]<http://dados.recife.pe.gov.br/>. Acesso 30/11/2014

[2]<http://dados.recife.pe.gov.br/storage/f/2013-07-14T13%3A18%3A53.433Z/arvores-tombadas.csv>. Acesso: 01/10/2014

Figura 18 Conteúdo do CSV em tabela



Fonte: o autor

As informações necessárias para obter o conjunto de dados "Árvores Tombadas da cidade do Recife" encontram-se na fonte de dados do tipo *PostgreSQL*. Os dados necessários sobre o conjunto de dados podem ser obtidos por meio de consulta em apenas uma tabela. Na próxima seção, será traçado o roteiro para execução do experimento.

5.3 Experimento

De posse da base de dados, foi traçado uma estratégia em sete passos, sendo que a demonstração do primeiro passo será realizada nesta seção, as demais encontram-se no apêndice C. Tais passos são:

1. Obter os dados necessários para criar o catálogo de dados e criar o catálogo;
2. Obter os dados necessários para criar o *dataset* e criar *dataset*;
3. Obter os dados necessários para conectar-se à fonte de dados e criar conexão;
4. Obter os dados necessários para realizar a extração dos dados e fazer a extração dos dados;
5. Criar representação dos dados em notação CSV;
6. Criar distribuição contendo a representação gerada;
7. Obter distribuição de dados por meio do recurso "*data*".

Vale ressaltar que durante a execução do experimento foram omitidos os parâmetros "*Date*", "*Server*", "*X-Powered-By*", "*User-Agent*", "*Keep-Alive*" e "*Connection*" contidas nas respostas do protótipo da abordagem.

Na criação do protótipo do cliente, foram utilizadas bibliotecas como "*crypto-js*"[3], "*base64*"[4] e "*XMLHttpRequest*"[5]. A *cripto-js* fornece o algorítimo utilizado para gerar as chaves baseadas em HMAC-SHA1. A *base64* fornece criptografia para empacotar as informações armazenadas em *cookies* no navegador. Por fim, a última fornece suporte ao recurso conhecido como *Asynchronous JavaScript and XML* (AJAX).

Foi utilizado o "*json-generator*"[6] para validar e formatar os documentos em notação JSON. Ele permite uma verificação em tempo real da estrutura da notação. Graças a esta validação, as chances de enviar uma requisição ruim são minimizadas.

No conteúdo das mensagens em notação JSON é possível observar também a presença de caracteres "estranhos" que dificultam o entendimento. Estes caracteres estranhos fazem parte da codificação utilizada e especificada na RFC 4627[7] "*The application/json*" JSON para serializar os dados. As bibliotecas que implementam o protocolo JSON fornecem forma para "corrigir" os dados para uma apresentação atraente ao usuário. Foi utilizada a ferramenta "JSON *parser online*" para organizar os dados, tornando os dados codificados em textos legíveis e apresentáveis de forma hierárquica. A próxima seção dá início à execução das simulações.

As informações necessárias para criar o catálogo são apresentadas na Figura 19. Primeiramente, são obtidos os dados e organizados na Tabela 22. Após aquisição dos dados, eles são organizados na estrutura de notação JSON.

Figura 19 Informações sobre o portal de dados abertos

Fonte: o autor

Para preenchimento da propriedade "*dct:language*", foi utilizado o vocabulário controlado (FREED; BAKER; HOEHRMANN, 2014). Outras informações sobre este campo podem ser encontradas em *W3C Language tags in HTML and XML*[8]. Ao usar este vocabulário controlado, os dados contribuem com futuras integrações de dados. As propriedades do recurso catálogo e valores podem ser vistos na Tabela 22.

[3]<http://crypto-js.googlecode.com/>. Acesso: 01/10/2014
[4]<http://www.webtoolkit.info/>. Acesso: 01/10/2014
[5]<http://www.w3.org/TR/XMLHttpRequest/>. Acesso: 01/10/2014
[6]<http://www.json-generator.com>. Acesso: 01/10/2014
[7]<http://tools.ietf.org/html/rfc4627>. Acesso: 01/10/2014
[8]<http://www.w3.org/International/articles/language-tags/Overview.en.php>. Acesso: 01/10/2014

Tabela 22 Propriedades do Catálogo

Propriedade	Valor
dct:title	Catálogo de dados da *EMPREL*
dct:description	Este catálogo de dados agrupa vários conjuntos que oferecem dados sobre a cidade do Recife.
dct:identifier	*emprel*
dct:language	BR
dct:license	*<http://www.opendatacommons.org/licenses/pddl/1.0/>*
dct:rights	Domínio Público
dct:spatial	Recife
foaf:homepage	*http://dados.recife.pe.gov.br/*

Fonte: o autor

Figura 20 Estrutura em notação JSON presente na tabela 22

```
{
  "dct:title": "Catalogo de dados da EMPREL",
  "dct:description": "Dentro deste catalogo varios Datasets agrupam dados",
  "dct:language": "BR",
  "dct:license": "http://www.opendatacommons.org/licenses/pddl/1.0/",
  "dct:rights": "Dominio Publico",
  "dct:spatial": "",
  "foaf:homepage": "http://dados.recife.pe.gov.br",
  "dct:identifier": "emprel"
}
```

Fonte: o autor

A Figura 20 apresenta o conteúdo da Tabela 22 utilizando notação JSON para representar os dados. Em especial ao recurso "*catalog*", todos os métodos de suporte no protótipo da abordagem serão explorados. Primeiramente, será acessado o método *OPTIONS* para obter a lista dos métodos de suporte e, em seguida, o método *HEAD* para obter metadados do recurso.

Atividade: obter métodos do recurso "*catalog*"

A Figura 21 apresenta a mensagem enviada ao protótipo da abordagem. Esta mensagem é enviada usando uma requisição por meio do protocolo HTTP/1.1 e método *OPTIONS*. A propriedade "*Accept*" indica o formato da representação aceita como resposta ao pedido enviado.

Figura 21 Requisição enviada ao recurso "*catalog*" usando método *OPTIONS*

```
OPTIONS /api/v1/catalog HTTP/1.1
Host: localhost:8080
Accept: application/json;charset=UTF-8
```

Fonte: o autor

A Figura 22 apresenta a mensagem retornada pelo protótipo da abordagem. Esta mensagem retornada contém as propriedades "*Allow*" que fornece uma lista com os métodos do HTTP de suporte do recurso "*/api/v1/catalog*". As propriedades "*Content-Length*" e "*Content-Type*", respectivamente, correspondem ao tamanho do corpo da mensagem e o formato da representação contida no corpo da mensagem respondida. No corpo da mensagem também consta uma lista contendo os métodos

e restrições. O parâmetro "*access*" igual a "*private*" informa ao cliente que aquele método exige autenticação. Já a restrição "*public*" não exige credencial. O código de estado "200" representa que esta solicitação foi bem sucedida.

Figura 22 Reposta ao pedido do método *OPTIONS*

```
HTTP/1.1 200 OK
Allow: POST, GET, PUT, DELETE, HEAD, OPTIONS
Content-Length: 271
Content-Type: application/json; charset=UTF-8
{
    "method": "POST",
    "access": "private"
  },
  {
    "method": "GET",
    "access": "public"
  },
  {
    "method": "PUT",
    "access": "private"
  },
  {
    "method": "DELETE",
    "access": "private"
  },
  {
    "method": "HEAD",
    "access": "public"
  },
  {
    "method": "OPTIONS",
    "access": "public"
  }
```

Fonte: o autor

Atividade: obter *metadados* sobre o recurso "*catalog*"

Para obter informações (de *metadados*) sobre o recurso "*catalog*", será utilizado o método *HEAD*. Ele retorna dados que contribuem com a descrição do recurso. A Figura 23 apresenta a mensagem enviada ao protótipo da abordagem, o qual é a implementação da abordagem proposta. Esta mensagem envia uma requisição usando o protocolo HTTP/1.1, por meio do método *HEAD*.

Figura 23 Requisição enviada ao recurso "*catalog*" usando método *HEAD*

```
HEAD /api/v1/catalog HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Accept: application/json;charset=UTF-8
```

Fonte: o autor

A Figura 24 apresenta a mensagem retornada pelo protótipo da abordagem. É característica deste método não retornar corpo da mensagem, e toda informação deve ser trafegada por meio de parâmetros embutidos no cabeçalho "*hearders*" da mensagem HTTP.

Figura 24 Resposta ao pedido do método *HEAD*

```
HTTP/1.1 200 OK
Content-Description: Recurso pode ser utilizado para editar e visualizar catalogos de ↩
    dados.
Content-Language: PT
Content-author: Bruno Maciel<brunom4ciel@gmail.com>
Cache-Control: public
Content-Type: text/html; charset=UTF-8
```

Fonte: o autor

Atividade: obter novo estado para o recurso "*catalog*"

Obtidas as informações necessárias (Figura 20) para utilizar o método *POST*, será então enviada requisição para criar um novo catálogo de dados. Para criar uma instância do recurso (ou novo estado do recurso) "*catalog*", será utilizado o método *POST*. A Figura 25 apresenta a mensagem enviada ao protótipo da abordagem (deste ponto em diante será usado o termo "servidor" para representar o protótipo da abordagem) para a criação do catálogo de dados "*emprel*". Esta mensagem é enviada em uma requisição, usando o protocolo HTTP/1.1, por meio do método *POST*.

Figura 25 Requisição enviada ao recurso "*catalog*" usando método *POST*

```
POST /api/v1/catalog HTTP/1.1
Host: localhost:8080
X-Public: 8f1c7765e183f80125081e3813f9be58
X-Message: ab30dbf3ec7f0aa34f1dfd3f2af247c8935175e4
X-Microtime: 1406453845.663
Content-Type: application/json
Accept: application/json;charset=UTF-8

{
  "dct:title": "Catalogo de dados da EMPREL",
  "dct:description": "Dentro deste catalogo varios Datasets agrupam dados",
  "dct:language": "BR",
  "dct:license": "http://www.opendatacommons.org/licenses/pddl/1.0/",
  "dct:rights": "Dominio Publico",
  "dct:spatial": "",
  "foaf:homepage": "http://dados.recife.pe.gov.br"
  "dct:identifier": "emprel"
}
```

Fonte: o autor

A Figura 26 apresenta a mensagem retornada pelo protótipo da abordagem. Nesta implementação, o método *POST* apenas retorna o código de estado HTTP e propriedade "*Location*", sendo ela responsável por fazer o *hyperlink* para o novo estado do recurso criado.

Figura 26 Resposta ao pedido do método *POST*

```
HTTP/1.1 201 Created
Location: /api/v1/catalog/emprel
Content-Length: 0
Content-Type: text/html
```

Fonte: o autor

Atividade: obter estado criado do recurso "*catalog*"

Para obter a instância "*emprel*" do recurso "*catalog*", será utilizado o método *GET*, que

retorna o conteúdo da instância do recurso em uma representação específica. A Figura 27 apresenta a mensagem enviada ao protótipo da abordagem. Esta mensagem envia uma requisição, usando o HTTP/1.1 e método *GET* solicitando o catálogo em representação de dados JSON, conforme descrito na propriedade "*Accept*".

Figura 27 Requisição enviada ao catálogo "*emprel*" do recurso "*catalog*" usando o método *GET*

```
GET /api/v1/catalog/emprel HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Accept: application/json;charset=UTF-8
```

Fonte: o autor

A Figura 28 apresenta a mensagem retornada pelo protótipo da abordagem.

Figura 28 Resposta contendo o catálogo solicitado

```
HTTP/1.1 200 OK
Content-Length: 471
Content-Type: application/json; charset=UTF-8

{
  "dct:title": "Catalogo de dados da EMPREL",
  "dct:description": "Dentro deste catalogo varios Datasets agrupam dados.",
  "dct:language": "BR",
  "dct:license": "http://www.opendatacommons.org/licenses/pddl/1.0/",
  "dct:rights": "Dominio Publico",
  "dct:spatial": "",
  "foaf:homepage": "http://dados.recife.pe.gov.br"
  "dct:identifier": "emprel"
}
```

Fonte: o autor

Atividade: alterar estado de recurso um "*catalog*"

Para exemplificar uma atualização nos dados de uma instância, será utilizado o método *PUT*. Para atualizar a instância "*emprel*" do recurso "*catalog*", será utilizada a mensagem HTTP da Figura 29, onde apenas a propriedade "*dct:description*" será alterado. A Figura 29 apresenta a mensagem enviada ao protótipo da abordagem. Esta mensagem envia uma requisição, usando o protocolo HTTP/1.1 e método *PUT*.

Figura 29 Requisição enviada à instância "*emprel*" do recurso "*catalog*" usando o método *PUT*

```
PUT /api/v1/catalog/emprel HTTP/1.1
Host: localhost:8080
X-Public: 8f1c7765e183f80125081e3813f9be58
X-Message: bfbefe7e14d46c9278d605095221f6dd1bb5b783
X-Microtime: 1406454377.5
Content-Type: application/json
Accept: application/json;charset=UTF-8

{
  "dct:title": "Catalogo de dados da EMPREL",
  "dct:description": "Dentro deste catalogo varios Datasets agrupam dados.",
  "dct:language": "BR",
  "dct:license": "http://www.opendatacommons.org/licenses/pddl/1.0/",
  "dct:rights": "Dominio Publico",
  "dct:spatial": "",
  "foaf:homepage": "http://dados.recife.pe.gov.br"
  "dct:identifier": "emprel"
}
```

Fonte: o autor

A Figura 30 apresenta a mensagem retornada pelo protótipo da abordagem. Este método *PUT* apenas retorna o código de estado HTTP, informando o processo da operação.

Figura 30 Resposta ao pedido do método *PUT* para atualizar a instância "*emprel*" do recurso "*catalog*"

```
HTTP/1.1 202 Accepted
Content-Length: 0
Content-Type: application/json; charset=UTF-8
```

Fonte: o autor

Para remover uma instância existente, será utilizado o método *DELETE*. Para a instância "*emprel*" ser removida do recurso "*catalog*", será utilizada a mensagem HTTP da Figura 31. Esta mensagem envia uma requisição, usando o HTTP/1.1 e método *DELETE*.

Figura 31 Requisição enviada à instância "*emprel*" do recurso "*catalog*" usando o método *DELETE*

```
DELETE /api/v1/catalog/EMPREL HTTP/1.1
Host: localhost:8080
X-Public: 8f1c7765e183f80125081e3813f9be58
X-Message: 30d12ac6c43d219ac9dc431e1d57015513e08f1b
X-Microtime: 1406088438.956
Content-Type: application/json
Accept: application/json;charset=UTF-8
```

Fonte: o autor

A Figura 32 apresenta a mensagem retornada pelo protótipo. O método *DELETE*, quando executado com sucesso, apenas retorna o código de estado HTTP, informando o processo da operação.

Figura 32 Resposta ao pedido do método *DELETE* para remover um estado do recurso

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: application/json; charset=UTF-8
```

Fonte: o autor

Os demais passo do experimento foram incluídos no apêndice C, com o intuito de tornar a leitura deste capítulo menos cansativa. A próxima seção apresenta as conclusões desta capítulo.

5.4 Conclusões

Este capítulo apresentou a validação da abordagem de publicação e uso de dados abertos, por meio de experimentos. Inicialmente, foi descrito o protótipo utilizado para avaliar a implementação da abordagem proposta, o qual oferece funcionalidades para que o usuário manipule as requisições, respostas, parâmetros e o próprio corpo da mensagem. Em seguida, foi descrito o cenário no qual os experimentos foram executados.

O experimento realizado teve como objetivo ilustrar as requisição que a abordagem proposta é capaz de processar. Para o cenário apresentado, foram enviadas várias requisições e obtidas respostas. Ao final da execução do plano de ação, foi obtido o conjunto de dados “arvores tombadas”.

Portanto, apesar da existência de limitações, estas foram consideradas, a fim de minimizar seu impacto sobre o resultado final da avaliação. Na seção seguinte, serão apresentadas as considerações finais e recomendações de trabalhos futuros.

6

Considerações Finais

Não se lembrarão de você, se lembrarão da sua reputação.

—SYLVESTER STALLONE - ROCKY BALBOA

Este trabalho apresentou uma abordagem baseada em serviços para publicação e uso de dados abertos. Este capítulo tem por objetivo apresentar as considerações finais sobre este trabalho, enfatizando suas principais contribuições e resultados, além da sugestão de trabalhos futuros.

Foi apresentada uma visão geral da fundamentação teórica desta dissertação, discutindo os principais assuntos relacionados à esta abordagem. Para isso, foi dada uma visão geral sobre Dados Abertos, *Web Service* e processo de ETL. Além disso, foi apresentada, de maneira mais detalhada, sobre os trabalhos que possuem características mais próximas ao foco desta dissertação e, por fim, realizou-se uma análise comparativa entre eles.

Um dos pontos altos desta dissertação, está na definição e especificação de uma arquitetura que adota os princípios do REST para sua concepção. Com esta arquitetura, é possível realizar:

– Geração automática de múltiplas representações dos dados;
– Geração automática de múltiplas representações dos Metadados;
– Permite catalogar os dados publicados na Web utilizando o DCAT;
– Suporta URI Amigável;
– Integração de tarefas (manipulação e publicação);
– Permite a reutilização de Metadados.
– Automatização do trabalho.

A principal contribuição deste trabalho é a abordagem de publicação e uso de dados abertos baseada em serviços *Web*. Ademais, ela valoriza o conhecimento do usuário sobre a aplicação, permitindo que ele escolha quais recursos necessitam ou não utilizar. Desse modo, é possível explorar os diferentes recursos, contribuindo com a proposta de criar uma cadeia de valor pra os dados abertos e permitindo a utilização por diferentes tipos de aplicações, quanto à necessidade do utilizador.

Outra contribuição deste trabalho, dá-se pelo desenvolvimento dos protótipos, cliente e servidor. O primeiro contribui na avaliação da abordagem proposta por meio da execução do experimento, facilitando o tratamento das mensagens do protocolo HTTP. O experimento apresentadas no

capítulo anterior, permitiu uma avaliação da proposta e ofereceu maiores detalhes técnicos sobre a utilização.

Por fim, conclui-se que o objetivo estabelecido no início do desenvolvimento do trabalho - **Como fornecer uma abordagem capaz de contribuir com as atividades de publicação e uso de dados abertos governamentais** - foi cumprido. Nas seções seguintes, apresentam-se o escopo negativo do trabalho bem como as recomendações de trabalhos futuros.

6.1 Escopo Negativo

A abordagem proposta faz parte de um contexto mais amplo, no qual um conjunto de aspectos relacionados não foram tratados. Mesmo que a cobertura fornecida em sua versão atual implementada sejam baseada em aspectos bem fundamentados de qualidade e desempenho, futuras melhorias para atender de forma mais eficiente o seu propósito não são descartadas. Enquanto isso, os aspectos não diretamente abordados por este trabalho são listados a seguir:

- eficiência na utilização dos recursos da infraestrutura física e lógica;
- confiabilidade no sentido de tolerância a falhas e recuperabilidade;
- manutenibilidade da solução relacionada à escalabilidade

6.2 Trabalhos Futuros

Como proposta de trabalhos futuros, recomenda-se, principalmente, o desenvolvimento de APIs do lado do desenvolvedor. Estas APIs podem contribuir de forma a facilitar o uso, reduzir códigos, reutilizar funcionalidades, minimizar o tempo de desenvolvimento, tratar exceções, dentre outras atribuições. Ademais, acredita-se que há muito que pode ser feito no contexto do trabalho, existem muitas lacunas a serem preenchidas, tanto relacionadas a infraestrutura como uso dos dados abertos. A seguir, alguns pontos em aberto que podem ser tomados como uma possível continuação do trabalho:

- Explorar a integração com plataformas de dados abertos, por exemplo CKAN.
- Explorar o contexto da recuperação da informação.
- *Triplificar* os dados usando o repositório central de dados.
- Explorar o enriquecimento semântico dos dados nos novos conjuntos de dados de maneira automaticamente.
- Explorar o desenvolvimento de um ambiente integrado que seja possível criar aplicações utilizando os recursos da abordagem.

Por fim, recomenda-se o desenvolvimento de aplicações que utilizem a abordagem proposta (integrar e automatizar o processo publicação de dados abertos). Aplicações com naturezas variadas podem se beneficiar dos recursos para acelerar o tempo de desenvolvimento e manutenção dos dados.

Referências

AGHAEE, S.; PAUTASSO, C. The mashup component description language. In: **Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services**. New York, NY, USA: ACM, 2011. (iiWAS '11), p. 311–316. ISBN 978-1-4503-0784-0. Disponível em: <http://doi.acm.org/10.1145/2095536.2095591>.

ARAUJO, M. H. de *et al.* Dados governamentais abertos: Uma análise sob a ótica das dimensões de qualidade da informação. **Anais do EnANPAD 2012**, p. 1–16, 2012.

BERNERS-LEE, T. **Linked Data**. 2006. Disponível em: <http://www.w3.org/DesignIssues/LinkedData.html>. Acesso em: 21 ago. 2014.

CASTERS, M.; BOUMAN, R.; DONGEN, J. v. **Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration**. [S.l.]: Wiley Publishing, 2010. ISBN 0470635177, 9780470635179.

DAIGNEAU, R.; ROBINSON, I. **Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services**. 1nd. ed. [S.l.]: Addison-Wesley Professional, 2011. ISBN 032154420X, 978-0321544209.

DINIZ, V. **Como conseguir dados governamentais abertos**. 2009. Congresso Consad de Gestão Pública. 3., Anais... Disponível em: <http://www.consad.org.br/sites/1500/1504/00001870.pdf>. Acesso em: 21 ago. 2014.

ERL, T. **Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services**. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004. ISBN 0131428985.

FERNANDEZ, F. M. **ANALYZING THE REST ARCHITECTURAL STYLE WITH EXTENDED INFLUENCE DIAGRAMS**. Dissertação (Mestrado) — PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE, ESCUELA DE INGENIERIA, Santiago de Chile, Julho 2009. Disponível em: <http://repositorio.uc.cl/xmlui/bitstream/handle/123456789/1353/525569.pdf?sequence=1>.

FIELDING, R. T. **REST: Architectural Styles and the Design of Network-based Software Architectures**. Tese (Doctoral dissertation) — University of California, Irvine, 2000. Disponível em: <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>.

FIELDING, R. T. *et al.* **RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1**. [S.l.], 1999. <http://www.ietf.org/rfc/rfc2616.txt>.

FREED, p. N.; BAKER, s. M.; HOEHRMANN, s. B. **Media Types**. 2014. Disponível em: <http://www.iana.org/assignments/media-types/media-types.xhtml>. Acesso em: 21 ago. 2014.

GARLAN, D. *et al.* **Documenting Software Architectures: Views and Beyond**. 2nd. ed. [S.l.]: Addison-Wesley Professional, 2010. ISBN 0321552687, 9780321552686.

INMON, W. H. **Building the Data Warehouse (2Nd Ed.)**. New York, NY, USA: John Wiley & Sons, Inc., 1996. ISBN 0-471-14161-5.

ISO/IEC, I. **Systems and software engineering - Architecture description**. 2011. Disponível em: <http://cabibbo.dia.uniroma3.it/asw/altrui/iso-iec-ieee-42010-2011.pdf>. Acesso em: 21 ago. 2014.

JONES, M. T. **Understand Representational State Transfer (REST) in Ruby**. 2012. Disponível em: <http://www.ibm.com/developerworks/library/os-understand-rest-ruby/>. Acesso em: 21 ago. 2014.

KIMBALL, R.; CASERTA, J. **The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data**. [S.l.]: John Wiley & Sons, 2004. ISBN 0764567578.

KRUCHTEN, P. **The Rational Unified Process: An Introduction**. 3nd. ed. Boston, MA, USA: Addison-Wesley Professional, 2003. 336 p. ISBN 0201707101.

KRUCHTEN, P. R. S. C. Architectural blueprints - the "4+1"view model of software architecture. **IEEE Softw.**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 12, n. 6, p. 42–50, nov. 1995. ISSN 0740-7459. Disponível em: <http://dx.doi.org/10.1109/52.469759>.

LIU, L.; ZSU, M. T. **Encyclopedia of Database Systems**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 0387355448, 9780387355443.

MAALI, F.; ERICKSON, J. **Data Catalog Vocabulary (DCAT)**. 2014. <http://www.w3.org/TR/vocab-dcat/>. Disponível em: <http://www.w3.org/TR/vocab-dcat/>. Acesso em: 21 ago. 2014.

MANUAL, O. D. **Manual dos Dados Abertos: Governo**. 2011. Disponível em: <http://www.w3c.br/pub/Materiais/PublicacoesW3C/Manual_Dados_Abertos_WEB.pdf>. Acesso em: 21 ago. 2014.

MEDJAHED, B.; BOUGUETTAYA, A. **Service Composition for the Semantic Web**. [s.n.], 2011. Disponível em: <http://dx.doi.org/10.1007/978-1-4419-8465-4>.

MPINDA, S. A. T.; BUNGAMA, P. A.; MASCHIETTO, L. G. Graph database application using neo4j (railroad planner simulation). In: ESRSA PUBLICATIONS. **International Journal of Engineering Research and Technology**. [S.l.], 2015. v. 4, n. 04 (April-2015).

NOTTINGHAM, M. **Web Linking**. 2010. Disponível em: <http://tools.ietf.org/html/rfc5988>. Acesso em: 21 ago. 2014.

OKFN.BR. **Cinco Estrelas**. 2013. Disponível em: <http://br.okfn.org/2013/01/17/maturidade-em-dados-abertos-entenda-as-5-estrelas>. Acesso em: 21 ago. 2014.

OLIVEIRA, R. R. d. **Avaliação de manutenibilidade entre as abordagens de web services RESTful e SOAP-WSDL**. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação - USP, Maio 2012. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-24072012-164751/pt-br.php>.

PAUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F. Restful web services vs. "big"' web services: Making the right architectural decision. In: **Proceedings of the 17th International Conference on World Wide Web**. New York, NY, USA: ACM, 2008. (WWW '08), p. 805–814. ISBN 978-1-60558-085-2. Disponível em: <http://doi.acm.org/10.1145/1367497.1367606>.

PIPINO, L. L.; LEE, Y. W.; WANG, R. Y. Data quality assessment. **Commun. ACM**, ACM, New York, NY, USA, v. 45, n. 4, p. 211–218, abr. 2002. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/505248.506010>.

PORTAL BRASILEIRO DE DADOS ABERTOS. **Página oficial do portal brasileiro de dados abertos**. 2014. Disponível em: <http://dados.gov.br/dados-abertos/>. Acesso em: 21 ago. 2014.

RICHARDSON, L.; RUBY, S. **Restful Web Services**. First. [S.l.]: O'Reilly, 2007. ISBN 9780596529260.

SILVA, D. B. **TRANSPARÊNCIA NA ESFERA PÚBLICA INTERCONECTADA**. Dissertação (Mestrado) — FACULDADE CÁSPER LÍBERO, SÃO PAULO, Fevereiro 2010.

SUMARAY, A.; MAKKI, S. K. A comparison of data serialization formats for optimal efficiency on a mobile platform. In: **Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication**. New York, NY, USA: ACM, 2012. (ICUIMC '12), p. 48:1–48:6. ISBN 978-1-4503-1172-4. Disponível em: <http://doi.acm.org/10.1145/2184751.2184810>.

The Open Definition. **Página oficial para definição de abertura.** 2014. Disponível em: <http://opendefinition.org>. Acesso em: 21 ago. 2014.

Tim Davies. **Supporting open data use through active engagement**. 2012. Disponível em: <http://www.w3.org/2012/06/pmod/pmod2012_submission_5.pdf>. Acesso em: 21 ago. 2014.

VASSILIADIS, P. A survey of extract-transform-load technology. **IJDWM**, v. 5, n. 3, p. 1–27, 2009.

W3C, W. G. **Web Services Glossary**. 2014. Disponível em: <http://www.w3.org/TR/ws-gloss/>. Acesso em: 21 ago. 2014.

Apêndice

A

Descrição da Mensagem do HTTP Implementado na Abordagem

Este apêndice A, apresenta a descrição detalhada da implementação da mensagem do HTTP que foi utilizada na abordagem desse trabalho. A seguir, serão apresentados alguns dos detalhes técnicos utilizados sobre o HTTP.

Cabeçalhos de Requisições

Nas mensagens do HTTP enviadas a implementação da abordagem, algumas propriedades de cabeçalhos dependendo do método, devido a privacidade dos dados. As operações que exigem autenticação do cliente exigem as propriedades "*X-Public*", "*X-Microtime*" e "*X-Message*".

Tabela 23 Parâmetros de cabeçalhos HTTP exigidos e recomendados

Header	Detalhe
X-Public	Chave pública que identifica o cliente no servidor.
X-Message	Chave da Mensagem, gerada com algorítimo HMAC e criptografia SHA1.
X-Microtime	*Timestamp* da máquina que solicita a requisição ao *Web service*.
Content-Type	Tipo da mídia "*Multipurpose Internet Mail Extensions* (MIME)" que a mensagem esta sendo enviada ao servidor.
Accept	Tipo da mídia "MIME" que o cliente aceita receber a mensagem de resposta.

Fonte: o autor

Na Tabela 24, são apresentas as propriedades presentas nas respostas das requisições ao protótipo da abordagem.

Tabela 24 Propriedades de cabeçalhos HTTP exigidos e recomendados

Header	Detalhe
Content-Length	Especifica o tamanho em *bytes* da mensagem.
Content-Type	Tipo da mídia "MIME" que a mensagem esta sendo enviada ao servidor.
Content-Description	*Metadados* retornados em consultado ao método HTTP *HEAD*
Cache-Control	Informa ao cliente que ele pode armazenar o conteúdo processado do lado cliente. Exemplo, valor=*public*.
Content-author	Informa o autor do recurso.
Content-keywords	Informa as palavras chaves.
application-name	Nome do *Web service*.
X-Powered-By	*Restful Web service/beta*.
Allow	Informa os método de suporte do recurso.
Content-Language	Idioma do recurso.

Fonte: o autor

Tratamento de Erro

Para este trabalho foram adotados alguns dos códigos de estados do HTTP. A relação completa dos códigos de estado pode vista em Fielding *et al.* (1999, p. 40–41). No Tabela 25 pode ser vista a relação de códigos adotados neste trabalho e seu respectivo significado.

Tabela 25 Relação de códigos do HTTP utilizados na especificação

Código	Definição	Detalhe
200	*Ok*	Solicitação foi concluída com sucesso.
201	*Created*	Criado com sucesso.
202	*Accepted*	Aceito.
204	*No Content*	Nenhum conteúdo.
302	*Found*	Redirecionamento.
400	*Bad Request*	Sua solicitação está mal formada e não pode ser entendida.
401	*Unauthorized*	Não autorizado.
404	*Not Found*	Recurso não existe.
409	*Conflict*	Existe recurso nesta URI.
415	*Unsupported Media Type*	Não suporta o tipo de mídia solicitada.
423	*Locked*	Recurso bloqueado.
500	*Internal Server Error*	Houve um problema interno.

Fonte: o autor

A relação completa dos códigos de estados do protocolo HTTP pode ser encontrada na RFC 2616 especificada por Fielding *et al.* (1999, p. 40–41) ou por meio do IANA[1].

[1] http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

B

Descrição da Implementação dos Recursos da Abordagem

São apresentados detalhes da implementação dos recursos que foram utilizados na abordagem apresentada neste trabalho. A seguir, para cada recurso serão apresentadas as URI e possíveis propriedades.

Recurso *resource*

O recurso *resource* é utilizado para construir o conjuntos de dados. A partir da criação de uma novo estado para o recurso *resource* é possível realizar as operações de extração, transformação e enriquecimento semântico dos dados com metadados. A Tabela 26, fornece as URIs e métodos suportados do HTTP.

Tabela 26 URIs do recurso *resource* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource*	x	x		
2	*/catalog/:id/dataset/:id/resource/:id*	x	x	x	x

Fonte: o autor

A Tabela 27, apresenta as propriedades, descrições e restrições quanto ao uso da propriedade, especificamente existem dois possíveis valores, são eles: automático e opcional. O primeiro, corresponde se a propriedade possui valor inserido automaticamente por meio da implementação. Seguidamente, faz referência se ela pode ser deixada em branco.

Tabela 27 Propriedades do recurso *resource*

Propriedade	**Restrição**	**Detalhe**
dct:title	obrigatório	Descreve o título do recurso de dados.
dct:description	opcional	Descreve detalhes sobre o recurso.
dct:issued	automático	Data da publicação.
dct:modifield	automático	Última data de alteração.
dct:creator	automático	Descreve a entidade.
dct:identifier	obrigatório	Referência não ambígua ao recurso.
dcat:keyword	opcional	Palavras chaves ou *tags* que descrevem o recurso.
dct:language	opcional	O idioma dos dados.
dct:accrualPeriodicity	obrigatório	A frequência com que o conjunto de dados é publicado.
b:updateAutomatic	opcional	define se o recurso deve atualizar automaticamente os dados.

Fonte: o autor

Os dados deste recurso serão armazenados no lado do servidor em SGBD *PostgreSQL*. Ele fornecer dentre outras informações o título, autor, idioma e uma descrição para o conjunto de dados.

Recurso "*connector*"

O recurso *connector* é utilizado para listar os conectores existentes. A Tabela 28, fornece a URI e método suportado do HTTP.

Tabela 28 URIs do recurso *connector* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/connector/*	x			

Fonte: o autor

Recurso "*postgresql*"

O recurso *postgresql* é utilizado para configurar a conexão e consulta a serem utilizadas na fonte dados de origem. Em especial, este recurso *postgresql*, oferece suporte apenas a tecnologia *PostgreSQL*. A Tabela 28, fornece a URI e método suportado do HTTP.

Tabela 29 URIs do recurso *postgresql* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/connector/postgresql*		x	x	

Fonte: o autor

A Tabela 29, apresenta as propriedades, descrições e restrições quanto ao uso da propriedade.

Tabela 30 Propriedades recurso *postgresql*

Propriedade	**Restrição**	**Detalhe**
dct:identifier	automático	Identificador único e sequencial.
dct:issued	automático	Data de publicação.
dct:modifield	automático	Última data de alteração.
dct:creator	automático	Descreve entidade principal responsável.
b:db_user	obrigatório	Especifica o nome do usuário do SGBD.
b:db_password	obrigatório	Especifica a senha do usuário do SGBD.
b:db_host	obrigatório	Especifica o host de acesso ao SGBD.
b:db_name	obrigatório	Especifica o nome da base de dados no SGBD.
b:db_port	obrigatório	Especifica a porta de acesso no SGBD.
b:db_query	obrigatório	Especifica a consulta a ser realizada no SGBD.

Fonte: o autor

Recurso "*mysql*"

O recurso *mysql* é utilizado para configurar a conexão e consulta a serem utilizadas na fonte dados de origem. Em especial, este recurso *mysql*, oferece suporte apenas a tecnologia *MySQL*. A Tabela 31, fornece a URI e método suportado do HTTP.

Tabela 31 URIs do recurso *mysql* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/connector/mysql/*		x	x	

Fonte: o autor

A Tabela 32, apresenta as propriedades, descrições e restrições quanto ao uso da propriedade.

Tabela 32 Propriedades do recurso *mysql*

Propriedade	**Restrição**	**Detalhe**
dct:identifier	automático	Identificador único e sequencial.
dct:issued	automático	Data de publicação.
dct:modifield	automático	Última data de alteração.
dct:creator	automático	Descreve entidade principal responsável.
b:db_user	obrigatório	Especifica o nome do usuário do SGBD.
b:db_password	obrigatório	Especifica a senha do usuário do SGBD.
b:db_host	obrigatório	Especifica o host de acesso ao SGBD.
b:db_name	obrigatório	Especifica o nome da base de dados no SGBD.
b:db_port	obrigatório	Especifica a porta de acesso no SGBD.
b:db_query	obrigatório	Especifica a consulta a ser realizada no SGBD.

Fonte: o autor

Recurso "csv"

O recurso *csv* é utilizado para configurar o endereço de fonte dados de origem baseada em arquivo remoto. Em especial, este recurso *csv*, oferece suporte apenas a notação CSV. A Tabela 33, fornece a URI e método suportado do HTTP.

Tabela 33 URIs do recurso *csv* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/connector/csv*		x	x	

Fonte: o autor

A Tabela 34, apresenta as propriedades, descrições e restrições quanto ao uso da propriedade.

Tabela 34 Propriedades do recurso *csv*

Propriedade	**Restrição**	**Detalhe**
dct:identifier	automático	Identificador único e sequencial.
dct:issued	automático	Data de publicação.
dct:modifield	automático	Última data de alteração.
dct:creator	automático	Descreve entidade principal responsável.
b:remote_file	obrigatório	URL do arquivo.

Fonte: o autor

Recurso "*extraction*"

O recurso *extraction* é utilizado para realizar à extração dos dados configurados em uma fonte de dados de origem. Em especial, este recurso *extraction*, oferece suporte aos três tipos de conectores oferecidos na implementação do protótipo da abordagem. A Tabela 35, fornece as URIs e métodos suportados do HTTP.

Tabela 35 URIs do recurso *extraction* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/connector/postgresql/extraction*		x	x	
2	*/catalog/:id/dataset/:id/resource/:id/connector/mysql/extraction*		x	x	
3	*/catalog/:id/dataset/:id/resource/:id/connector/csv/extraction*		x	x	

Fonte: o autor

Recurso "*operation*"

O recurso *operation* é utilizado para listar as operações de transformações existentes. A Tabela 36, fornece a URI e método suportado do HTTP.

Tabela 36 URI do recurso *operation* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/operation/*	x			

Fonte: o autor

Recurso "*convert*"

O recurso *convert* é utilizado para configurar operações de substituições de dados. Em especial, este recurso *convert*, oferece suporte apenas substituição direta baseada em três parâmetros, são eles: coluna, origem, destino. A Tabela 37, fornece a URI e métodos suportados do HTTP.

Tabela 37 URIs do recurso *convert* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/operation/convert*		x	x	

Fonte: o autor

A Tabela 38, apresenta as propriedades, descrições e restrições quanto ao uso da propriedade.

Tabela 38 Propriedades do recurso *convert*

Propriedade	**Restrição**	**Detalhe**
dct:identifier	automático	Identificador único e sequencial.
dct:issued	automático	Data de publicação.
dct:modifield	automático	Última data de alteração.
dct:creator	automático	Descreve entidade principal responsável.
b:remote_file	obrigatório	URL do arquivo.
dct:identifier	obrigatório	Coluna de dados existente nos dados extraídos.
b:from	obrigatório	Valor de origem.
b:to	obrigatório	Valor a ser alterado.
b:order	obrigatório	Ordem das operações de transformações.

Fonte: o autor

Recurso "*select*"

O recurso *select* é utilizado para configurar operação de seleção de dados. Em especial, este recurso *select*, oferece suporte para selecionar colunas de dados extraídos. A Tabela 39, fornece a URI e métodos suportados do HTTP.

Tabela 39 URIs do recurso *select* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/operation/select*		x	x	

Fonte: o autor

A Tabela 40, apresenta as propriedades, descrições e restrições quanto ao uso da propriedade.

Tabela 40 Propriedades do recurso *select*

Propriedade	**Restrição**	**Detalhe**
dct:issued	automático	Data de publicação.
dct:modifield	automático	Última data de alteração.
dct:creator	automático	Descreve entidade principal responsável.
b:orderColumns	obrigatório	Ordem de apresentação das colunas.
b:order	obrigatório	Ordem das operações de transformações.

Fonte: o autor

Recurso "*order*"

O recurso *order* é utilizado para configurar operação de ordenação de dados. A ordenação é com base nas colunas de dados, pois são adotados dois tipos de ordenação, deste modo é possível ordenar de maneira crescente ou decrescente. A Tabela 41, fornece a URI e métodos suportados do HTTP.

Tabela 41 URIs do recurso *order* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/operation/order*		x	x	

Fonte: o autor

A Tabela 42, apresenta as propriedades, descrições e restrições quanto ao uso da propriedade.

Tabela 42 Propriedades do recurso *order*

Propriedade	**Restrição**	**Detalhe**
dct:issued	automático	Data de publicação
dct:modifield	automático	Última data de alteração
dct:creator	automático	Descreve entidade principal responsável
dct:identifier	obrigatório	Coluna de dados existente nos dados extraídos.
b:orderType	obrigatório	Forma de ordenação da coluna. Exemplo: crescente (prefixo *asc*) ou decrescente (prefixo *desc*).
b:order	obrigatório	Ordem das operações de transformações.

Fonte: o autor

Recurso "*transformation*"

O recurso *transformation* é utilizado para executar as operações de transformação de dados. Em especial, cada uma das operações configuradas poderá ser executada por vez. A Tabela 43, fornece as URIs e métodos suportados do HTTP.

Tabela 43 URI do recurso *transformation* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/operation/convert/transform*		x	x	
2	*/catalog/:id/dataset/:id/resource/:id/operation/select/transform*		x	x	
3	*/catalog/:id/dataset/:id/resource/:id/operation/order/transform*		x	x	

Fonte: o autor

Recurso "*resourcemetadata*"

O recurso *resourcemetadata* é utilizado para configurar os metadados do estado do recurso. Os metadados são incluídos com base em algumas propriedades, em especial a que define uma URI do metadado. A Tabela 44, fornece as URIs e métodos suportados do HTTP.

Tabela 44 URIs do recurso *resourcemetadata* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/resourcemetadata/*	x	x		
2	*/catalog/:id/dataset/:id/resource/:id/resourcemetadata/:id*	x	x	x	

Fonte: o autor

A Tabela 45, apresenta as propriedades, descrições e restrições quanto ao uso da propriedade.

Tabela 45 Propriedades do recurso *resourcemetadata*

Propriedade	**Restrição**	**Detalhe**
dct:issued	automático	Data de publicação.
dct:modifield	automático	Última data de alteração.
dct:creator	automático	Descreve entidade principal responsável.
dct:identifier	obrigatório	Coluna de dados existente nos dados extraídos.
foaf:homepage	obrigatório	URI que descreve *metadado*.

Fonte: o autor

Recurso "*columns*"

O recurso *columns* é utilizado para as colunas de um conjunto de dados extraídos. Este recurso contribui na atividade de enriquecimento semântico, com base nas colunas retornadas são incluídos os metadados. A Tabela 46, fornece a URI e métodos suportados do HTTP.

Tabela 46 URIs do recurso *columns* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/catalog/:id/dataset/:id/resource/:id/columns/*		x		

Fonte: o autor

Recurso "*metadata*"

O recurso *metadata* é utilizado para manter os metadados do repositório de central. Para cada metadado criado uma URI deverá ser definida. A Tabela 47, fornece as URIs e métodos suportados do HTTP.

Tabela 47 URIs do recurso *metadata* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/metadata*	x	x		
2	*/metadata/:id*	x	x	x	

Fonte: o autor

A Tabela 48, apresenta as propriedades, descrições e restrições quanto ao uso da propriedade.

Tabela 48 Propriedades do recurso *metadata*

Propriedade	**Restrição**	**Detalhe**
dct:issued	automático	Data de publicação
dct:modifield	automático	Última data de alteração
dct:creator	automático	Descreve entidade principal responsável
dct:identifier	obrigatório	Coluna de dados existente nos dados extraídos.
dct:comment	opcional	Comentário adicional ao *metadado*.
dct:definition	obrigatório	Descreve o metadado.
dct:language	obrigatório	Informa o idioma que foi escrito.
dct:type	obrigatório	Área de domínio.
dct:keyword	opcional	Palavras chaves.
foaf:homepage	opcional	Página *Web* com explicação sobre *o metadado*.

Fonte: o autor

Recurso "Data"

O recurso *data* é utilizado para disponibilizar os dados e metadados na Web de maneira pública. Para cada recurso do tipo *resource* criado, uma URI de conjunto de dados será criada. A Tabela 49, fornece as URIs e métodos suportados do HTTP.

Tabela 49 URIs do recurso *Data* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/data*		x		
2	*/data/:id*		x		
3	*/data/:id/csv*		x		
4	*/data/:id/xml*		x		
5	*/data/:id/json*		x		
6	*/data/:id/html*		x		

Fonte: o autor

Recurso "*resourcemetadata*"

O recurso *resourcemetadata* é utilizado para disponibilizar os metadados na Web de maneira pública. A Tabela 50, fornece as URIs e métodos suportados do HTTP.

Tabela 50 URIs do recurso *metadata* e métodos do HTTP

URIs canônicas do Recurso		C	R	U	D
1	*/data/:id/metadata*		x		
2	*/data/:id/metadata/csv*		x		
3	*/data/:id/metadata/xml*		x		
4	*/data/:id/metadata/json*		x		
5	*/data/:id/metadata/html*		x		

Fonte: o autor

C

Descrição da Avaliação Experimental

O apêndice C apresenta a descrição detalhada da avaliação experimental que foi utilizada na abordagem desse trabalho. A seguir, será explicado como foi realizado o experimento para avaliar a implementação da abordagem.

Criação do conjunto de dados "*dataset*"

As informações necessárias para criar o *dataset* são apresentadas na Figura 36.

Propriedade	**Valor**
dct:title	Árvores Tombadas
dct:description	O tombamento das árvores consiste em um instrumento legal de preservação de espécies vegetais de porte arbóreo, fundamentado pelo art. 7 da Lei Federal nº 4.771/65 do Código Florestal e reafirmado pela Lei Municipal nº 15.072/88. Segundo essa lei, qualquer árvore poderá ser declarada imune de corte, mediante Ato do Poder Público, por motivo de localização, beleza, raridade e condição de porta-sementes, bem como boas condições fitossanitárias e área de projeção da copa livre.
dct:identifier	arvores-tombadas
dcat:keyword	arvores, tombadas
dct:language	BR
dcat:contactPoint	*http://cin.ufpe.br/ bifm/contact.vcf*
dct:temporal	
dct:spatial	
dct:accrualPeriodicity	Um exemplo para uso pode ser visto na Figura 33.
dct:publisher	Um exemplo para uso pode ser visto na Figura 34.

Tabela 51 Propriedades do recurso dataset

Figura 33 Exemplo do Atributo Multivalorado da propriedade *dct:accrualPeriodicity*

```
<dct:Frecuency>
        <time:DurationDescription>
                <label>
                        Anual
                </label>
                <time:days>
                        360
                </time:days>
        </time:DurationDescription>
</dct:Frecuency>
```

Fonte: o autor

Figura 34 Exemplo do Atributo Multivalorado da propriedade *dct:publisher*

```
<foaf:Organization>
        <foaf:name>
                EMPREL
        </foaf:name>
        <foaf:mbox>
                contato@emprel.recife.pe.gov.br
        </foaf:mbox>
</foaf:Organization>
```

Fonte: o autor

Figura 35 Conteudo gerado em notação JSON da estrutura do *dataset*

```
{
  "dct:identifier": "arvores-tombadas",
  "dct:title": "rvores Tombadas",
  "dct:description": "O tombamento das arvores consiste em um instrumento legal de preservacao de especies vegetais de porte arboreo, fundamentado pelo art. 7 da Lei Federal n 4.771/65 do Cdigo Florestal e reafirmado pela Lei Municipal n 15.072/88. Segundo essa lei, qualquer arvore poder ser declarada imune de corte, mediante Ato do Poder Publico, por motivo de localizacao, beleza, raridade e condicao de porta-sementes, bem como boas condicoes fitossanitarias e rea de projeo da copa livre.",
  "dcat:keyword": "arvores, tombadas",
  "dct:language": "pt-br",
  \caption{Conteudo gerado em notacao JSON da estrutura do \textit{dataset}}
  \label{fig:cap5-estrutura-dataset-json}
  "dcat:contactPoint": "http://cin.ufpe.br/~bifm/contact.vcf",
  "dct:temporal": "",
  "dct:spatial": "",
  "dct:accrualPeriodicity": {
    "dct:Frecuency": {
      "time:DurationDescription": {
        "label": "Anual",
        "time:days": 365
      }
    }
  },
  "dct:publisher": {
    "foaf:Organization": {
      "foaf:name": "EMPREL",
      "foaf:mbox": "contato@emprel.recife.pe.gov.br"
    }
  }
}
```

Fonte: o autor

Obtida as informações necessárias para criar o *dataset*, será então enviada a requisição para criar um novo estado ao recurso *dataset*. A Figura 36 apresenta a mensagem enviada ao protótipo da abordagem. Esta mensagem envia uma requisição usando o protocolo HTTP/1.1 por meio do método *POST*.

Figura 36 Requisição enviada ao recurso "*dataset*" usando método *POST*

```
POST /api/v1/catalog/emprel/dataset HTTP/1.1
Host: localhost:8080
X-Public: 8f1c7765e183f80125081e3813f9be58
X-Message: 808b65f5a0c257f482b829eace2235bd09951bcf
X-Microtime: 1407096415.737
Content-Type: application/json
Accept: application/json;charset=UTF-8

{
  "dct:identifier": "arvores-tombadas",
  "dct:title": "Arvores Tombadas",
  "dct:description": "O tombamento das arvores consiste em um instrumento legal de preservacao de especies vegetais de porte arboreo, fundamentado pelo art. 7 da Lei Federal n 4.771/65 do Cdigo Florestal e reafirmado pela Lei Municipal n 15.072/88. Segundo essa lei, qualquer arvore poder ser declarada imune de corte, mediante Ato do Poder Publico, por motivo de localizacao, beleza, raridade e condicao de porta-sementes, bem como boas condicoes fitossanitarias e rea de projeo da copa livre.",
  "dcat:keyword": "arvores, tombadas",
  "dct:language": "pt-br",
  "dcat:contactPoint": "http://cin.ufpe.br/~bifm/contact.vcf",
  "dct:temporal": "",
  "dct:spatial": "",
  "dct:accrualPeriodicity": {
    "dct:Frecuency": {
      "time:DurationDescription": {
        "label": "Anual",
        "time:days": 365
      }
    }
  },
  "dct:publisher": {
    "foaf:Organization": {
      "foaf:name": "EMPREL",
      "foaf:mbox": "contato@emprel.recife.pe.gov.br"
    }
  }
}
```

Fonte: o autor

A Figura 37 apresenta a mensagem retornada pelo protótipo da abordagem. Neste protótipo da abordagem, o método *POST* apenas retorna o código de estado HTTP e o atributo "*Location*". Este atributo "*Location*" é responsável por fazer o *hyperlink* para o novo estado do recurso.

Figura 37 Resposta ao pedido do método *POST*

```
HTTP/1.1 201 Created
Location: http://localhost:8080/api/v1/catalog/emprel/dataset/arvores-tombadas
Content-Length: 0
Content-Type: text/html
```

Fonte: o autor

As informações necessárias para criar o novo estado para o recurso "*resource*". Primeiramente, são obtidos os dados e organizados na Tabela 52.

Criação do recurso "*resource*"

O processo para "criar uma instância" (também podendo ser referenciada como "novo estado para recurso") será usado novamente. O método POST será utilizado nas mensagens que serão enviadas ao protótipo da abordagem (ou implementação da abordagem, deste ponto em diante será adotado protótipo da abordagem).

Tabela 52 Propriedades do recurso *Resource*

Propriedade	**Valor**
dct:title	Árvores Tombadas
dct:description	O tombamento das árvores consiste em um instrumento legal de preservação de espécies vegetais de porte arbóreo, fundamentado pelo art. 7 da Lei Federal nº 4.771/65 do Código Florestal e reafirmado pela Lei Municipal nº 15.072/88. Segundo essa lei, qualquer árvore poderá ser declarada imune de corte, mediante Ato do Poder Público, por motivo de localização, beleza, raridade e condição de porta-sementes, bem como boas condições fitossanitárias e área de projeção da copa livre.
dct:identifier	arvores-tombadas-2014-07-21
dcat:keyword	arvores, tombadas
dct:language	BR
dct:accrualPeriodicity	Um exemplo para uso pode ser visto na Figura 33.

Fonte: o autor

Figura 38 Conteúdo gerado em notação JSON da estrutura do *resource*

```
{
  "dct:title": "rvores Tombadas",
  "dct:description": "O tombamento das arvores consiste em um instrumento legal de preservacao de especies vegetais de porte arboreo, fundamentado pelo art. 7 da Lei Federal n 4.771/65 do Codigo Florestal e reafirmado pela Lei Municipal n 15.072/88. Segundo essa lei, qualquer arvore poder ser declarada imune de corte, mediante Ato do Poder Publico, por motivo de localizao, beleza, raridade e condicao de porta-sementes, bem como boas condicoes fitossanitarias e area de projeo da copa livre.",
  "dct:identifier": "arvores-tombadas-2014-07-21",
  "dct:language": "BR",
  "dct:keyword": "arvores, raridade",
  "dct:accrualPeriodicity": {
    "dct:Frecuency": {
      "time:DurationDescription": {
        "label": "Anual",
        "time:days": 365
      }
    }
  }
}
```

Fonte: o autor

Obtida as informações necessárias (Figura 38) para utilizar o método *POST*, será então enviada a requisição para criar um novo estado ao recurso *resource*. Este método é utilizado para criar um novo estado ao recurso. A Figura 39 apresenta a mensagem enviada ao protótipo da abordagem. Esta mensagem envia uma requisição usando o protocolo HTTP/1.1 por meio do método *POST*.

Figura 39 Criar novo recurso "*resource*" (arvores-tombadas-2014-07-21)

```
POST /api/v1/catalog/emprel/dataset/arvores-tombadas/resource/ HTTP/1.1
Host: localhost:8080
X-Public: 8f1c7765e183f80125081e3813f9be58
X-Message: 6ef63607f5adb8e7079f9cef449ec976baa83c3f
X-Microtime: 1407096640.075
Content-Type: application/json
Accept: application/json;charset=UTF-8

{
  "dct:title": "arvores Tombadas",
  "dct:description": "O tombamento das arvores consiste em um instrumento legal de preservacao de especies vegetais de porte arboreo, fundamentado pelo art. 7 da Lei Federal n. 4.771/65 do Codigo Florestal e reafirmado pela Lei Municipal n. 15.072/88. Segundo essa lei, qualquer arvore poder ser declarada imune de corte, mediante Ato do Poder Publico, por motivo de localizacao, beleza, raridade e condicao de porta-sementes, bem como boas condicoes fitossanitarias e area de projecao da copa livre.",
  "dct:identifier": "arvores-tombadas-2014-07-21",
  "dct:language": "BR",
  "dct:keyword": "arvores, raridade",
  "dct:accrualPeriodicity": {
    "dct:Frecuency": {
      "time:DurationDescription": {
        "label": "Anual",
        "time:days": 365
      }
    }
  }
}
```

Fonte: o autor

A Figura 40 apresenta a mensagem retornada pelo protótipo da abordagem.

Figura 40 Resposta do servidor ao pedido da criação do *resource*

```
HTTP/1.1 201 Created
Location: /api/v1/catalog/emprel/dataset/arvores-tombadas/resource/arvores-tombadas-2014-07-21
Content-Length: 0
Content-Type: text/html
```

Fonte: o autor

Uma vez que o novo estado para o recurso "*resource*" foi criado, é preciso criar agora o conector para importar os dados. Será então criado um conector do tipo "*postgresql*" para extrair os dados da fonte de origem.

Criação do conector *Postgresql*

Primeiramente, são obtidos os dados necessárias para criar o novo estado para o recurso *postgresql* (Tabela 53).

Tabela 53 Propriedades recurso *Postgresql*

Propriedade	**Valor**
b:db_user	*root*
b:db_password	123
b:db_host	*localhost*
b:db_name	DAG
b:db_port	5433
b:db_query	*select nome_popular, numero, familia, nome_cientifico, este, endereco, norte, decreto, microregiao, rpa, identifica, observacao, latitude, longitude from arvores_tombadas*;

Fonte: o autor

Figura 41 Conteúdo gerado em notação JSON da estrutura do "*postgresql*"

```
1 {
2   "b:db_user": "root",
3   "b:db_password": "123",
4   "b:db_host": "localhost",
5   "b:db_name": "DAG",
6   "b:db_port": "5433",
7   "b:db_query": "select nome_popular, numero, familia, nome_cientifico, este, endereco,↩
       norte, decreto, microregiao, rpa, identifica, observacao, latitude, longitude from↩
       arvores_tombadas;"
8 }
```

Fonte: o autor

Obtida as informações necessárias (Figura 41), será então enviada a requisição para criar um novo estado ao recurso "*postgresql*". A Figura 42 apresenta a mensagem enviada ao protótipo da abordagem. A mensagem por meio do HTTP/1.1 e método *POST*.

Figura 42 Criar novo estado ao recurso "*postgresql*" (arvores-tombadas-2014-07-21)

```
1  POST /api/v1/catalog/emprel/dataset/arvores-tombadas/resource/arvores-tombadas↩
       -2014-07-21/connector/postgresql HTTP/1.1
2  Host: localhost:8080
3  X-Public: 8f1c7765e183f80125081e3813f9be58
4  X-Message: 361c53150e09606470a43fc03fa7a1db43f48425
5  X-Microtime: 1406455198.556
6  Content-Type: application/json
7  Accept: application/json;charset=UTF-8
8
9  {
10   "b:db_user": "root",
11   "b:db_password": "123",
12   "b:db_host": "localhost",
13   "b:db_name": "DAG",
14   "b:db_port": "5433",
15   "b:db_query": "select nome_popular, numero, familia, nome_cientifico, este, endereco,↩
       norte, decreto, microregiao, rpa, identifica, observacao, latitude, longitude from↩
       arvores_tombadas;"
16 }
```

Fonte: o autor

A Figura 43 apresenta a mensagem retornada pelo protótipo da abordagem.

Figura 43 Resposta do servidor ao pedido da criação do conector *postgresql*

```
HTTP/1.1 201 Created
Location: /api/v1/catalog/emprel/dataset/arvores-tombadas/resource/arvores-tombadas←↩
    -2014-07-21/connector/postgresql/1
Content-Length: 0
Content-Type: text/html
```

Fonte: o autor

Extração dos dados

Criado o conector *postgresql* será necessário agora executar a operação de extração por meio do método *PUT*. Para dar início à extração será utilizada a mensagem HTTP da Figura 44, que envia uma requisição usando o protocolo HTTP/1.1 e método *PUT*.

Figura 44 Criar novo recurso "*extração*" (arvores-tombadas-2014-07-21)

```
PUT /api/v1/catalog/emprel/dataset/arvores-tombadas/resource/arvores-tombadas←↩
    -2014-07-21/connector/postgresql/1/extraction HTTP/1.1
Host: localhost:8080
X-Public: 8f1c7765e183f80125081e3813f9be58
X-Message: 1dcb9d980d0b0ae1862c5d6ba9f68184e69d8c65
X-Microtime: 1406455578.335
Content-Type: application/json
Accept: application/json;charset=UTF-8
```

Fonte: o autor

A Figura 45 apresenta a mensagem retornada pelo protótipo da abordagem. Nesta implementação da abordagem, o método *PUT* apenas retorna o código de estado HTTP.

Figura 45 Resposta do servidor ao pedido da criação do *dataset*

```
HTTP/1.1 202 Accepted
Content-Length: 0
Content-Type: text/html
```

Fonte: o autor

Como não será necessário realizar a limpeza nos dados, então é iniciada a criação dos tipos de representações. Será solicitado a criação da representação dos dados em notação CSV.

Criação da representação dos dados

Executada a extração será necessário agora executar a operação de carregamento dos dados por meio do método *PUT*. Para dar início à execução será utilizado a mensagem HTTP da Figura 46, que envia uma requisição usando o protocolo HTTP/1.1 por meio do método *PUT*.

Figura 46 Criar novo recurso "*loading*" (arvores-tombadas-2014-07-21)

```
POST /api/v1/catalog/emprel/dataset/arvores-tombadas/resource/arvores-tombadas↩
    -2014-07-21/loading/csv HTTP/1.1
Host: localhost:8080
X-Public: 8f1c7765e183f80125081e3813f9be58
X-Message: 9efb0cb557a5dfe92ec6fdb932089ba05e43980c
X-Microtime: 1406455985.624
Content-Type: application/json
Accept: application/json;charset=UTF-8
```

Fonte: o autor

A Figura 47 apresenta a mensagem retornada pelo protótipo da abordagem.

Figura 47 Resposta do servidor ao pedido da criação do *dataset*

```
HTTP/1.1 201 Created
Location: /api/v1/catalog/emprel/dataset/arvores-tombadas/resource/arvores-tombadas↩
    -2014-07-21/loading/csv
Content-Length: 0
Content-Type: text/html
```

Fonte: o autor

Uma vez criado o documento contendo a representação dos dados escolhida, será possível criar uma distribuição. A distribuição é obrigatória mas, é capaz de organizar os documentos e contribuir para futuras integrações de dados. Este conjunto de dados (estado do recurso) poderia ter várias representações e todas elas poderia ser cadastradas seguindo estes mesmos passos.

Criação da distribuição

As informações necessárias para criar o novo estado para o recurso distribuição. Primeiramente são obtidos os dados e organizados na Tabela 54.

Tabela 54 Propriedades do recurso *Distribution*

Propriedade	**Valor**
dct:identifier	arvores-tombadas-csv
dct:title	Distribuição contendo representação sobre árvores tombadas
dct:description	Esta distribuição possui os dados necessários para identificação das árvores tombadas
dct:license	
dct:rights	
dcat:accessURL	
dcat:downloadURL	http://localhost:8080/api/v1/catalog/emprel/dataset/arvores-tombadas/resource/arvores-tombadas-2014-07-21/loading/csv
dcat:mediaType	text/csv
dcat:format	
dcat:byteSize	

Fonte: o autor

Figura 48 Conteúdo gerado em notação JSON da estrutura do *dataset*

```
{
    "dct:identifier": "arvores-tombadas-csv",
    "dct:title": "Distribuicao contendo representacao sobre arvores tombadas",
    "dct:description": "Esta distribuicao possui os dados necessarios para identificao das arvores tombadas",
    "dct:rights": "",
    "dcat:accessURL": "",
    "dcat:downloadURL": "http://localhost:8080/api/v1/catalog/emprel/dataset/arvores-tombadas/resource/arvores-tombadas-2014-07-21/loading/csv",
    "dcat:mediaType": "text/csv"
}
```

Fonte: o autor

Obtida as informações necessárias (Figura 48) para utilizar o método *POST*, será então enviada a requisição para criar um novo estado ao recurso *resource*. Este método é utilizado para criar um novo estado ao recurso. A Figura 49 apresenta a mensagem enviada ao protótipo da abordagem. Esta mensagem é enviada por requisição usando o HTTP/1.1 e método *POST*.

Figura 49 Criar novo recurso "*distribuicao*" (arvores-tombadas-2014-07-21)

```
POST /api/v1/catalog/emprel/dataset/arvores-tombadas/distribution HTTP/1.1
Host: localhost:8080
X-Public: 8f1c7765e183f80125081e3813f9be58
X-Message: 639cfea3cbee2fa3e861fd3e525c3bb9c4b841eb
X-Microtime: 1407098587.089
Content-Type: application/json
Accept: application/json;charset=UTF-8

{
    "dct:identifier": "arvores-tombadas-csv",
    "dct:title": "Distribuicao contendo representacao sobre arvores tombadas",
    "dct:description": "Esta distribuicao possui os dados necessarios para identificao das arvores tombadas",
    "dct:rights": "",
    "dcat:accessURL": "",
    "dcat:downloadURL": "http://localhost:8080/api/v1/catalog/emprel/dataset/arvores-tombadas/resource/arvores-tombadas-2014-07-21/loading/csv",
    "dcat:mediaType": "text/csv"
}
```

Fonte: o autor

A Figura 50 apresenta a mensagem retornada pelo protótipo da abordagem.

Figura 50 Resposta do servidor ao pedido da criação do *distribution*

```
HTTP/1.1 201 Created
Location: /api/v1/catalog/emprel/dataset/arvores-tombadas/distribution/arvores-tombadas-csv
Content-Length: 0
Content-Type: text/html
```

Fonte: o autor

Acessando o conteúdo por meio do recurso "*data*"

Será agora solicitado a coleção de estados do recurso "*data*" por meio de uma requisição usando o método *GET*. A representação solicitada será JSON. A Figura 51 apresenta a mensagem enviada ao protótipo da abordagem. Esta mensagem envia uma requisição usando o HTTP/1.1 e método *GET*.

Figura 51 Requisição enviada ao recurso "*catalog*" para obter coleção de estados

```
GET /api/v1/data/emprel/arvores-tombadas/arvores-tombadas-2014-07-21  HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Accept: application/json;charset=UTF-8
```

Fonte: o autor

A Figura 52 apresenta a mensagem retornada pelo protótipo da abordagem. A representação do corpo da mensagem está em JSON. O conteúdo retornado na mensagem foi limitado para apenas um registro ou seja, apenas uma linha de dados foi retornada.

Figura 52 Resposta contendo a coleção de estados do recurso *catalog*

```
HTTP/1.1 200 OK
Content-Length: 1100
Content-Type: application/json; charset=UTF-8

[
  {
    "nome_popular": "Barriguda",
    "numero": 10,
    "familia": "Bombacaceae",
    "nome_cientifico": "Ceiba pentandra (L.) Gaertn",
    "este": "29.038.400.000.000.000",
    "endereco": "Rua Julio Lima, agua Fria ao lado do imovel 256",
    "norte": "911.307.000.000.000.000",
    "decreto": "14.288/88",
    "microregiao": 2.2,
    "rpa": 2,
    "identifica": 10,
    "observacao": " ",
    "latitude": "-8,019418",
    "longitude": "-34,901908"
  },
  {
    "nome_popular": "Cajueiro",
    "numero": 11,
    "familia": "Anacardiaceae",
    "nome_cientifico": "Anacardium occidentale L.",
    "este": "28.981.800.000.000.000",
    "endereco": "Rua Padre Roma, 375 ? Ed. Via Mariana ? Parnamirim",
    "norte": "911.151.100.000.000.000",
    "decreto": "14.288/88",
    "microregiao": 3.1,
    "rpa": 3,
    "identifica": 11,
    "observacao": "Erradicao e substituio pela resoluo 04/03 de 5 de maio de 2003. Recm↩
     plantada ? inform. GIM ? 2010.",
    "latitude": "-8,033488",
    "longitude": "-34,907107"
  }
]
```

Fonte: o autor