Universidade Federal do Rio de Janeiro
Instituto de Matemática
Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais
Programa de Pós-Graduação em Informática

SEMANTICHUB: UM ARCABOUÇO PARA INTEGRAÇÃO DE DADOS PRODUZIDOS POR OBJETOS DA INTERNET DAS COISAS NA WEB DE DADOS INTERLIGADOS

Fabrício Firmino de Faria

Rio de Janeiro
2013

Fabrício Firmino de Faria

SEMANTICHUB: UM ARCABOUÇO PARA INTEGRAÇÃO DE DADOS PRODUZIDOS POR OBJETOS DA INTERNET DAS COISAS NA WEB DE DADOS INTERLIGADOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Instituto de Matemática e Instituto Tércio Pacciti, Universidade Federal do Rio de Janeiro, como requisito parcial à obtenção do título de Mestre em Informática.

Universidade Federal do Rio de Janeiro

Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais

Programa de Pós-Graduação em Informática

Orientador: Prof. Maria Luiza Machado Campos, Ph.D

Coorientador: Prof. Paulo de Figueiredo Pires, DSc

Rio de Janeiro

2013

Faria, Fabrício Firmino de
F224s SEMANTICHUB: Um arcabouço para integração de dados produzidos por objetos da internet das coisas na web de dados interligados. Fabrício Firmino de Faria. - - Rio de Janeiro, 2013
79 f.

Orientador: Prof. Maria Luiza Machado Campos, Ph.D

Coorientador: Prof. Paulo de Figueiredo Pires, DSc

Dissertação (mestrado)- Universidade Federal do Rio de Janeiro
Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais
Programa de Pós-Graduação em Informática, 2013.

1. Web Semântica. 2. Internet das Coisas. 3. Machado Campos, Maria Luiza, orient. II. de Figueiredo Pires, Paulo, coorient. III. Título

Fabrício Firmino de Faria

SEMANTICHUB: UM ARCABOUÇO PARA INTEGRAÇÃO DE DADOS PRODUZIDOS POR OBJETOS DA INTERNET DAS COISAS NA WEB DE DADOS INTERLIGADOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Instituto de Matemática e Instituto Tércio Pacciti, Universidade Federal do Rio de Janeiro, como requisito parcial à obtenção do título de Mestre em Informática.

Aprovado em: 28 de agosto de 2013, Rio de Janeiro.

Prof. Maria Luiza Machado Campos, Ph.D, UFRJ

Prof. Paulo de Figueiredo Pires, DSc, UFRJ

Prof. Adriana Santarosa Vivacqua, DSc, UFRJ

Prof. Silvana Rossetto, DSc, UFRJ

Prof. Daniel Schwabe, Ph.D, PUC-RIO

AGRADECIMENTOS

Talvez a melhor forma de começar a fazer o agradecimento de um trabalho não seja com uma reclamação, mas dizer que fui feliz nas escolhas que fiz no mestrado e na forma como eu o conduzi não seria uma verdade, mas ainda assim posso dizer que tive muita sorte. Talvez não pelo trabalho que fiz, pois como todo autor sempre fica o sentimento de que algo ainda está faltando, do que foi feito não é o suficiente. Fui muito feliz pelas pessoas maravilhosas que pude conhecer durante esses anos e ao conhecimento que me foi apresentado.

O primeiro agradecimento será para aqueles que talvez tenham sofrido um pouquinho mais do que os outros, minha família. Aos meus pai e mãe que, mesmo aos 28 anos, sempre me apoiaram, muitas vezes até com dinheiro, já que uma bolsa de mestrado está muito longe da realidade de uma vida com dignidade. Minha esposa, que doou muito tempo, e serei enfático neste muito, para que eu pudesse concluir este trabalho. O que posso dizer mais do que um eu amo muito vocês, sem dúvidas são meus pilares.

O segundo agradecimento irá para os amigos que me acompanharam durante o mestrado. A professora Maria Luiza, por ter sido sempre tão gentil comigo, apesar do meu gênio difícil. A professora Jonice, pois nunca conheci um ser humano tão doce e gentil. A professora Adriana, por ter proporcionado as aulas mais divertidas que eu tive. Ao professor João Carlos, me arrependo de não ter continuado o trabalho com ele, é uma pessoa tão íntegra e inteligente, que acho muito difícil encontrar outro como ele.

Aos amigos Bianca, Expedito, Rubinho e Inês. Sem dúvida vocês fazem parte das pessoas mais especiais que em tenha trabalhado algum dia, tanto pela inteligência incrível de cada um, quanto pela postura de ajuda nos momentos em que ficamos para baixo. É sério, sem vocês eu não teria conseguido chegar aqui.

Um agradecimento especial ao Cláudio Miceli, afinal, não é qualquer um que fica até de noite em uma sexta feira ajudando a revisar a sua dissertação. Tiago e Rafael, também não posso deixar de falar de vocês que sempre foram excelente companhia. Na minha lista não poderia deixar de contar pessoas que considero como irmãos, então fica a referência para o Fabio e a Evelyn, o casal mais "mojinho"de todos.

A galera do laboratório Dismo, Diego, Paty e Pedro, muito obrigado por sua companhia. A galera de MASI também não pode ser esquecida, um muito obrigado

para meus amigos Yanko, Tinnus e Mariam. Meu amigo Edgar, quase não lhe chamo de padre, para mim o título meu amigo precede qualquer formalidade, e assim eu lhe vejo. Meu amigo Serra, que me apoiou nos momentos de conflito mental, um muito obrigado. Minhas amigas Carol e Fernanda, os congressos que fomos foram os melhores, não posso nem mais pensar em submarinos.

Um obrigado especial aos amigos que fiz enquanto estive na Irlanda, trabalhando no DERI. André, pelo papos mais filosóficos que algum dia eu tenha tido, a Lauane e ao Murilo (meu querido!), muito obrigado pela companhia. Mesmo sabendo que provavelmente nunca vão ler isto, agradeço aos professores Edward Curry e Sean O'Rian pela oportunidade de ir para o DERI. E um agradecimento para meu novo amigo Yongrui Qin, pelas horas de diversão e pelos links que recebo até hoje e ao Danilo, nunca vou esquecer do SMDC2KBCB.

Também não posso deixar de agradecer a uma pessoa que neste ponto ainda não conheci, mas que num futuro será extremamente importante para a conclusão desse trabalho. Amiga Samara, espero muito que 2015 chegue logo para que possamos ser amigos. E espero que em 2017 você consiga me ajudar a completar o mestrado com a entrega do meu texto. Você nem existe ainda na minha vida mas quero deixar o meu muito obrigado adiantado.

Gostaria de agradecer também a banca, pela gentileza de aceitar o convite. Muito obrigado professor Daniel Schwabe, professora Silvana e professora Adriana. Agradeço também ao professor Paulo Pires que aceitou me coorientar. Agradeço ao secretário do programa, Aníbal, por todos os "galhos quebrados".

Por fim, apenas para não ter que ouvir reclamações durante horas, muito obrigado Miguel Gabriel (mesmo que não saiba o que eu estou agradecendo). Por favor, não me perturbe mais!

RESUMO

Estimativas do Cluster of European Research Projects on the Internet of Things (CERP-IoT) apontam que objetos do mundo real produzirão a maior parte dos dados na Web a partir de 2020. Os diferentes formatos de representação de dados utilizados entre objetos e a semânticas distinta e não explícita para computadores (disponível apenas em documentos compreensíveis apenas para humanos) impossibilita o processamento automático dos dados, necessário para a manipulação e análise de grandes volumes de dados heterogêneos. A proposta apresentada neste trabalho é criar um arcabouço, denominado SemanticHub, capaz de capturar, padronizar estruturas e explicitar o significado dos dados produzidos por objetos seguindo os princípios da Web Semântica, permitindo desta forma a computabilidade destes dados. O diferencial do SemanticHub frente as demais propostas presentes na literatura é que este é capaz de lidar com dados produzidos por milhares de objetos sem que seja necessário modificações nos mecanismos de comunicação destes objetos, mantendo atualizado um repositório com dados que seguem os princípios da Web Semântica.

Palavras-chaves: Internet das Coisas, Web das Coisas, Dados Abertos Interligados, Redes de Sensores Semânticas

ABSTRACT

An estimation of the Cluster of European Research Projects on the Internet of Things (CERP-IoT) point that objects form the real world will produce the major part of data in the Web in 2020. The different representation of data formats used among objects and its different semantics that are not explicit for computers (available only in formats readable by humans) make impossible the automated data processing, needed to manipulate and to analyze huge amount of heterogeneous data. The approach proposed in this work is to create a framework, named SemanticHub, able to capture, standardize structures and explicit the meaning of produced data by objects following Web Semantics principles, allowing, in this way, data compatibility. SemanticHub's differs from the other proposals present in the literature is that SemanticHub is able to deal with data produced by thousands of objects without the need of modifying in the communication mechanisms among objects, maintaining an updated repository with data that follows the principle of Web semantics.

Keywords: Internet of Things, Web of Things, Linked Open Data, Semantic Sensor Networks

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GQM	Goal-Question-Metric
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
JSON	JavaScript Object Notation
LOD	Linked Open Data
OWL	Ontology Web Language
RDF	Resource Description Framework
REST	Representional State Transfer
WoT	Web of Things
WS-*	Big Web Services
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Criada com propósitos militares para manter as comunicações em caso de ataque, a Internet transformou-se na rede mundial de computadores conectando mais de um bilhão de dispositivos em 2009 (SUNDMAEKER et al., 2010). Tornou-se assim, uma rede de propósito geral abrigando inúmeras aplicações, como por exemplo, mecanismos de correio eletrônico, mecanismos de transferência de arquivos e mais recentemente como plataforma para integração de dispositivos do mundo real.

O paradigma que propõe a presença de objetos na Internet é denominado Internet das Coisas (Internet of Things: IoT). Seu principal objetivo é que esses objetos possam interagir e cooperar com outros dispositivos que possuam objetivos em comum (ATZORI; IERA; MORABITO, 2010). Segundo dados da International Telecomunication Union (ITU), a IoT é um dos maiores potenciais de revolução tecnológica para o futuro da computação e comunicação (UNION, 2005). Neste cenário, objetos do mundo físico se tornarão os maiores produtores e consumidores de dados na Internet.

A IoT pode ser entendida como a conjunção entre diversas visões, como apresentado no diagrama da Figura1 adaptado do trabalho de (ATZORI; IERA; MORABITO, 2010). Dentro da visão orientada a Internet existe a proposta da Web das Coisas (Web of Things: WoT), que tem como objetivo que todos os objetos utilizem os protocolos e tecnologias da Web como padrões de comunicação (FRANÇA, 2012), ou seja, todos os objetos estarão integrados no nível de aplicação.

A Web é o sistema mais bem sucedido da Internet. Criada por Tim Berners-Lee como uma proposta para a publicação de documentos interconectados (hipertexto). A Web tornou-se uma plataforma para disponibilização de serviços e integração de dados. No futuro é esperado que a Web se configure como uma base de dados global, armazenando os dados de forma estruturada e com significado explícito. Estas características irão permitir que computadores possam processar vastos volumes de dados de forma automatizada.

A capacidade de manipular grandes quantidade de dados de forma automatizada vem fazer frente ao problema que será gerado quando um grande número de dispositivos estiverem integrados na Web. (SUNDMAEKER et al., 2010) estimam que em 2020 entre 50 a 100 bilhões de dispositivos estarão acessíveis na Web, levando-se em consideração apenas dispositivos com capacidade de processamento. Se forem levados em consideração todos os objetos com capacidade de comunicação, esse número pode

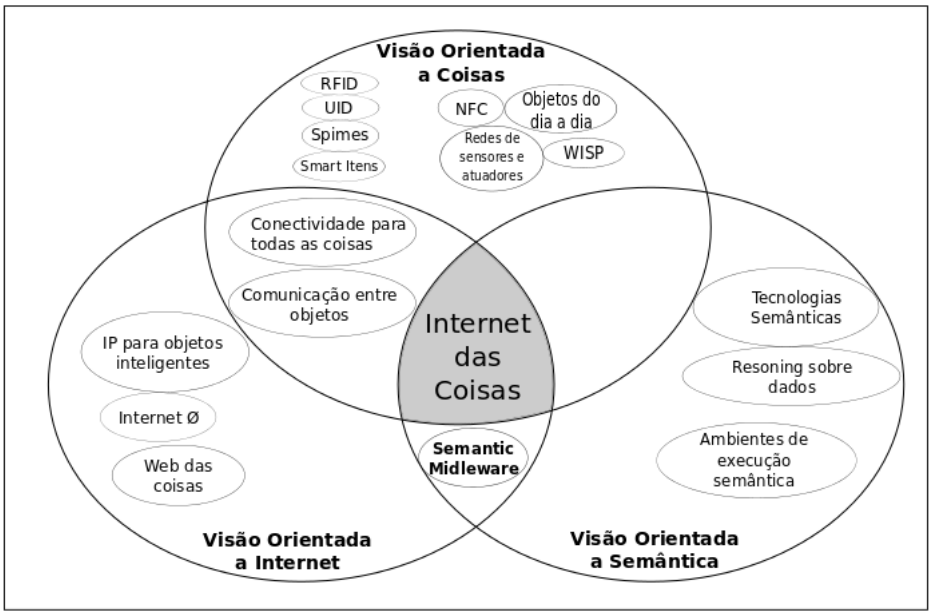


Figura 1 – Visões da Internet das Coisas (ATZORI; IERA; MORABITO, 2010)

ultrapassar a ordem dos 100 bilhões.

O processamento automático de dados por computadores é um assunto estudado pela área de Web Semântica (BERNERS-LEE et al., 2001). A Web Semântica surgiu com a meta de adicionar significado aos dados publicados na Web, de forma que possam mais facilmente serem processados por máquina. Sendo capaz de processar todo o conhecimento contido em um documento, a máquina passa a poder efetuar buscas otimizadas retornando apenas as respostas realmente relevantes. Dentre os primeiros métodos utilizados para tentar tornar a Web Semântica uma realidade, podemos citar as buscas facetadas, o uso de tesauros e taxonomias, e a contextualização de documentos através do uso de marcações (tags) (BREITMAN, 2006).

Como a Web Semântica é uma proposta recente, boa parte dos métodos e tecnologias relacionadas estão em desenvolvimento. Visando viabilizar a evolução e a adoção das tecnologias semânticas foram propostos os princípios de Dados Abertos Interligados(LOD, do termo em inglês Linked Open Data) (BIZER, 2009). LOD é definido como um conjunto de boas práticas para a publicação e conexão de dados estruturados na Web. Um ponto central é a utilização de URIs (URI, do termo em inglês, Uniform Resource Identifier) como identificadores únicos para entidades e conceitos descritos por bases de dados na Web. O uso de URIs permite a identificação da individualidade de um item dentre todos os outros descritos na Web além de facilitar a sua consulta.

A nova geração da Web, iniciada pela publicação de dados no formato LOD, denominamos Web de Dados pois, diferentemente do que ocorria até então na Web, a

ligação agora se dá entre dados diversos e não mais entre documentos. Enquanto na versão anterior (Web de Documentos) existem ligações sem nenhum significado associado, nessa nova Web é apresentado um conjunto de ontologias e vocabulários que têm por finalidade descrever não só o que os dados expressam mas o significado das ligações existentes entre eles (PEREIRA, 2012).

A área da Web Semântica sofre do problema de baixa adoção por parte dos desenvolvedores da Web (LANTHALER; GUTL, 2011). Em contrapartida, segundo dados do ProgrammableWeb (ASHTON, 2011), houve um incremento das interfaces baseadas em REST (do termo em inglês Representational State Transfer). Segundo esta fonte 74% das Interface de Programação de Aplicativos (Application Programming Interface: API) são interfaces REST que usam JSON(Javascript Object Notation) como formato de serialização. Interfaces REST utilizam a própria Web como plataforma para implementar mecanismos de integração a partir da chamada de um dos verbos do protocolo HTTP (GET, POST, PUT, DELETE) para execução de um conjunto de instruções sobre um recurso em específico.

Possibilitar que o crescente volume de dados produzidos por objetos na Web possa ser manipulado por computadores de forma automática, fazendo com que grandes volumes de dados de diferentes fontes possam ser utilizados é um desafio para a Web das Coisas. Da mesma forma, possibilitar que a visão Web Semântica seja implementada também é um desafio, principalmente devido a baixa adoção de tecnologias semânticas por parte dos desenvolvedores.

1.1 DEFINIÇÃO DO PROBLEMA

Apesar dos esforços para padronizar os métodos de acesso a objetos na Web através da utilização de serviços Web do tipo REST, só é possível saber o significado e a estrutura dos dados recebidos através da leitura de um documento legível apenas por humanos, como documentos HTML ou documentos PDF, obrigando os desenvolvedores das aplicações a tratar cada fonte de dados de uma maneira específica. Como a tendência apontada pelo ITU é que o número de dispositivos cresça para a casa dos bilhões, a capacidade de se utilizar múltiplas fontes de dados pode ficar comprometida.

Por exemplo, na Tabela 1 são apresentados exemplos de dois dados produzidos por sensores de temperatura. Apesar de serem equivalentes quanto à informação contida, apresentam dois problemas. O primeiro é que a própria estrutura distinta dos dados faz com que seja necessário criar diferentes parsers para cada fonte de dado. O segundo problema é que apesar de representarem a mesma informação, o fato de utilizarem

Tabela 1 – Exemplo de dado produzidos por sensores de temperatura no formato XML

```
<sensor_measure id="1" >
    <value>293</value>
    <unit_of_measure>kelvin</unit_of_measure>
</sensor_measure>
```

```
<sensor_measure id="2" >
    <sensing_data unit="Celsius" >20</sensing>
</sensor_measure>
```

unidades distintas de medida, obriga que esses dados sejam convertidos para serem utilizados.

Outros problemas possíveis seriam as próprias capacidades de precisão das unidades de sensoriamento utilizadas. Isso significa que diferentes unidades de sensoriamento possuem diferenças de medição quanto à acuidade dos dados coletados, o intervalo de atualização destes dados, etc. Todas essas informações são disponíveis em documentos de texto, mas poderiam estar disponíveis em conjunto com os dados.

Uma representação para o cenário apresentado utilizando a representação <sujeito><predicado><objeto> pode ser vista na Tabela 2. Esta representação é amplamente utilizada pelas abordagens semânticas. Outro ponto explorado é a adição de informação sobre os dados publicados, neste caso foram adicionados informações relativas ao intervalo de leitura de cada sensor. Por fim, os descritores destes dados poderiam ser publicados em conjunto com os dados, permitindo que técnicas de processamento automático sejam aplicadas.

Tabela 2 – Representação no formato <sujeito><predicado><objeto> dos dados da Tabela 1

```
<medida-1><tipo><medida_sensoriamento>.
<medida-1><possui valor>"293" .
<medida-1><unidade-medida><kelvin>.
<medida-1><intervalo-leitura-segundos> "1".
```

```
<medida-2><tipo><medida_sensoriamento>.
<medida-2><possui valor> "20" .
<medida-2><unidade-medida><celsius>.
<medida-2><intervalo-leitura-segundos> "15".
```

1.2 OBJETIVO GERAL

O objetivo deste trabalho é apresentar um arcabouço de publicação de dados produzidos por objetos, padronizando a estrutura desses dados e explicitando a semântica

tal que computadores possam processá-los de forma automatizada. Este arcabouço utilizará como mecanismo de comunicação com os objetos as principais propostas para integração de objetos na Web, como serviços Web do tipo REST e interfaces assíncronas de comunicação através de WebHooks. Webhook é um método de integração assíncrona para Web, neste método a aplicação que implementa o Webhook envia para uma aplicação previamente cadastrada os dados através da chamada de um método POST de um serviço disponibilizado pelo cliente. A explicitação de semântica será feita utilizando padrões e tecnologias semânticas como RDF (Resource Description Framework), repositórios de triplas, Sparql-Endpoints e OWL. Por utilizar os métodos recorrentes de integração de objetos na Web, não exigirá modificações na forma como os objetos funcionam, como por exemplo, estes objetos não precisarão de um formato específico para publicação de seus dados e não necessitarão armazenar os dados produzidos ao longo do tempo.

1.3 OBJETIVOS ESPECÍFICOS

A partir de nosso objetivo geral e da proposta apresentada, identificamos um conjunto de objetivos específicos a serem alcançados durante a execução de nosso trabalho:

- Uma arquitetura para publicação de dados de objetos na Web de Dados Interligados
- Desenvolver um arcabouço baseado na arquitetura desenvolvida
- Avaliação da capacidade do arcabouço desenvolvido através de estudo do funcionamento de cada componente, analisando o volume de dados que pode ser consumido dos objetos, o número de transformações de dados que podem ser processadas, o número de triplas RDF que podem ser geradas e os recursos computacionais, como processamento e memória, necessários para realizar estas operações.

1.4 ORGANIZAÇÃO DO TRABALHO

Neste primeiro capítulo desta dissertação são apresentadas as motivações para o desenvolvimento deste trabalho, identificando o problema que será abordado e o objetivo a ser alcançado. No segundo capítulo será feito o levantamento dos conceitos básicos necessários para o entendimento da proposta. No capítulo três serão apresentados trabalhos relacionados, mostrando pesquisas na área buscando tratar do problema em questão.

A partir do capítulo quatro será iniciada a descrição do arcabouço, mostrando sua arquitetura, os componentes que o compõem e como esses componentes se relacionam, justificando a funcionalidade e as estratégias aplicadas. Ainda no capítulo quatro será feita a avaliação de cada componente, mostrando a capacidade e limitação de uma implementação específica do arcabouço. No capítulo cinco será apresentado ainda um cenário motivacional com dados reais produzidos por objetos sendo disponibilizados seguindo o princípio de dados abertos interligados. O capítulo seis apresenta as conclusões obtidas no trabalho.

2 CONCEITOS BÁSICOS

Neste capítulo serão apresentados conceitos sobre Internet das Coisas, Serviços Web, Web Semântica e Dados Abertos Interligados a fim de servir como base para o entendimento deste trabalho.

2.1 INTERNET DAS COISAS

Os primeiros trabalhos sobre IoT (ASHTON, 2011) pregavam uma forma de identificação única de objetos do mundo real na Internet, focando principalmente em tecnologias de identificação por radio frequência. Com o desenvolvimento e a ampla adoção da Internet, os objetivos da IoT foram estendidos para se tornar uma rede de dispositivos interligados capazes de prover tanto dados quanto de receber e reagir a comandos. Segundo (ATZORI; IERA; MORABITO, 2010), a ideia básica para IoT é o conceito de presença pervasiva em torno de nós por uma variedade de coisas ou objetos. Neste mesmo trabalho os autores afirmam que apesar de ser um único paradigma, a IoT apresenta uma gama de visões, sendo elas a visão orientada a Internet, a visão orientada a coisas e a visão orientada a semântica. O paradigma de IoT tem como resultado a convergência entre estas diferentes visões. Na Figura 1 é possível observar um diagrama apresentado estas visões.

De fato, a Internet das Coisas pode ser entendida como uma rede de objetos interconectados, unicamente endereçáveis, baseado em protocolos padrões de comunicação. Como o presente trabalho tem como foco a integração de dispositivos que estão acessíveis na Web (visão orientada a Internet) na Web de Dados (visão orientada a Semântica), estes dois aspectos serão trabalhados nas próximas subseções em detrimento da visão orientada a coisas.

2.2 WEB DAS COISAS

A maior parte dos dispositivos interconectados na Internet utiliza softwares e interfaces proprietárias, o que torna dispendioso a construção de aplicações para interagir com estes dispositivos (GUINARD, 2010). Para facilitar o desenvolvimento de aplicações além de ser necessário utilizar uma linguagem comum a diferentes dispositivos ((GUINARD; TRIFA; WILDE, 2010); (GUINARD, 2010)).

A Web das Coisas propõe a utilização de protocolos WEB como linguagem comum para integração de dispositivos reais no meio digital, permitindo que dados e serviços sejam reutilizados em diferentes aplicações (DUQUENNOY; GRIMAUD; VANDEWALLE, 2009). Utilizar a Web como camada de integração no nível de aplicação permite que técnicas comuns na Web, como navegação, busca, indexação, cache, etc possam ser aplicadas (GUINARD; TRIFA; WILDE, 2010).Uma das abordagens propostas para integrar dispositivos na Web é a utilização dos padrões WS-* (subseção 2.1). Entretanto, a interoperabilidade obtida através dos padrões WS-* é alcançada através da adição de uma camada de software nas aplicações. Esta adição implica em uma maior complexidade, tornando-se uma solução não adequada para ser aplicada em dispositivos com recursos limitados (GUINARD, 2010).

A WoT propõe que a Web seja estendida de modo a incorporar objetos e dispositivos embarcados do mundo real como qualquer outro serviço Web. Essa extensão e feita através da adoção do protocolo HTTP (BERNERS-LEE; FIELDING; FRYSTYK, 1996) como protocolo de aplicação, ou seja, este protocolo deve ser utilizado como interface base para realizar todas as interações com os dispositivos (GUINARD; TRIFA; WILDE, 2010). A WoT foca nos recursos providos pelos objetos, adotando REST e ROA (RICHARDSON; RUBY, 2007) para disponibilizá-los na Web.

Os princípios de REST são aplicados na Web das coisas através de duas abordagens. Na primeira abordagem, quando os dispositivos tem recursos computacionais suficientes, um servidor Web embarcado é executado em cada dispositivo. A segunda abordagem é aplicada quando os recursos computacionais são escassos, neste caso os dispositivos se comunicam com um dispositivo especial, chamado de Gateway, e este dispositivo disponibiliza as interfaces de acesso para a Web (FRANÇA, 2012).

2.3 WEB SEMÂNTICA

O primeiro trabalho sobre Web Semântica foi o artigo publicado por (BERNERS-LEE et al., 2001). No primeiro parágrafo deste trabalho, transcrito abaixo, é apresentado parte da visão da Web Semântica proposta pelos autores:

> “O sistema de entretenimento estava tocando Beatles "We Can Work It Out", quando o telefone tocou. Quando Pete atendeu o telefone abaixo o som para enviar uma mensagem para todos os outros dispositivos locais que tinham um controle de volume. Sua irmã, Lucy, estava na linha de consultório médico: "Mamãe precisa ver um especialista e depois tem que ter uma série de sessões de fisioterapia. Quinzenal ou algo assim. Eu tenho que definir estas anotações no meu agente."Pete imediatamente concorda em compartilhar a carona."

O cenário proposto continua pelos próximos parágrafos do trabalho. De forma resumida, ele mostra uma série de ações comuns no dia a dia que são realizadas de maneira automatizada graças à estruturação e definição do significado dos conceitos e dados utilizados e pela comunicação de objetos do mundo real com o mundo virtual. Estas características permitem a computabilidade e tomada de decisão por parte de programas inteligentes.

A Web Semântica tem como meta ser uma extensão da Web atual, onde as informações terão significado explícito, permitindo que computadores e pessoas possam trabalhar em cooperação. O primeiro passo para permitir a visão proposta pela Web Semântica foi a estruturação de dados na Web. O surgimento das Web APIs possibilitou que sistemas publicassem seus dados para outros sistemas através de serviços Web. O próximo passo para tornar a visão da Web Semântica possível é a estruturação dos dados através de formatos não proprietários comuns e com a explicitação do significado dos mesmos.

O W3C (SEMANTICWEB, 2009) mantém um grupo de trabalho para desenvolvimento de padrões e tecnologias que apoiem o desenvolvimento da Web Semântica. Neste grupo é proposto uma pilha de tecnologias e padrões, como mostrado na Figura 2. Alguns componentes deste diagrama já estão sendo aplicados, como URIs (BERNERS-LEE; FIELDING; MASINTER, 2005) e XML (BRAY et al., 1997), outros, como RDF e SPARQL (PRUD'HOMMEAUX; SEABORNE, 2011), estão em desenvolvimento.

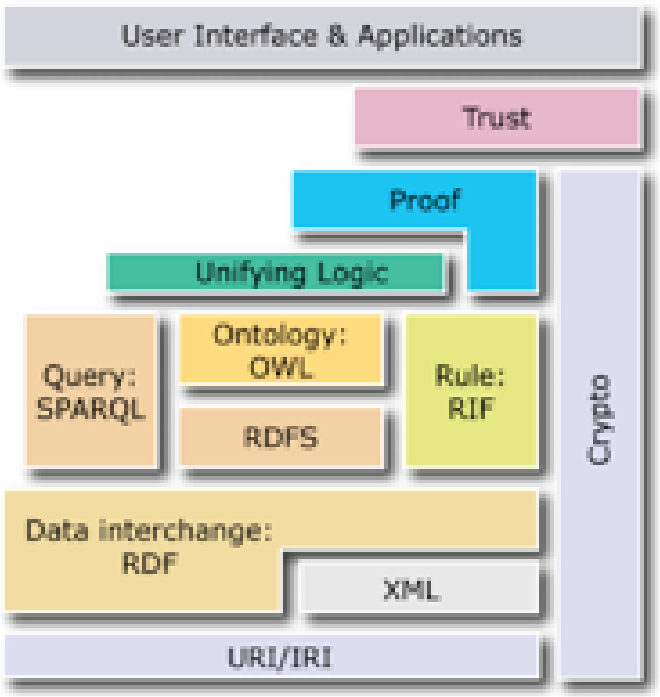


Figura 2 – Pilha de tecnologias da Web Semântica (SEMANTICWEB, 2009)

2.3.1 Web de Dados Interligados

Dados abertos interligados, mais conhecido pela sigla LOD (Linked Open Data) é uma proposta de viabilização da Web Semântica. Ao invés de considerar o uso de todos os componentes da tecnologias e padrões, como mostrado na Figura 2, LOD propõe a utilização de apenas parte desta pilha de tecnologias. LOD tem como principal objetivo a interligação de dados de fontes distintas na Web, permitindo a adição de informações semânticas.

Segundo Bizer, Heath e Berners-Lee (BERNERS-LEE et al., 2001), LOD é um conjunto de boas práticas para publicação de dados estruturados na Web. Para tanto é necessário que os dados sigam quatro princípios básicos (BERNERS-LEE, 2011):

1. Use URIs como identificador único para as coisas
2. Use HTTP URIs para que aplicações e pessoas possam acessar estes dados
3. Os dados devem estar estruturados utilizando padrões abertos (ex: XML, RDF, etc)
4. Os dados devem estar interligados, através da inclusão de links, para outras bases

Dados que segue os princípios de LOD podem apresentar mais uma característica, este dados podem estar representados em RDF (DECKER et al., 1998). Isto permite descrever não somente os dados mas também relacionamentos e atributos. Ontologias e/ou vocabulários são utilizados para definição formal do significado de cada recurso. Com o surgimento de bases de dados em RDF na Web, surgiu o que é ficou conhecido como nuvem de LOD (CYGANIAK; JENTZSCH, 2011). Na Figura 3 é apresentado um diagrama com a versão de 2011 desta nuvem. Cada círculo apresenta uma base de dados em um contexto específico, as arestas representam algum tipo de interligação entre bases.

2.3.2 Arcabouço de Descrição de Recursos

O Resource Description Framework (RDF), é um modelo padrão para interoperabilidade de dados na Web. Fortemente baseado nas premissas do modelo de dados Entidade-Atributo-Valor(do termo inglês Entity-attribute-value, EAV) (DINU; NADKARNI, 2007). A principal característica RDF é sua capacidade para representar dados descritos sob os mais variados esquemas, e por isso, foi escolhido como formato padrão para descrição de dados em LOD.

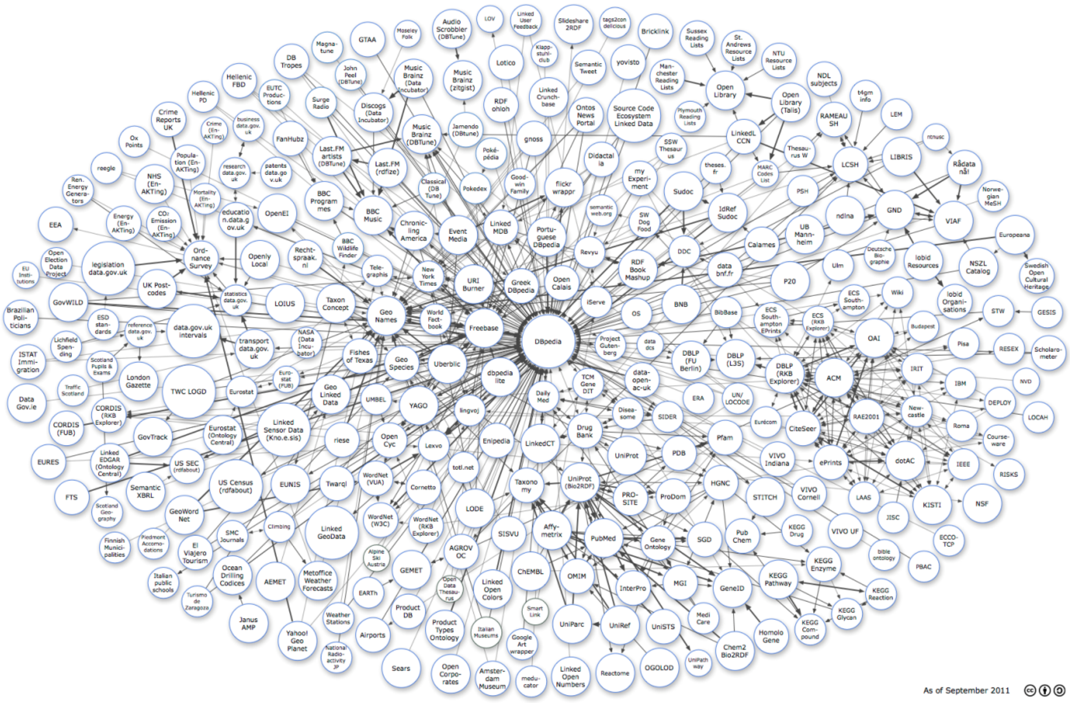


Figura 3 – Representação da nuvem de bases de dados abertos interligados

Dados em RDF são representados no formato de triplas sujeito-predicado-objeto, provendo um modelo flexível para representação de dados baseada em grafos. O sujeito sempre é um recurso a ser descrito. O predicado pode ser de dois tipos: o primeiro tipo inter-relaciona o sujeito (denominado propriedade de dados) com um atributo (literal); o segundo tipo de predicado inter-relaciona o sujeito com outro recurso e é denominado propriedade de objeto. O objeto por sua vez pode assumir dois papeis distintos, ou ele é um literal (um valor propriamente dito) ou um outro recurso. Na Tabela 1 é possível ver dois exemplos de triplas RDF. A primeira apresenta a relação de um recurso de um objeto com um literal. A segunda representa a relação entre dois recursos.

RDF pode ser serializado de diversas formas, as mais comuns são em XML, notação N3, notação n-triplas e turtle (FERNÁNDEZ; MARTÍNEZ-PRIETO; GUTIERREZ, 2010). O formato XML é recomendado pelo W3C. Entretanto, a notação N-triplas e turtle são frequentemente utilizadas pela comunidade de Web Semântica e pela maior parte das ferramentas que suportam RDF.

Para definição de vocabulários em RDF é utilizado o RDF Schema (RDFS) (ALLEMANG; HENDLER, 2008). RDFS permite a descrição de taxonomias de classes e propriedades além de estender definições de elementos do RDF. A Web Ontology Language (OWL) (MCGUINNESS; HARMELEN et al., 2004) tem proposta similar a

RDFS, porém possui maior capacidade de representação para modelos complexos. As bases que seguem os princípios de LOD vem adotando soluções mistas, utilizando tanto OWL quanto RDFS para definição dos modelos de dados. Para armazenar e prover os dados no formato RDF existem repositórios específicos para tal fim, denominados repositórios de triplas. A linguagem de consulta adotada para estes repositório é a linguagem Sparql. Sparql foi proposta inicialmente por Tim Berners-lee (BERNERS-LEE, 2006), tornando-se posteriormente um padrão apoiado pelo W3C [1].

Tabela 3 – Exemplo de dados em RDF (notação N-Tripla)

<http://ex.org/resource1> <http://ex.org/title> "Example".
<http://ex.org/resource1> <http://ex.org/creator> <http://ex.org/resource1>.

2.4 SERVIÇOS WEB

Serviços Web são um conjunto de metodologias que visam a integração entre aplicações, utilizando a Web como plataforma de comunicação. O principal objetivo dos serviços Web é prover mecanismos de comunicação de baixo acoplamento entre sistemas, utilizando padrões abertos para isto (ALONSO et al., 2003). Serviços Web podem ser separados em dois grupos, Big Services, também conhecido como serviços baseado em SOAP, e serviços REST (PAUTASSO; ZIMMERMANN; LEYMANN, 2008).

Big Web services apresentam uma pilha característica de tecnologias associadas (SOAP, WSDL, WS-Addressing, WS-ReliableMessaging, WSSecurity, etc.) ((ALONSO et al., 2003); (WEERAWARANA et al., 2005)), provendo integração tanto através de chamadas de procedimento remotos (Remote Procedure Call (RPC)) quanto pela troca de mensagens assíncronas através de barramentos de mensagens. A comunicação é realizada através da adição de abstrações sobre o protocolo HTTP, servido este apenas para comunicação.

O W3C [2] possui um grupo de trabalho que vem propondo padrões, além de promover a utilização dos serviços baseado em SOAP. Apesar de ser amplamente adotado em ambientes corporativos, este tipo de serviço vem perdendo espaço na Web para serviços do tipo REST (LANTHALER; GUTL, 2011).

Serviços REST promovem integração através de RPC, entretanto, utilizando um conjunto pré-definido de métodos. Sua implementação e utilização, de forma geral,

[1] Basic Geo (WGS84 lat/long) Vocabulary. Disponível em: <http://www.w3.org/2003/01/geo/>. Acesso em: 21 jul. 2013.

[2] Extensible Markup Language (XML). Disponível em: <http://www.w3.org/XML/>. Acesso em: 6 mar. 2013.

é mais simples se comparada com serviços baseados em SOAP. Devido a esta relativa simplicidade e de apresentarem uma menor sobrecarga para transmissão de dados, serviços REST vem sendo largamente empregados para integração de dispositivos na Web. Na subseção 2.5 será detalhado o funcionamento dos serviços REST e nas subseções 2.1 e 2.2 como os dispositivos utilizam estes serviços para compor a proposta de Internet das Coisas (IoT).

2.5 SERVIÇOS WEB REST

O termo REST (Representational State Transfer), foi cunhado por Roy Fielding (FIELDING; TAYLOR, 2000) em sua tese de doutorado, sendo um estilo arquitetural de alto nível que pode ser implementado utilizando tecnologias diversas. Uma implementação da arquitetura REST é a própria WEB. O protocolo HTTP instancia a interface uniforme proposta pelo REST como uma interface especial. Essa interface padronizada é composta pelos próprios verbos HTTP (GET, POST, PUT, DELETE, HEAD, etc) (TILKOV, 2007). Segundo estatísticas do ProgrammableWeb (ASHTON, 2011), em 2010 74% das APIs da WEB eram RESTful, sendo que 45% usam JSON como formato de representação.

REST faz uma definição de como padrões WEB, por exemplo HTTP e URIs, devem ser utilizados. Quando uma aplicação adota os princípios REST ela irá explorar a própria arquitetura da WEB em seu benefício. REST possui cinco princípios básicos, sendo eles: atribua um identificador a todas as coisas; interligue coisas; utilize métodos padronizados; recursos podem ter múltiplas representações; e comunique-se sem estado.

2.5.1 Identificado para Todas as Coisas

Este princípio afirma que todos os recursos de uma aplicação devem ser identificados de forma única. Na Web, o conceito unificado para IDs são as URIs. URIs compõe um namespace global para identificação única de recursos.

URIs também são utilizadas para indicar o escopo da informação, provendo a navegação entre recursos relacionados. Ou seja, um identificador por indicar os sub-recursos relacionados a um determinado recurso. Por exemplo, na Figura 4, o caminho do recurso $user_2 2$, indica que $purchase_1$ é um sub-recurso associado. O mesmo vale para o sub-recurso $purchase_1$ associado ao recurso $item_4$. A URI também pode ser utilizada para endereçar a interação entre recursos sem a necessidade do uso de um corpo da resposta. Na WEB, esta característica permite a utilização de links para recursos através da utilização de esquemas bem conhecidos (MAYER, 2010).

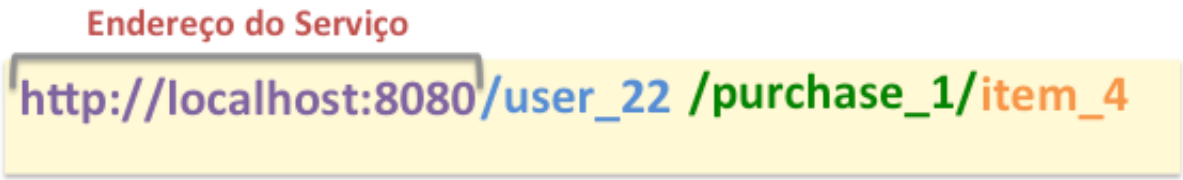


Figura 4 – Exemplo de URI com sub-recursos associados

2.5.2 Recursos Interligados

O segundo princípio de REST tem como descrição formal "Hipermídia como motor de estados". O cerne deste princípio está no conceito de hipermídia, mais especificamente na ideia de links entre recursos. Apesar da ideia de link estar fortemente associada, pelo uso comum, a documentos HTML, eles não estão restritos à utilização exclusiva por humanos. Por exemplo, temos uma representação de um código em XML na listagem 2.1 apresentado abaixo:

```
<ordem self="http://exemplo.com/clientes/1234">
        <qtd>23</qtd>
        <produto ref="http://exemplo.com/produtos/4554"></produto>
        <cliente ref="http://exemplo.com/clientes/1234"></cliente>
</ordem>
```

Figura 5 – Fragmento XML de uma ordem de compra

Neste fragmento XML temos um exemplo de uma ordem de compra efetuada por um cliente identificado através da URI http://exemplo.com/clientes/1234. Esta ordem de compra esta interligada a um outro recurso, neste caso, um produto, através da URI http://exemplo.com/produtos/4554. Uma aplicação poderia facilmente navegar entre este dois recursos através destes links. Outro aspecto importante com esta abordagem é que a URI pode pertencer a uma outra aplicação de uma outra instituição, uma vez que o esquema de nomes é global. Todos os recursos que compõe a Web potencialmente podem ser ligados uns aos outros.

2.5.3 Métodos Padronizados

A habilidade que um browser tem de consumir e apresentar qualquer página disponível na Web é possível, pois existe um grupo limitado de operações que um servidor Web provê. Ou seja, todos os recursos disponíveis na Web possuem uma mesma interface, e o mesmo conjunto de métodos. O protocolo HTTP possui, além dos dois verbos mais conhecidos GET e POST, os verbos PUT, DELETE, HEAD e OPTIONS. Outros verbos existem, mas são raramente utilizados. A função destes verbos, além de algumas garantias sobre seus comportamentos, são definidas na especificação do protocolo HTTP

(SEMANTICWEB, 2009). O princípio da interface uniforme define que o servidor deve ser capaz de determinar o que deve ser feito ao receber uma requisição HTTP em uma URI apenas observando o método presente nessa requisição (PAUTASSO; ZIMMERMANN; LEYMANN, 2008). Os principais métodos e suas funcionalidades estão descritos na Tabela 2.

Tabela 4 – Tabela com os principais métodos do protocolo HTTP

Método	**Descrição**
GET	Usado para recuperar uma informação
POST	Usado para atualizar uma informação
PUT	Usado para criar uma informação
DELETE	Usado para deletar uma informação

2.5.4 Recursos com Múltiplas Representações

O protocolo HTTP permite a separação entre as funções de manipulação de dados e invocação de métodos. Isto possibilita que um cliente que saiba lidar com um formato específico de dado possa interagir com todos os recursos que ofereça uma representação neste formato. Por exemplo, na Figura 7 temos o resultado da chamada a um serviço REST fictício, neste caso a representação associada a este dado é um XML. Na Figura 6 temos a mesma chamada, com única diferença que o dado recebido tem como formato de representação em JSON.

```
{
    "ordem":{
    "self": "http://exemplo.com/clientes/1234",
    "qtd": "23",
    "produto": { "-ref": "http://exemplo.com/produtos/4554"},
    "cliente":   { "-ref":  "http://exemplo.com/clientes/1234" } }
}
```

Figura 6 – Representação de uma ordem de compra no formato JSON

```
<ordem client="http://exemplo.com/clientes/1234">

    <produto ref="http://exemplo.com/produtos/4554" qtd="23"></produto>

</ordem>
```

Figura 7 – Representação de uma ordem de compra no formato XML

O formato de representação não se limita unicamente ao formato de arquivo, podemos ver na listagem 2.3 a chamadas ao mesmo dado com representações em XML da listagem 2.1. Apesar de ambas estarem no formato XML, a estrutura do mesmo é diferente. De forma geral o esquema associado a cada representação de dados bem

como o significado associado a cada unidade da estrutura do documento é proprietária de cada aplicação. Uma das justificativas para o surgimento de LOD é justamente a necessidade de um formato padrão para representação de dados utilizando esquemas não proprietários com semântica bem definida para possibilitar que estes dados sejam automaticamente processados por aplicações.

2.5.5 Comunicação sem Estado

O princípio da comunicação sem estados é um dos aspecto mais importantes de REST. Este princípio prega que, embora uma aplicação possa manter estados de comunicação com os clientes, isto não é aconselhável. A razão principal tem a ver com a escalabilidade. Se a aplicação mantivesse estes estados, o número de clientes que poderiam interagir com esta aplicação seria diretamente reduzido. Outra função importante de não manter estados é isolar o cliente de mudanças da aplicação, desta forma duas ou mais requisições consecutivas não dependem da comunicação com a mesma aplicação.

Quando necessário, a forma proposta (TILKOV, 2007) para lidar com estados é transforma-los em recursos ou delegar a gerência para os clientes que estão utilizando a aplicação.De maneira geral, na maior parte dos cenários onde REST é aplicado, manter a gerência de estados torna a abordagem inútil.

3 TRABALHOS RELACIONADOS

Nesta capítulo são apresentados trabalhos relacionados que propõem métodos de integração de objetos do mundo real na Web de Dados interligados. As propostas identificadas na literatura podem ser separadas em três grandes grupos. O primeiro grupo utiliza tecnologias semânticas para descrever através de conceitos de alto nível os dados produzidos por objetos, permitindo a utilização dos dados sem conhecimento prévio do domínio. O segundo grupo refere-se as propostas que utilizam tecnologias semânticas para fazer a descrição dos serviços de acesso aos objetos. A última abordagem, onde o SemanticHub se enquadra, estão as propostas que fazem uso de intermediadores que aplicam transformações de dados a schemas proprietários para que estes utilizem abordagens semânticas. Também será feito a comparação do SemanticHub com estas propostas, com o objetivo de elucidar os pontos de interseção e diferenciação com os trabalhos levantados.

3.1 REPRESENTAÇÃO DE CONCEITOS SEMÂNTICOS SOBRE DADOS PRODUZIDOS POR OBJETOS

Semantic Streams é o nome do arcabouço produzido por (WHITEHOUSE; ZHAO; LIU, 2006) com o objetivo de permitir que usuários façam consultas declarativas sobre fluxos de dados de sensores baseada não no domínio onde o sensor foi implementado, mas em conceitos de um domínio de alto nível. Esses conceitos são relacionados ao fluxos através de um arcabouço criado no trabalho.

(LIU; ZHAO, 2005) descreve uma arquitetura e modelo de programação para uma plataforma orientada a serviços semânticos. O foco desta abordagem é a manipulação de grandes volumes de dados produzidos por objetos através do mapeamento automático de conceitos ontológicos em fluxos de dados brutos. Este mecanismos é baseado em um serviço de planejamento automático que converte consultas declarativas de usuários em um grafo de composição de serviços.

(FRIEDLANDER; PHOHA, 2002) utiliza a representação semântica de alto nível através de ontologias para criar um método de cognição dinâmica distribuída usando redes de sensores ad hoc. O método apresentado permite a aplicação de técnicas de fusão através da detecção, identificação e rastreamento de alvos em ambientes com muito ruído. Algoritmos foram desenvolvidos para agregação de conhecimentos abstratos em atributos

relevantes com semântica bem definida.

A anotação automática e o raciocínio automatizado para dados de sensores é tratado no trabalho de (WEI; BARNAGHI, 2009). Neste trabalho os autores propõe a ideia de anotação semântica com dados publicados na Web Semântica. Através da utilização de inferência baseada em regras, podem ser criadas anotações de conceitos de alto nível. Estes conceitos são definidos por ontologias previamente definidas para representação de domínios de propósito geral. No mesmo trabalho, os autores também propõe a criação de ligação com outros recursos da Web de dados baseado na derivação de conhecimento a partir das anotações.

As abordagens apresentadas nesta subseção focam na representação de dados produzidos por objetos através da representação de conceitos definidos em ontologias. Cada uma das abordagens é aplicada em um cenário específico, ao contrário do SemanticHub, que tem como meta ser de propósito geral. A definição de conceitos no SemanticHub é feita a priori, o que contrasta com as proposta de anotações automáticas apresentadas, entretanto, ainda possibilita que dados brutos de sensores sejam representados utilizando as tecnologias semânticas. Por utilizar um repositório de triplas como componente do arcabouço, questões relativas a grandes volumes de dados podem ser tratadas com as mesmas técnicas que estão sendo desenvolvidas pela comunidade científica para armazenamento de dados em RDF.

3.2 DESCRIÇÃO DE SERVIÇOS ATRAVÉS DE TECNOLOGIAS SEMÂNTICAS

O Sensor Web Enablement (SWE) (BRÖRING et al., 2009) é uma iniciativa do Open Geospatial Consortium (OGC) para definição de interfaces para serviços Web e codificação de dados para permitir a descoberta e a acessibilidade destas interfaces na Web. As especificações do SWE permitem a comunicação padronizada e interação de tipos arbitrários de sensores e sistemas de monitoramento através do uso de ontologias para definição de tipos e para a descrição das interfaces de acesso.

No trabalho de (KATASONOV et al., 2008) é apresentado um middleware chamado UBIWARE para fazer anotações sobre streams de dados, com foco na descrição dos serviços de entrega de dados e na descrição do comportamento das plataformas de sensoriamento. O UBIWARE classifica e registra vários dispositivos ubíquos e os conecta com recursos da Web, serviços e componentes de processo de negócios. Também considera sensores, redes de sensores, sistemas embarcados, alarmes, atuadores, infraestrutura da

comunicação como objetos inteligentes e os tratará como mais um recurso adicional.

No trabalho de (LI; TAYLOR, 2008) é apresentado um arcabouço baseado em tecnologias semânticas para prover consultas sobre streams de dados. Estas consultas são definidas a partir de filtros aplicados sobre os descritores de serviços, permitindo que sejam selecionados sensores a partir da escolha de localidade, tempo e do tipo de serviços que os objetos estejam provendo. Já no trabalho (GRAY et al., 2011) foi apresentado um arcabouço chamado de SemSorGrid4Env. SemSorGrid4Env explora anotações semânticas para prover interações bem informadas entre serviços heterogêneos.

A grande diferença das abordagens apresentadas acima para o SemanticHub é que elas focam na descrição dos serviços, enquanto o SemanticHub foca na descrição dos dados produzidos por estes objetos. A descrição dos serviços é adequada quando o volume de dados gerados pelos objetos é superior a capacidade de armazenamento disponível. Entretanto, não é adequado quando é necessário fazer análises mais complexas sobre os dados já produzidos.

3.3 DESCRIÇÃO DE DADOS PRODUZIDOS POR OBJETOS ATRAVÉS DE TECNOLOGIAS SEMÂNTICAS

Os trabalhos encontrados mais próximos da proposta do SemanticHub foram os de (LE-PHUOC et al., 2010) e o trabalho de (PFISTERER et al., 2011).

O trabalho de (PFISTERER et al., 2011), chamado Spitfire, tem o foco na busca de entidades que representam sensores ou entidades relacionadas. Baseado na criação automática de anotação, e nesta parte difere do SemanticHub pois a nossa abordagem foca na definição manual de entidades. O Spitfire também oferece mecanismos de ligação com outras bases de dados da nuvem de LOD. A utilização do Spitfire é baseado na utilização de entidades de alto-nível, permitindo que usuário que não tenham conhecimento do domínio ou das tecnologias associados aos sensores possam utilizar os dados produzidos através da busca eficiente. Usando princípios de linked data torna os dados dos sensores facilmente acessíveis para aplicações via mecanismos existentes implantados na Web, que aumentam significativamente a velocidade de resposta das aplicações IoT.

No trabalho de (LE-PHUOC et al., 2010) é apresentado um arcabouço de SSN. Este arcabouço possui um conjunto de quatro camadas, sendo elas: camada de aquisição de dados; camada de dados interligados; camada de acesso aos dados; e camada de aplicação.A camada de aquisição de dados é responsável por coletar os dados dos

fluxos de dados fontes. Os dados são coletados através de wrappers, um para cada tipo de interface de acesso, como portas seriais, serviços Web e interfaces assíncronas para Web. Neste aspecto, o diferencial entre este trabalho e o SemanticHub é que estes wrappers são de propósito específico, sendo necessário gerar um para cada fonte de dados. O SemanticHub, ainda que só trabalhe com serviços REST e interfaces assíncronas para Web, apresenta um mecanismo de propósito geral para estas fontes. Para a camada de dados interligados, o trabalho diz que são aplicadas transformações aos dados para convertê-los em RDF, mas não descreve como esta tarefa é realizada e as restrições, caso existam. Portanto, não é possível comparar com o arcabouço proposto neste trabalho. Para a camada de acesso aos dados foram criados dois mecanismos de consulta, LD Query Processor para dados estáticos e o CQELS Engine. Para processar os dados dos fluxos de dados dos sensores continuamente. A principal diferença entre a abordagem proposta é que no SemanticHub os dados são continuamente armazenados, não sendo perdidos caso a consulta em um fluxo tenha sido iniciado antes do tempo de interesse para análise. Como limitante, o SemanticHub é limitado pelo volume de dados que pode armazenar de acordo com a capacidade de armazenamento onde o arcabouço está sendo executado. A camada de aplicação mostra mecanismos para criação de mashups a partir dos dados estáticos armazenados e dos dados produzidos pelos fluxos de sensoriamento. Este aspecto não foi explorado no SemanticHub, pois se considera que extrapola o objetivo pretendido para este trabalho.

4 DESENVOLVIMENTO

O objetivo do SemanticHub é permitir a publicação de dados produzidos por objetos que seguem as diretivas da WoT na Web de Dados Interligados. Para que este objetivo seja atingido é necessário definir quais são as principais características de comunicação dos objetos na WoT além das características exigidas por LOD para permitir o processamento de dados de forma automatizada.

Estas características são a base para a definição de requisitos a serem atendidos por nossa solução, os quais são discutidos na seção 4.1. Na seção 4.2 apresenta as funcionalidades e componentes da arquitetura, enquanto o funcionamento do arcabouço é detalhado na seção 4.3. Finalmente, na seção 4.4, discute-se a implementação e uma avaliação realizada de cada componente.

4.1 DEFINIÇÃO DE REQUISITOS

As principais características e tecnologias utilizadas pelas propostas de WoT e LOD foram apresentadas no capítulo 2. A partir do levantamento realizado foi definido um conjunto de requisitos que nortearam o desenvolvimento do SemanticHub.

Das propostas associadas à Web das Coisas, as características mais relevantes para o desenvolvimento do SemanticHub são os métodos de comunicação dos objetos com a Web e os formatos de representação de dados utilizados pelos mesmos. Do ponto de vista dos métodos de comunicação dos objetos com a Web, existem propostas baseadas no consumo de serviços Web e em métodos de comunicação assíncrona, cada um com uma série de variações. O primeiro requisito a ser apontado para o desenvolvimento do SemanticHub é a comunicação com os objetos através dessas duas abordagens. Ainda que existam variações, o foco é utilizar o conjunto de métodos e tecnologias que estão sendo mais amplamente adotados, como a utilização de serviços do tipo REST e a comunicação assíncrona através de WebHooks, sem que isto apresente limitações à extensão do arcabouço utilizando-se abordagens diferentes.

Das propostas de LOD, a descrição de dados em RDF apresenta-se como contraponto à proposta de WoT que utilizam preferencialmente JSON e XML com schemas proprietários, ou seja, RDF pode utilizar XML ou JSON como formato de serialização, mas a estrutura dos dados está sempre na forma <sujeito><predicado><objeto> e a semântica destes dados está descrita através de ontologias e/ou vocabulários que seguem

os princípios de LOD. Permitir que os dados recebidos dos objetos e descritos através de schemas proprietários possam seguir os princípios de LOD é o segundo requisito levantado para o SemanticHub.

O último requisito apontado é a possibilidade de realizar navegação entre os dados ou consultas sobre as bases. Para a realização das consultas existe a proposta da linguagem Sparql, que é adotada pela maior parte das bases da nuvem de LOD, e, portanto, escolhida para o arcabouço. A navegação, por sua vez, ocorre da mesma forma que na Web de documentos, mas ao invés de ser acessado uma página HTML (ou equivalente) para cada URI, que neste caso também é uma URL, são acessados as propriedades de dados e de objetos que aquela URI possui.

Em resumo, os requisitos do arcabouço proposto são:

1. Permitir a comunicação com objetos através de serviços Web e de métodos assíncronos de comunicação para Web
2. Permitir que os dados produzidos por objetos em XML e JSON, com schemas proprietários, sejam representados segundo os princípios de LOD
3. Permitir a navegação e consulta sobre os dados obtidos

4.2 DESCRIÇÃO DE FUNCIONALIDADES DO ARCABOUÇO SEMANTICHUB

Com base nos requisitos definidos na seção 4.1, o SemanticHub foi criado para ser um mediador entre os objetos integrados na Web e a Web de Dados Interligados. O funcionamento do arcabouço pode ser dividido em duas partes. Na primeira parte, referente a esta seção, será feita a descrição funcional do SemanticHub, na seção 4.3 será feita a descrição por componentes.

O funcionamento do SemanticHub ocorre em duas macro-etapas: a macro-etapa de conFiguração e a macro-etapa de execução. Na macro-etapa de conFiguração um usuário com privilégios especiais define os parâmetros de execução, as fontes de dados utilizadas e como os dados serão transformados para RDF. A segunda macro-etapa representa o funcionamento do arcabouço em si, descrevendo assim, o fluxo de execução dos componentes de software. Nas próximas subseções será feito o detalhamento das macro-etapas.

4.2.1 Macro-Etapa de ConFiguração do SemanticHub

A macro-etapa de conFiguração do SemanticHub é composta de nove atividades, representadas no diagrama de atividades da Figura 8. Essas atividades têm dois objetivos principais: definir a forma de acesso do SemanticHub com os objetos da WoT; e definir como os dados recebidos serão transformados para RDF.

Todo o processo de conFiguração do SemanticHub é feito através da interação de um usuário com privilégios especiais, denominado Usuário ConFigurador. Este usuário define os métodos de acesso aos objetos, o intervalo de tempo em que as interfaces síncronas devem ser atualizadas e o intervalo mínimo entre recebimento de dados das interfaces assíncronas. Outra função do Usuário ConFigurador é a definição das transformações de dados que serão aplicadas nos dados recebidos dos objetos, para que estes sigam as definições das ontologias e/ou vocabulários do domínio que serão representadas pelo SemanticHub. As ontologias e vocabulários que estão sendo utilizados devem estar armazenados no repositório de triplas.

Durante o processo de construção das transformações de dados, o SemanticHub provê uma interface que permite a visualização de tipos e propriedades representados através das ontologias e vocabulários presentes no repositório de triplas.

A etapa de conFiguração das fontes de dados consiste no cadastro dos serviços Web que serão consumidos durante a etapa de execução. O SemanticHub se limita a utilizar WebHooks e Webservices REST para comunicação com as coisas do mundo real por serem técnicas adotadas pela WoT para conectar dispositivos na Web. Também assume-se que os dados recebidos através destas interfaces estarão nos formatos semi-estruturados XML ou JSON, já que estes formatos são os mais utilizados para representação de dados na Web (LANTHALER; GUTL, 2011).

A comunicação do SemanticHub com os objetos da WoT pode ser feita de duas maneiras: síncrona através do consumo periódico de serviços REST; e assíncrona através de WebHooks. A comunicação síncrona ocorre quando algum objeto do mundo real provê algum serviço REST para acesso a seus dados. Como não é possível saber, a priori, quando novos dados estão disponíveis, é necessário consumir o serviço em intervalos periódicos de tempo. Neste caso, durante o cadastro do serviço no arcabouço é necessário definir além da URL e parâmetros do serviço REST, o intervalo de tempo em que este serviço deverá ser consumido. Como limitação devido às estratégias adotadas para o funcionamento do arcabouço, não é possível consumir serviços REST em intervalos inferiores a um segundos.

Para a comunicação assíncrona através de WebHooks (TRIFA et al., 2010) o arcabouço passa a prover um serviço REST. Este serviço é cadastrado nas fontes de dados que implementam as definições de WebHooks. Quando a fonte provedora dos dados define que é necessário publicar seus dados, ela faz uso deste serviço como um canal de comunicação.

Para identificar qual fonte de dados fez a transmissão, a URL do serviço REST provida pelo SemanticHub possui um parâmetro de identificação, sendo assim, temos uma URI para cada fonte de dados. Na Figura 9 é apresentado um diagrama mostrando a sequencia de passos que ocorre nas duas formas de comunicação.

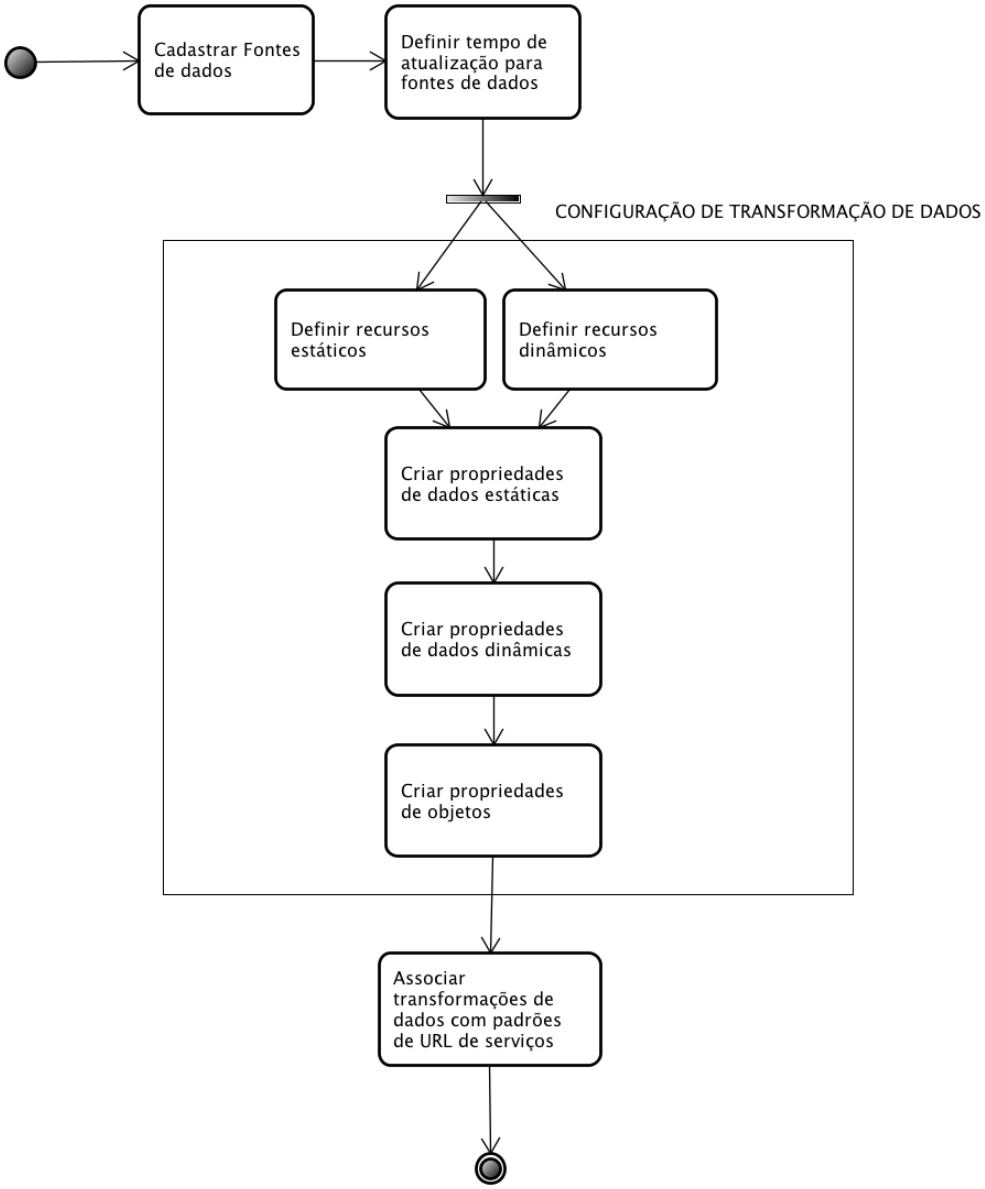


Figura 8 – Diagrama de atividades representando a macro-etapa de conFiguração do SemanticHub

Em ambas as formas de comunicação (síncrona e assíncrona), é necessário definir o formato de dados disponibilizados (XML ou JSON) para que sejam aplicadas as transformações necessárias no momento da extração e transformação dos dados, como será explicado na próxima subseção.

Os dados providos pelos objetos devem ser transformados para RDF seguindo as definições de uma ou mais ontologias e/ou vocabulários. Nesta subseção serão apresentados os mecanismos de conFiguração necessários para a geração de recursos.

O SemanticHub pode gerar dois tipos distintos de recursos: recursos estáticos e recursos dinâmicos. Recursos estáticos não se alteram ao longo do tempo, eles existem para estender a expressividade dos dados providos pelas fontes de dados e são totalmente independentes destas fontes. Ou seja, recursos estáticos existem desde o início de execução do arcabouço. Para criar um recurso estático é necessário definir uma URI para o recurso e o seu rdf:type seguindo os vocabulários/ontologias que estão sendo empregados.

Para auxiliar na criação dos recursos (neste caso tanto estáticos quanto dinâmicos) na interface de criação dos recursos são listados para seleção todas as classes armazenadas no repositório de triplas. Pode-se ver um exemplo de regra de mapeamento na Tabela 5 para um recurso além do dado em RDF produzido a partir deste mapeamento.

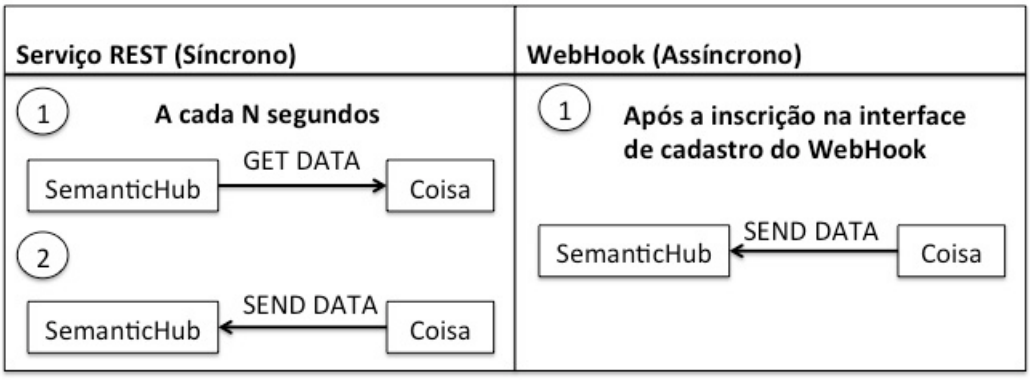


Figura 9 – Formas de comunicação com o SemanticHub

Recursos dinâmicos devem sua existência à chegada de dados das fontes de dados cadastradas previamente. Toda vez que novos dados chegam ao SemanticHub, de acordo com as regras de mapeamentos novos recursos são criados. Uma série de mapeamentos podem ser associadas a cada chegada de dados e, por sua vez, cada bloco de informação dentro dos pacotes de dados recebidos pode ser mapeado para um ou mais recursos. Uma vez que um recurso dinâmico é criado ele não é mais alterado, exceto no caso de ser um recurso já existente que deverá ser removido através do mecanismos de Entity Reconciliation que será apresentado posteriormente.

Ao contrário dos recursos estáticos que possuem na sua regra de mapeamento uma URI definida, recursos dinâmicos possuem apenas uma URI base, que será concatenada com um número representando o momento da criação do recurso para que possa ser tornar uma URI. Assim como um recurso estático, também é necessário definir o rdf:type associado a cada recurso dinâmico. Na Tabela 3 é possível ver o conjunto de mapeamentos associados a um serviço REST além do resultado deste mapeamento.

Uma vez que os recursos tenham sido criados, é esperado que uma série de propriedades de dados possa ser associada a estes recursos. Assim como os recursos podem ser de dois tipos, as propriedades de dados no SemanticHub podem ser de três tipos: estáticas, dinâmicas e virtuais. Propriedades de dados estáticas e dinâmicas podem ser associadas tanto a recursos estáticos quanto a recursos dinâmicos. Propriedades virtuais podem ser associadas apenas a recursos estáticos.

Propriedades de dados estáticas existem para prover informações adicionais para as informações providas pelos serviços. Por exemplo, caso seja necessário informar qual a plataforma de hardware um determinado dispositivo foi construído basta informar qual o recurso ao qual esta informação está associada e qual a propriedade que irá relacionar o recurso com a propriedade de dado. Se a propriedade de dado estiver relacionada a um recurso estático ela será criada apenas uma vez. Se ela estiver associada ao mapeamento de um recurso dinâmico, toda vez que o recurso dinâmico for criado, a este recurso uma nova propriedade de dados será associada a este recurso. Na Tabela 4 é possível ver um exemplo para propriedades de dados estáticas e dinâmicas.

Tabela 5 – Exemplo de conFiguração de um recurso dinâmico e de um recurso estático

Regra de Mapeamento	**XML**	**Resultado da Transformação**
TYPE: Recurso Estático **URI:** http://teste.com#sensor **RDF-TYPE:** http://teste.com/ontology#Sensor		<http://teste.com#sensor> rdf:type <http://teste.com/ontology#Sensor>
TYPE: Recurso Dinâmico **URI:** http://teste.com#data **RDF-TYPE:** http://teste.com/ontology#value **Fonte do Serviço:** http://example.com/sensor/1 **Grupo de Dados:** //data	<root> <data type="temp" <value>20</value> <unit>celsius</unit> <date>08/11/2011</date> <time>02:28:59</time> </data> <data type="temp" <value>21</value> <unit>celsius</unit> <date>08/11/2011</date> <time>02:29:36</time> </data> </root>	<http://teste.com#data123> rdf:type <http://teste.com/ontology#value> <http://teste.com#data456> rdf:type <http://teste.com/ontology#value>

Propriedades de dados dinâmicas são resultado de extrações de dados, XPath para dados em XML e Javascript para dados em JSON, sobre os dados recebidos das

fontes de dados. Quando elas estão associadas a um recurso estático os valores daquela propriedade serão atualizados toda vez que novos dados chegarem. Quando propriedades dinâmicas estão associadas a recursos dinâmicos esta propriedades serão criadas em conjunto com aquele recurso e não serão mais modificadas. Na Tabela 5 é apresentado um exemplo de propriedade dinâmica associada a um recurso estático e a recursos dinâmicos.

Como está sendo feito o consumo de serviços Web REST em intervalos periódicos de tempo, é esperado que recursos dinâmicos repetidos possam ser gerados. Isto ocorre quando um serviço é consumido mais de uma vez caso novos dados tenham sido gerados pelo objeto. Para corrigir este problema, o arcabouço apresenta um mecanismo que constantemente varre a base de dados procurando recursos com URIs distintas que possuam o mesmo conjunto de propriedade de dados e propriedades de objetos. Quando encontrados recursos repetidos com URIs distintas a ação padrão é manter a URI com timestamp mais antigo, já que ligações em bases de dados externas podem ter sido criadas para esta URI, e apagar os outros recursos redundantes e seus dados. Como o tempo de inconsistência é pequeno (na ordem de segundos), esta estratégia mostrou-se eficiente e eficaz nos testes realizados, conforme será visto mais adiante.

Tabela 6 – Exemplo de conFiguração de propriedades estáticas para recursos estáticos e dinâmicos

Regra de Mapeamento	**XML**	**Resultado da Transformação**
Recurso Estático		
TYPE: Recurso Estático **URI:** http://teste.com#sensor **RDF-TYPE:** http://teste.com/ontology#Sensor		<http://teste.com#sensor> rdf:type <http://teste.com/ontology#Sensor>
Recurso Dinâmico		
TYPE: Recurso Dinâmico **URI:** http://teste.com#data **RDF-TYPE:** http://teste.com/ontology#value **Fonte do Serviço:** http://example.com/sensor/1 **Grupo de Dados:** //data	<root> <data type="temp"> <value>20</value> <unit>celsius</unit> <date>08/11/2011</date> <time>02:28:59</time> </data> <data type="temp"> <value>21</value> <unit>celsius</unit> <date>08/11/2011</date> <time>02:29:36</time> </data> </root>	<http://teste.com#data123> rdf:type <http://teste.com/ontology#value> <http://teste.com#data456> rdf:type <http://teste.com/ontology#value>

Propriedades de dados virtuais são propriedades que são atualizadas em intervalos predefinidos de tempo e são geradas através da combinação de outras informações presentes. Sua função é prover um mecanismo, ainda que simples, de fusão de dados. Essas propriedades são associadas a operações sobre o resultado de consultas. A implementação desta característica é feita com a execução de uma ou mais consultas sparql em intervalo predefinido de tempo. O resultado destas consultas podem ser associados a um código

Javascript, permitindo que fórmulas matemáticas, por exemplo, possam ser aplicadas ao resultado da consulta. O resultado final é então transformado em RDF. Pode-se utilizar o resultado deste código diretamente ou associar faixas de valores específicas a um determinado valor de propriedade, assim como representado na Tabela 6.

Terminado o mapeamento para criação de recursos e propriedade de dados, é necessário descrever como são criadas as ligações entre recursos, ou seja, como são criados as propriedade de objetos no arcabouço proposto.

Recursos estáticos podem fazer referência a quaisquer outros recursos (tanto internos quanto de outras bases de dados), bastando para isso informar qual a propriedade que será utilizada nesta relação e a URI do outro recurso.

Tabela 7 – Exemplo de conFiguração de propriedades dinâmicas para recursos estáticos e dinâmicos

Regra de Mapeamento	**XML**	**Resultado da Transformação**
Recurso Estático		

TYPE:
Propriedade de Dados Dinâmica
RESOURCE:
http://teste.com#sensor
PROPERTY:
http://teste.com/ontology#lastUpdate
Fonte do Serviço:
http://example.com/sensor/1
Grupo de Dados:
/
Transformação:
//concat(//date[last()]/text(),",
//time[last()]/text())
	<root>
	<data type="temp">
	<value>20</value>
	<unit>Celsius</unit>
	<date>08/11/2011</date>
	<time>02:28:59</time>
	</data>
	<data type="temp">
	<value>21</value>
	<unit>Celsius</unit>
	<date>08/11/2011</date>
	<time>02:29:36</time>
	</data>
	</root>
		<http://teste.com#sensor>
		<http://teste.com/ontology#lastUpdate>
		"08/11/2011 02:29:36"

Recurso Dinâmico		

TYPE:
Propriedade de Dados Dinâmica
RESOURCE:
http://teste.com#data
PROPERTY:
http://teste.com/ontology#hasValue
Fonte do Serviço:
http://example.com/sensor/1
Grupo de Dados:
//data
Transformação:
//value/text()
	<root>
	<data type="temp">
	<value>20</value>
	<unit>Celsius</unit>
	<date>08/11/2011</date>
	<time>02:28:59</time>
	</data>
	<data type="temp">
	<value>21</value>
	<unit>Celsius</unit>
	<date>08/11/2011</date>
	<time>02:29:36</time>
	</data>
	</root>
		<http://teste.com#data123>
		<http://teste.com/ontology#hasValue>
		"20"
		
		<http://teste.com#data456>
		<http://teste.com/ontology#hasValue>
		"21"

Recursos dinâmicos, dentro do arcabouço proposto, não podem ser sujeitos, apenas objetos nas relações e necessariamente devem estar associados a recursos estáticos ou recursos de bases de dados externas. No mapeamento de recursos dinâmicos é necessário

Tabela 8 – Exemplo de conFiguração de propriedades uma propriedade virtual

Regra de Mapeamento	Resultado da Transformação
TYPE: Recurso Virtual **URI:** http://teste.com#sensor **UPDATE TIME:** 10min **Property:** http://teste.com/ontology#tempAVG **Query:** SELECT (AVG(?y) AS ?result) WHERE { ?a rdf:type example:value. ?a example:hasValue ?y. } **Intervalos:** result < 20 : cold 20 <= result >= 30 : confortable result > 30 : warm	<http://teste.com#sensor> rdf:type <http://teste.com/ontology#tempAVG> "confortable"

informar o recurso estático que fará o papel de sujeito na relação e a propriedade a ser utilizada. Na Tabela 8 é apresentado um exemplo da ligação entre esses recursos.

Tabela 9 – Exemplo de conFiguração de propriedades interligando dois objetos

Regra de Mapeamento	Resultado da Transformação
SOURCE: http://teste.com#sensor **TARGET:** http://teste.com#data **PROPERTY:** http://teste.com/ontology#hasSensingData	<http://teste.com#sensor> <http://teste.com/ontology#hasSensingData> <http://teste.com#data123> <http://teste.com#sensor> <http://teste.com/ontology#hasSensingData> <http://teste.com#data456>

4.2.2 Macro-Etapa de Execução do SemanticHub

O fluxo de execução dos componentes do SemanticHub pode ser visto no diagrama de atividades da Figura 10. Detalhes sobre a implementação e capacidade de cada componente serão discutidos na próxima seção. O fluxo básico pode ser dividido em quatro etapas, sendo que a primeira etapa (criação de recursos estáticos) é feita uma única vez, no princípio da execução do arcabouço. As demais atividades são executadas periodicamente, de acordo com o momento em que os dados dos objetos são atualizados.

A captura de dados, como descrito na subseção anterior, é feita de duas formas. A primeira síncrona, na qual o arcabouço verifica periodicamente o último instante de recebimento de dados de todos os serviços REST que devem ser consumidos. Quando esse intervalo está igual ou superior ao tempo pré-estabelecido na macro-etapa de conFiguração, uma nova chamada aquele serviço é feita. A interface assíncrona é passiva, fazendo apenas a verificação do intervalo mínimo aceitável para a atualização do dado.

Uma vez que um dado tenha sido recebido de um serviço, na terceira etapa, as

transformações associadas a cada padrão de URL na chamada de serviço serão aplicadas. Estes padrões são expressões regulares que procuram fazer o casamento com as URLs dos serviços. Uma vez que tenham sido geradas as triplas RDF, é feita a verificação da validade destas triplas e posterior armazenamento no repositório de triplas, sendo que triplas defeituosas são removidas e logs são gerados. Ainda nessa etapa, caso existam inconsistências na base, como URIs distintas representando o mesmo recurso, será aplicado o mecanismo de reconciliação de entidades para realizar as correções necessárias para manter a unicidade dos recursos no repositório de triplas.

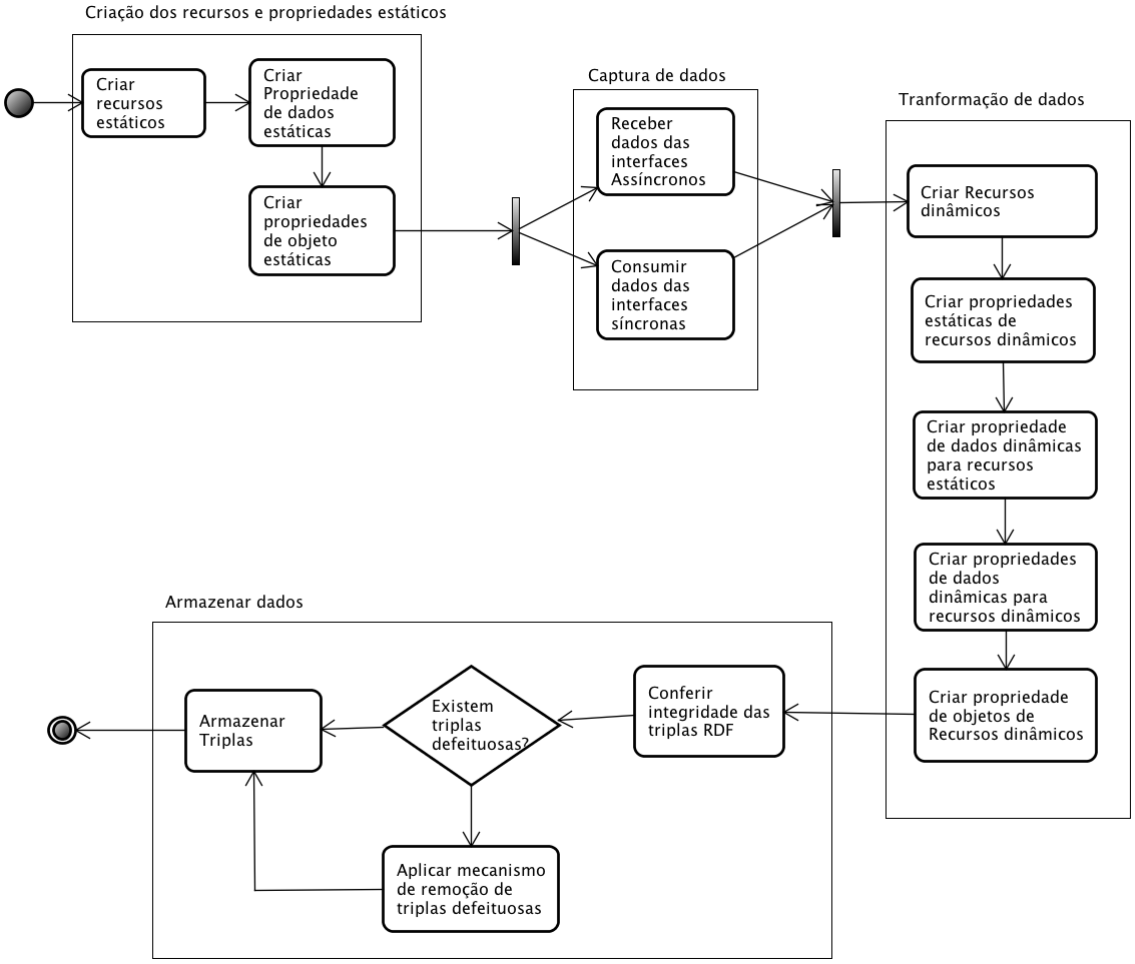


Figura 10 – Diagrama de atividade descrevendo a macro-etapa de execução do SemanticHub

4.3 ARQUITETURA

Nesta seção apresenta-se a descrição estrutural do SemanticHub. Na Figura 11 um diagrama de componentes representa a arquitetura proposta. A arquitetura possui sete componentes principais, sendo a função de cada componente sumarizada abaixo.

- Componente de Captura de Dados: Realiza a comunicação com os objetos na Internet. Possui dois sub-componentes, um para captura assíncrona de dados e outro para captura síncrona.
- Componente de Transformação de Dados: É o componente responsável por aplicar transformações pré-definidas ao dados que são recebidos pelo SemanticHub. Possui um sub-componente para dados estáticos (que não dependem diretamente dos dados recebidos dos objetos) e outro sub-componente para transformações dinâmicas (que executam operações de transformação nos dados recebidos pelos objetos)
- Componente de Alimentação de Dados em RDF: Responsável pela inserção dos dados transformados em RDF no repositório de triplas. Também efetua a verificação de consistências nos dados gerados além de atuar como um mecanismos de armazenamento temporário(cache)
- Componente de Reconciliação de Entidades: Responsável por manter a corretude do repositório de triplas, uma vez que dados repetidos podem ser inseridos com URIs distintas devido ao consumo síncrono de serviços e a determinados tipos de transformações de dados.
- Gerenciador de conFiguração: Tem como função coordenar o funcionamento dos demais componentes além de ser a interface de acesso para os usuários que são responsáveis pela conFiguração.
- Sparql-Endpoint: Interface para execução de consultas Sparql nos dados armazenados como triplas RDF.
- Navegador RDF: Interface para navegação entre triplas RDF.

4.3.1 Componente de Captura de Dados

O componente de captura de dados possui duas funcionalidades principais, consultar em intervalos periódicos se dados de objetos devem ser obtidos através do consumo de serviços Web destes objetos e receber dados dos objetos que possuem interfaces do tipo WebHook onde o SemanticHub foi previamente cadastrado para receber dados.

O consumo periódico é função do sub-componente de captura síncrona. Através do um ciclo infinito que verifica constantemente o último instante em que cada um dos serviços REST foi consumido e o intervalo conFigurado para atualização. Quando é detectado que um serviço deve ser atualizado, o sub-componente de captura síncrona faz

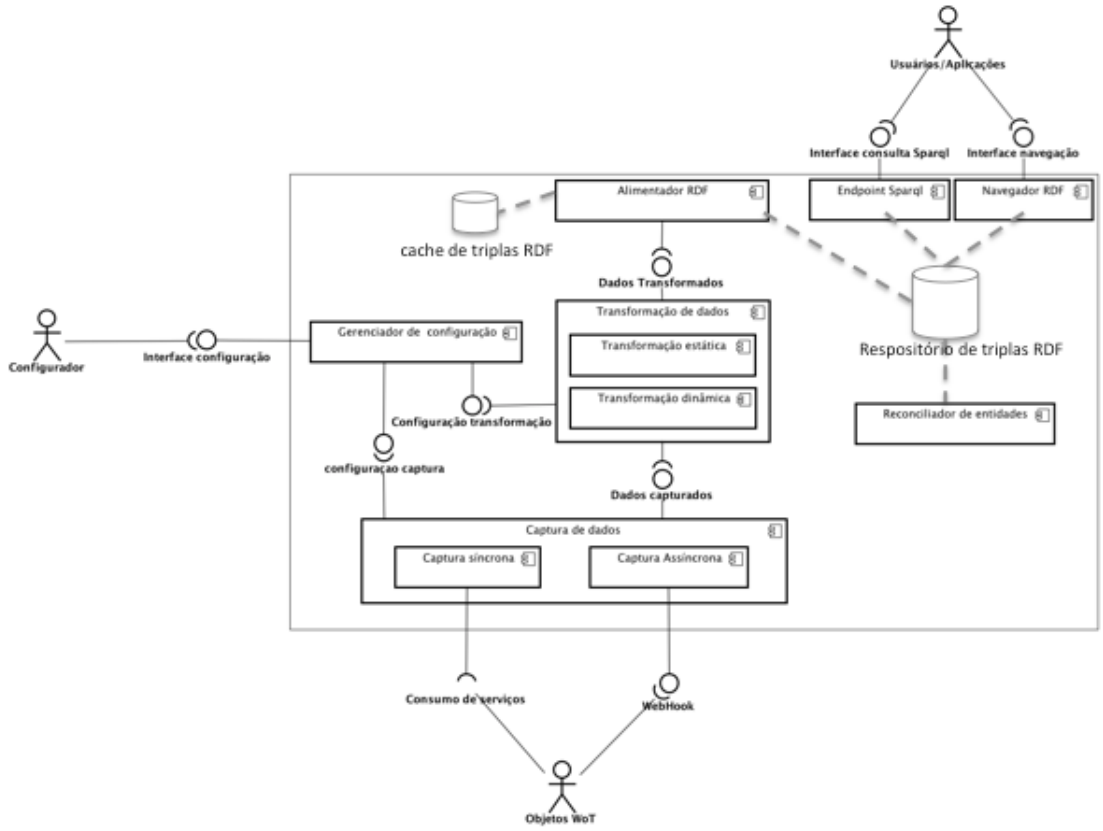


Figura 11 – Diagrama de Componentes representando a arquitetura do SemanticHub

a requisição para o objeto. O sub-compoente de captura assíncrona é o responsável por implementar os WebHooks e receber os dados dos objetos que utilizam esta técnica.

Uma vez tendo recebido os dados, estes serão repassados para o próximo componente, o componente de transformação de dados. Além do dado, informações adicionais são enviadas como a URL do serviço que proveu o dado. A URL é importante pois ela tem o papel de ser o identificador da origem do dados, sendo utilizada pelo componente seguinte para definição das ações que devem ser aplicadas. Outras informações enviadas são o instante de tempo e o formato do dado em que o dado foi recebido.

4.3.2 Componente de Transformação de Dados

Suas principais funcionalidades são aplicar as transformações previamente cadastradas para os dados produzidos pelas fontes de dados correlatas (sub-componente de transformação dinâmica) e criar novos dados, definidos pelo usuário, para estender a capacidade de representação do domínio (sub-componente de transformação estática).

Como descrito na seção de conFiguração, cada entrada de dado, identificada

pela URL do serviço, pode ser associada a uma ou várias transformações. Cada transformação por sua vez, pode ser associada a uma ou mais padrões de URL, onde este padrão é definido através de um expressão regular. Uma transformação pode também ser aplicada várias vezes dentro de uma mesma entrada de dados proveniente de um serviço. Como os dados em XML e JSON geralmente apresentam estruturas repetidas, é possível dizer que a transformação deve ser aplicada a padrões de repetição dentro de um mesmo pacote de dado.

As transformações são trechos de código que são aplicados aos dados recebidos dos serviços com o objetivo de transforma-los em RDF. Essas transformações representam o conjunto de operações necessárias para, a partir de um conjunto de dados estruturados em XML ou JSON, fazer com que estes dados sigam as definições de ontologias e/ou vocabulários. Como este conjunto de operações são definidos por pessoas, estas são as responsáveis pela corretude dos dados gerados. Visando auxiliar a criação destas transformações, o componente de Transformação oferece um serviço de busca nas classes e propriedades que estão armazenadas no repositório de triplas do SemanticHub. Uma vez que os dados estejam no formato RDF, eles são encaminhados para o componente de Alimentação de Dados RDF.

4.3.3 Componente de Alimentação de Dados RDF

O componente de Alimentação de Dados RDF recebe os dados provenientes da etapa de transformação. Sua função é ser a interface com o repositório de triplas, permitindo contornar os problemas relacionados a limitações para armazenamento de fluxos de dados dinâmicos no repositório. Para isso este componente implementa mecanismos de inserção, armazenamento temporário e de correção de erros.

Com o objetivo de otimizar o processo de inserção de dados o componente de Alimentação de Dados RDF insere blocos de triplas (na ordem de 100 triplas por bloco) ao invés de fazer inserções unitárias (tripla a tripla). Como o fluxo de dados pode gerar dados com baixa frequencia, caso um bloco não atinja o número de triplas necessárias para ser armazenado em um tempo pré-determinado (menor que um segundo), o componente faz o armazenamento, garantindo que a base seja periodicamente atualizada. Caso contrário, este mecanismo possibilita que um volume grande de triplas, na ordem de milhares por segundo, sejam inseridas.

Quando ocorre um erro ao tentar se inserir um bloco de triplas no repositório o componente de Alimentação de Dados RDF pode tomar duas ações, dependentes do tipo de erro. Quando o repositório está trabalhando na capacidade máxima, o componente atua

como um *buffer*, aumentando o tempo entre inserções, armazenando temporariamente os dados para que estes sejam posteriormente inseridos. A capacidade de dados que pode ser bufferizado está diretamente associada com o hardware disponível. Quando ocorrem erros devido a problemas no formato de uma ou mais triplas em um bloco de inserção, o componente começa um processo de inserção baseado em busca binária. Este mecanismo foi adotado visando reduzir o tempo total de inserção. Inicialmente o componente tenta dividir o bloco em dois, para realizar a inserção. Caso não seja suficiente, ele reparte novamente cada novo bloco até que o erro seja encontrado. As triplas que não forem inseridas são então armazenadas em um arquivo de log, permitindo a posterior análise. Durante o desenvolvimento do SemanticHub foi percebido que esses erros, de forma geral, estavam relacionados ou a problemas de codificação de caracteres ou a transformações de dados incorretas.

4.3.4 Componente de Reconciliação de Entidades

O componente de Reconciliação de Entidades executa a atividade Reconciliar Entidades do diagrama de atividades apresentado na Figura 7. Sua função é manter a unicidade dos recursos no repositório de triplas. Problemas de unicidade possuem duas causas principais. A primeira relativa ao consumo de serviços de forma periódica, a segunda devido a transformações inconsistentes.

O consumo síncrono de serviços tem como problema principal o fato do cliente que consome os serviços não saber se os dados foram atualizados. Portanto, se o intervalo de atualização for inferior ao tempo que o objeto gera novos dados, parte dos dados recebidos serão n-plicados. Já as transformações podem fazer recorte no dados recebidos, fazendo com que trechos repetidos em XML ou JSON sejam transformados em RDF, dando origem a conjuntos de triplas repetidos com identificadores distintos.

4.3.5 Componente de Gerenciamento de ConFiguração

O Gerenciamento de ConFiguração abrange todas as etapas do diagrama de atividades da Figura 6. Ele também é o responsável pela coordenação dos demais componentes do SemanticHub. O componente de gerenciamento provê as interfaces para cadastro de fontes de dados, cadastro de transformações e demais parâmetros de conFiguração.

O cadastro de serviços pode ser feito adicionando serviço a serviço, utilizando descritores WADL [1] ou com um serviço de descrição que provê as demais interfaces que

[1] WADL (Web Application Description Language) é um formato em XML para descrever aplicações Web baseadas em HTTP (geralmente serviços Web do tipo REST)

podem ser utilizadas. Os intervalos de atualização pode ser adicionado para cada entrada ou para grupo de entradas, permitindo assim que essa atividade possa ser feita em tempo hábil para um grande número de objetos.

O cadastro de transformações inclui ações detalhadas na subseção 4.2.1. Uma vez construídas as transformações, estas podem ser associadas a uma URL específica ou a uma expressão regular, permitindo que uma mesma transformação possa ser aplicada para um grande número de serviços que compartilham semelhanças entre suas URLs, o que é típico de grupos de objetos que publicam dados com estrutura semelhante. Esta característica permite o rápido setup para o início do funcionamento do SemanticHub.

4.3.6 Componente Sparql Endpoint e Componente Navegador RDF

Os componentes de Sparql Endpoint e Navegador RDF são as interfaces dos clientes e/ou aplicações que irão consumir os dados armazenados no repositório de triplas, provendo respectivamente uma interface para realização de consultas Sparql e uma interface para realização de navegação através das propriedade de objetos. A implementação de ambos componentes foi feita através da utilização das interfaces providas pelo repositório de triplas, sendo apenas uma camada sobre as interfaces de acesso existentes.

4.4 IMPLEMENTAÇÃO E AVALIAÇÃO

Nesta seção apresenta-se uma implementação do arcabouço proposto, mostrando a capacidade de cada componente, suas limitações e o funcionamento do SemanticHub. Para formalizar o processo de avaliação será utilizada a técnica GQM (Goal Question Metric) (CALDIERA; ROMBACH, 1994). Nesta técnica são definidos os objetivos (G) a serem avaliados , as questões (Q) que são feitas para verificar se o objetivo foi atingido e as métricas utilizadas para responder estas questões (M).

A implementação de todos os componentes foi paralelizada a nível de threads visando aumentar ao máximo a capacidade de execução do arcabouço como um todo. Durante o desenvolvimento do arcabouço, e de acordo com as características de hardware e software que serão descritas abaixo, o número de threads por componente que otimizam o funcionamento foram: quatro threads para o componente de captura de dados; dez threads para o componente de Transformação; quatro threads para o componente de Alimentação de Dados RDF. O componente de Reconciliação de Entidades não foi paralelizado, pois a estratégia adotada não permitiria o sincronismo entre as threads. O componente de Gerenciamento de ConFiguração, dadas as características de sua função,

não necessitou ser paralelizado.

A comunicação entre componentes foi feita através de filas, uma vez que uma das threads do componente de Captura recebe um dado, este é adicionado a uma fila, as threads do componente de Transformação consomem esta fila, adicionando em outra fila os dados em RDF transformados, o componente de Alimentação de Dados RDF consome a fila com blocos de dados em RDF, armazenando-os no repositório seguindo sua especificação.

4.4.1 Ambiente de Desenvolvimento e ConFiguração dos Testes

Como a capacidade do arcabouço depende diretamente da capacidade do hardware onde este é executado, é importante descrever onde os testes foram realizados. Essas informações são apresentadas na Tabela 11. Além disso, existem parâmetros de software relativos ao sistema operacional utilizado e a linguagem e plataforma em que o arcabouço foi implementado. Por último, o SemanticHub tem como objetivo principal manter um repositório de triplas atualizado com dados provenientes de dispositivos na WoT. Também é definido o repositório de triplas que será utilizado e as conFigurações deste repositório. Todos os parâmetros de software podem ser observados na Tabela 10.

Tabela 10 – ConFiguração de Hardware onde os testes foram realizados

Recurso de Hardware	**Descrição**
Memória	16GB DDR-3 1066MHz
Processador	Xeon 2.8 GHz com quatro núcleos
Armazenamento	HD SATA, 5400 rpm

Tabela 11 – ConFiguração de Software onde os testes foram realizados

Recurso de Software	**Descrição**
Sistema Operacional	Ubuntu 10.04 64 bits
Linguagem de Programação	Java 6, Oracle Edition
Parâmetros da JVM	(heap máxima) -Xmx = 8g (heap mínima) -Xms = 4g
Repositório de Triplas	Sesame 2.6.10

4.4.2 Avaliação do Componente de Captura de Dados

O componente de Captura corresponde à interface de comunicação do arcabouço com os dispositivos da IoT. O objetivo deste componente é capturar pacotes de dados providos pelos dispositivos e então inseri-los no fluxo de processamento do SemanticHub. Para esse componente foram avaliados a capacidade de recebimento de pacotes e o consumo de recursos computacionais para diferentes cargas de recebimento.

Para realizar este teste construiu-se uma aplicação com o propósito específico de gerar um grande número de pacotes de dados em XML e JSON com tamanho específico para as duas interfaces do SemanticHub (síncrona e assíncrona). Não foram avaliadas questões relativas à capacidade da conexão de rede disponível e todos os testes foram executados no dispositivo de loopback do computador de testes. Na Tabela 13 é apresentada a descrição em GQM para o experimento realizado.

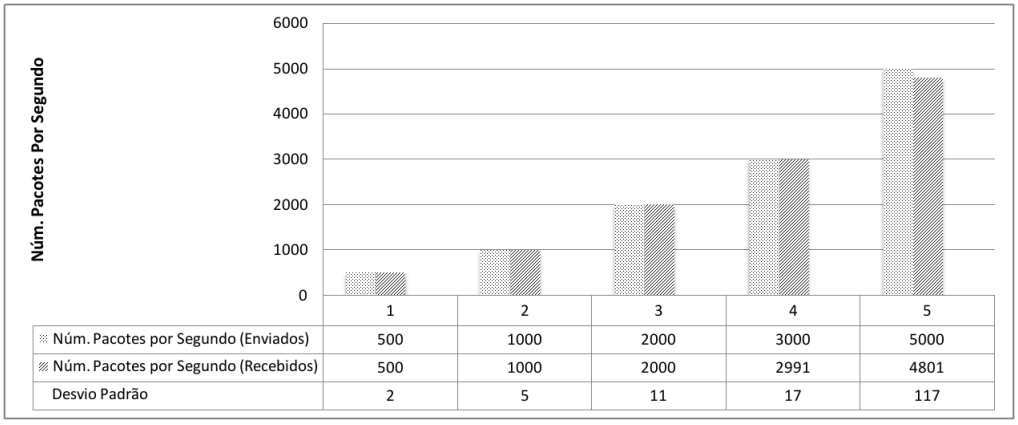
Tabela 12 – GQM para avaliação do componente de captura de dados

Objetivo Propósito Problema Objeto	 Analisar a capacidade de recebimento de pacotes de dados
Questão **Métricas**	Quantos pacotes de dados podem ser recebidos por unidade de tempo? -Número de pacotes recebidos por segundo -Desvio padrão
Questão **Métricas**	Qual o consumo de recursos computacionais em relação ao recebimento de pacotes? -Memória consumida -Percentual de processamento consumido -Desvio padrão

Para responde a questão 1 da Tabela 12, a aplicação de testes gerou pacotes de tamanho fixo de 10KB (nesta fase o formato dos dados não interfere nos testes), em cinco taxas pré-determinadas: 500 pacotes por segundo; 1000 pacotes por segundo; 2000 pacotes por segundo; 3000 pacotes por segundo; e 5000 pacotes por segundo. Os testes foram executados durante 30 minutos para cada cenário. Os resultados apresentados correspondem às médias de pacotes enviados/recebidos no intervalo do teste.

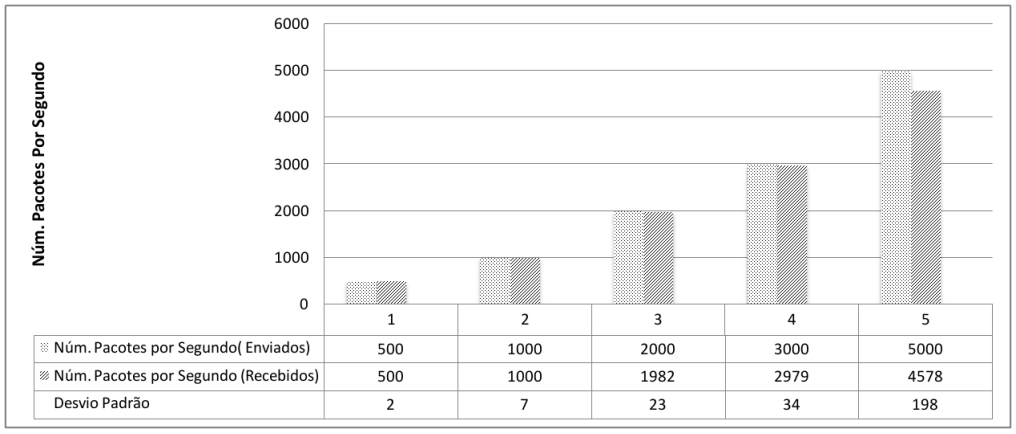
Os resultados dos testes para interface síncrona podem ser visto na Figura 12. Na Figura 13 são apresentados os resultados para comunicação assíncrona. A coluna pontilhada apresenta o número de pacotes por segundo enviados através do programa de teste. A outra coluna apresenta o número de pacotes que o SemanticHub foi capaz de capturar.

Testes com taxas maiores que 5000 pacotes por segundo não foram possíveis, pois a memória disponível foi completamente consumida. Como a avaliação realizada tem como objetivo mensurar a capacidade de recebimento de dados, não foi considerado que os outros componentes podem interferir com os resultados obtidos, reduzindo assim, o número de pacotes recebidos por unidade de tempo.



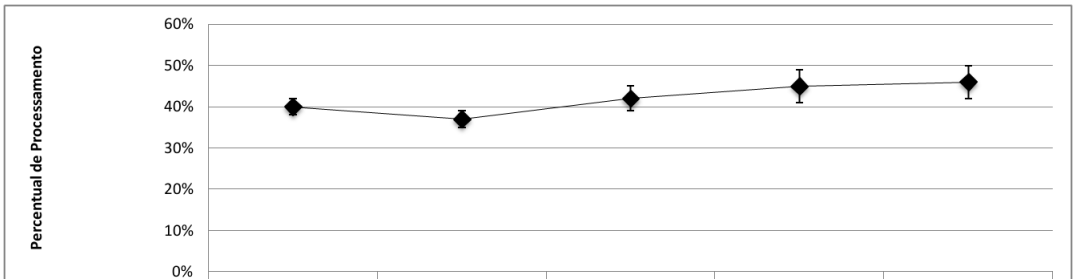
	1	2	3	4	5
Núm. Pacotes por Segundo (Enviados)	500	1000	2000	3000	5000
Núm. Pacotes por Segundo (Recebidos)	500	1000	2000	2991	4801
Desvio Padrão	2	5	11	17	117

Figura 12 – Número de pacotes recebidos por segundo (síncrono)



	1	2	3	4	5
Núm. Pacotes por Segundo(Enviados)	500	1000	2000	3000	5000
Núm. Pacotes por Segundo (Recebidos)	500	1000	1982	2979	4578
Desvio Padrão	2	7	23	34	198

Figura 13 – Número de pacotes recebidos por segundo (assíncrono)



	1	2	3	4	5
Processamento (%)	40%	37%	42%	45%	46%
Desvio Padrão	2%	2%	3%	4%	4%

Figura 14 – Percentual de processamento consumido pelo componente de captura de dados (síncrono e assíncrono)

4.4.3 Avaliação do Componente de Transformação de Dados

Como visto anteriormente, o componente de Transformação de dados tem como função aplicar uma série de extrações e conversões aos dados dos pacotes obtidos pelo SemanticHub, convertendo estes dados para o formato RDF. Essas transformações, como dito na subseção 4.3.1, podem ser transformações XPath ou scripts Javascript.

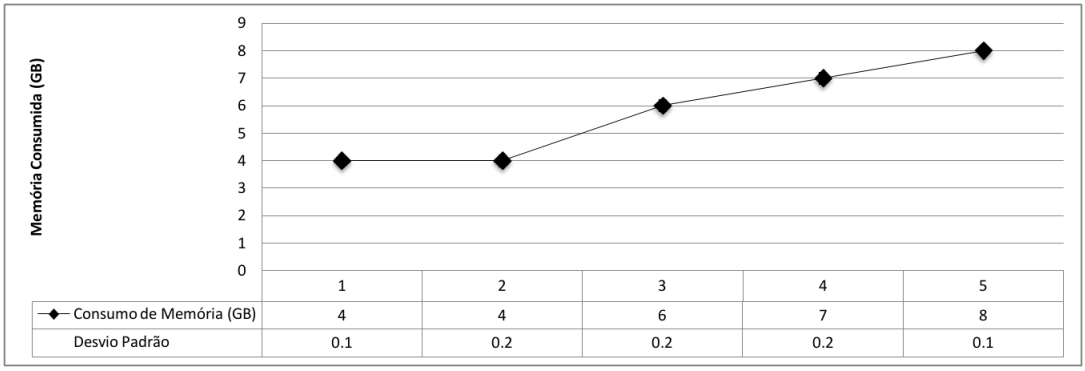


Figura 15 – Consumo de memória pelo componente de captura de dados (síncrono e assíncrono)

De acordo com a complexidade dessas transformações ou com o seu tipo (XPath ou Javascript) ocorrem variações de desempenho. Assim como na subseção anterior, será feito o GQM (Tabela 13) para descrever o objetivo, questões e métricas da avaliação deste componente.

Nesta avaliação, o componente de transformação de dados foi separado dos demais componentes do SemanticHub. A avaliação foi realizada com pacotes de dados em XML e JSON com tamanho de 10KB e 7KB respectivamente, contendo o mesmo número de estruturas de dados. As medidas foram feitas para cinco pontos, cada ponto representando uma quantidade de pacotes a serem transformados. Cada teste foi realizado durante 30 minutos, sendo os resultados apresentados são relativos a médias obtidas neste período de testes.

Para demonstrar o quanto os diferentes tipos de transformação podem interferir no desempenho deste componente, foram criados dois exemplos: um exemplo de transformação XPath, representada na Tabela 14 e aplicada sobre o conjunto de dados da Figura 16; e um exemplo de uma transformação Javascript, representada na Tabela 15 e aplicada sobre o conjunto de dados Figura 17. Tanto as transformações quanto o conjunto de dados de testes são equivalentes.

Para responder a primeira questão da Tabela 13, será avaliado o número de pacotes de dados transformados por unidade de tempo de acordo com o tipo de transformação aplicada . Os pacotes de dados em JSON e em XML que serão utilizados na avaliação (listagem 4.1 e 4.2) apresentam o mesmo número de blocos de dados (35 blocos). Cada bloco de dados irá gerar três triplas, totalizando 105 triplas por pacote. A avaliação será feita em cinco cenários: 500 pacotes de dados; 1000 pacotes de dados; 2000 pacotes de dados; 3000 pacotes de dados; e 5000 pacotes de dados. Para se determinar a taxa de transformação (número de pacotes transformados por unidade de tempo), foi

Tabela 13 – GQM para avaliação do componente de transformação de dados

Objetivo Propósito Problema Objeto	Analisar a capacidade de transformação de pacote de dados
Questão **Métricas**	Quantas triplas RDF podem ser geradas por unidade de tempo? -Número de pacotes transformados por segundo -Tipo de transformação (Xpath ou Javascript) -Desvio padrão
Questão **Métricas**	Qual o consumo de recursos computacionais em relação a transformação de dados? -Memória consumida -Percentual de processamento consumido -Desvio padrão

Tabela 14 – Regras de transformação para dados em XML

Regra de Mapeamento
Recurso **Tipo de recurso:** Recurso dinâmico **URI do recurso:** http://greco.ppgi.ufrj.br/resource# **RDF Type:** http://greco.ppgi.ufrj.br/ontology#data **Grupo de Dados:** /dataRead/data
Propriedades de dado **Tipo de propriedade:** Dinâmica **URI do recurso:** http://greco.ppgi.ufrj.br/resource# **Propriedade:** http://greco.ppgi.ufrj.br/property#hasValue **Grupo de Dados:** /dataRead/data **Transformação:** /value/text() **Tipo de propriedade:** Dinâmica **URI do recurso:** http://greco.ppgi.ufrj.br/resource# **Propriedade:** http://greco.ppgi.ufrj.br/property#hasDateTime **Grupo de Dados:** /dataRead/data **Transformação:** /current value/at

medido o tempo necessário para que o componente transformasse todos os pacotes de dados de cada cenário para RDF.

O número total de triplas geradas durante os 30 minutos de teste com 5000 pacotes por segundo foi de 945.000.000 triplas. Esse valor excede, de forma geral, o número de triplas em bases de dados na nuvem de LOD. Entretanto, o objetivo deste teste é mostrar a capacidade deste componente.

```
<dataRead>
    <data id="1">
        <tag>humidity</tag>
        <current_value at="2010-07-02T10:16:19␣.270708Z"
<value>10000.0</value>

    </data>
    ...
    <data id="n">
        <tag>humidity</tag>
        <current_value at="2010-07-02T10:16:27␣.278984Z"
<value>10001.0</value>
    </data>
</dataRead>
```

Figura 16 – Pacode de dados em XML

Tabela 15 – Regras de transformação para dados em JSON

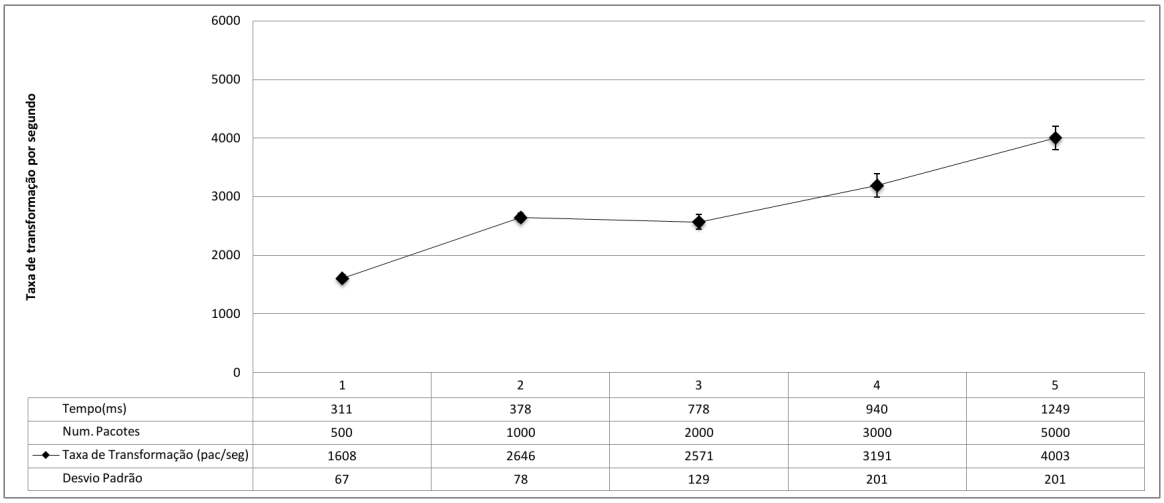
Regra de Mapeamento
Recurso **Tipo de recurso:** Recurso dinâmico **URI do recurso:** http://greco.ppgi.ufrj.br/resource# **RDF Type:** http://greco.ppgi.ufrj.br/ontology#data **Grupo de Dados:** data["dataRead"]["data"]
Propriedades de dado **Tipo de propriedade:** Dinâmica **URI do recurso:** http://greco.ppgi.ufrj.br/resource# **Propriedade:** http://greco.ppgi.ufrj.br/property#hasValue **Grupo de Dados:** data["dataRead"]["data"] **Transformação:** transf["value"] **Tipo de propriedade:** Dinâmica **URI do recurso:** http://greco.ppgi.ufrj.br/resource# **Propriedade:** http://greco.ppgi.ufrj.br/property#hasDateTime **Grupo de Dados:** data["dataRead"]["data"] **Transformação:** transf["current value"]

A Figura 18 apresenta os resultados obtidos para a transformação de XML para RDF. Na Figura 19 é apresentada a avaliação correspondente para pacotes de dados em JSON. Observou-se que a taxa de pacotes transformado através Javascript para pacotes JSON foi maior quando comparada com as transformações XPath para pacotes de dados em XML.

A segunda questão, relativa ao consumo de recursos computacionais pelo componente de transformação de dados, e apresentada na Tabela 13, é respondida através das métricas de percentual de processamento utilizado e consumo de memória durante a

```
{
        "dataRead ":[
            { "data":{
              " id":1  ,
              " tag" : " humidity " ,
              " current_value": "2010-07-02T10:16:19.270708Z ",
              "value ":10000
              }
           },
           ...
           {
           "data":{
             "id ":n ,
             " tag": "humidity ",
             " current_value":"2010-07-02T10:16:27.278984Z" ,
             " value ":10001.0}
           } ]
};
```

Figura 17 – Pacode de dados em JSON



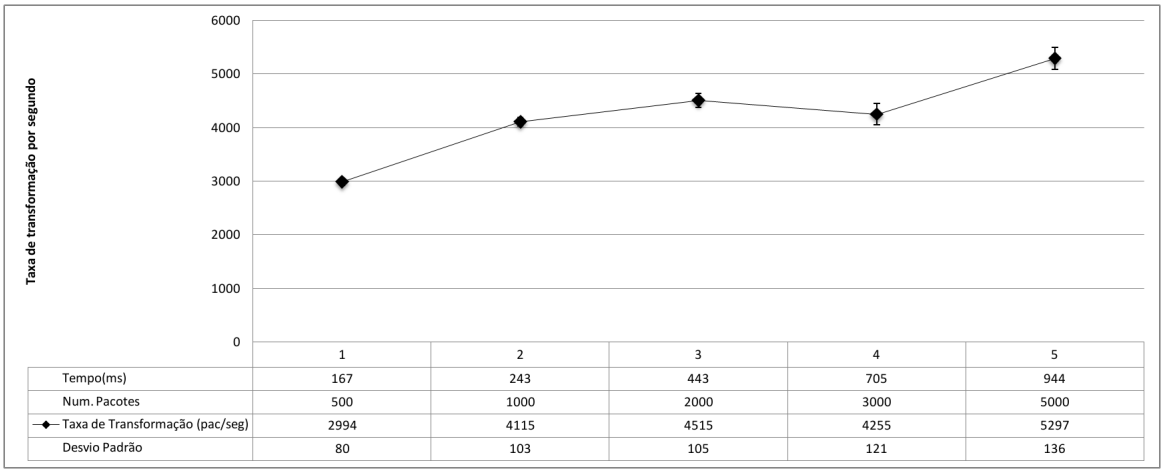
	1	2	3	4	5
Tempo(ms)	311	378	778	940	1249
Num. Pacotes	500	1000	2000	3000	5000
Taxa de Transformação (pac/seg)	1608	2646	2571	3191	4003
Desvio Padrão	67	78	129	201	201

Figura 18 – Número de pacotes XML transformados por segundo

realização dos experimentos. Estas métricas serão apresentadas em relação ao tipo de transformação utilizada (XPath ou Javascript) para os cinco cenários de experimento.

Em relação ao percentual de processamento utilizado, a primeira observação que deve ser feita é que as transformações Javascript consumiram mais processamento do que as transformações XPath. No gráfico da Figura 20 pode-se observar que no último cenário, com 5000 pacotes de dados sendo transformados por segundo para transformações XPath, o percentual de processamento utilizado foi menor do que o percentual de processamento para transformações em Javascript para 500 pacotes por segundo apresentado no gráfico da Figura 21. Entretanto, não houve aumento significativo no percentual de processamento das transformações Javascript para os diferentes cenários

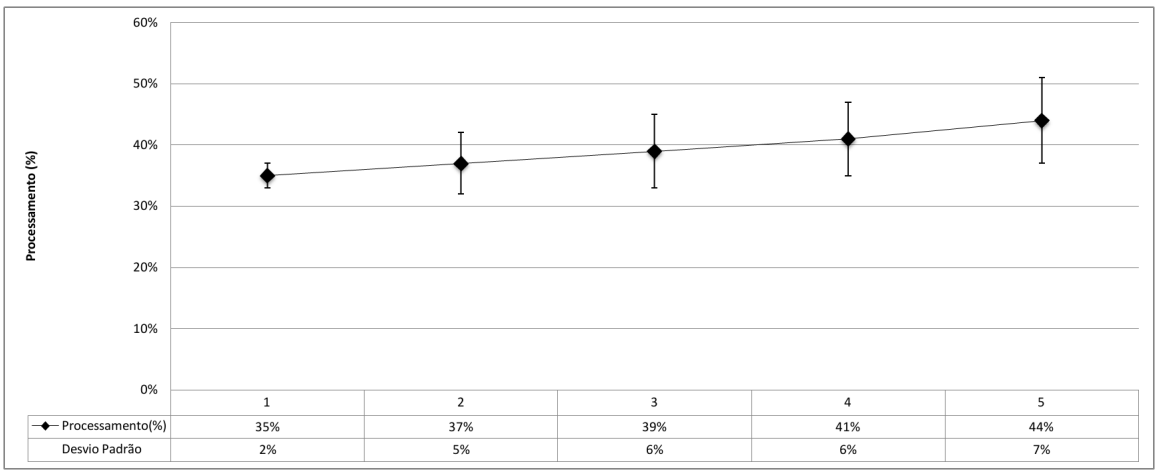
apresentados, sendo apenas 12% maior no último cenário quando comparado com o primeiro cenário.



	1	2	3	4	5
Tempo(ms)	167	243	443	705	944
Num. Pacotes	500	1000	2000	3000	5000
Taxa de Transformação (pac/seg)	2994	4115	4515	4255	5297
Desvio Padrão	80	103	105	121	136

Figura 19 – Número de pacotes JSON transformados por segundo

O consumo de memória, em ambos os tipos de transformação, cresceu até praticamente consumir a memória disponível. Neste caso, o crescimento do consumo de memória nos cenários de teste foi significativamente acentuado para as transformações XPath. No gráfico da Figura 22 é possível observar, quando comparado com o gráfico da Figura 23, referente ao consumo de memória das transformações Javascript, que inicialmente ambas as transformações iniciaram consumindo 4 GB. No quarto cenário a transformação XPath já consumia 1 GB a mais do que a transformação Javascript.

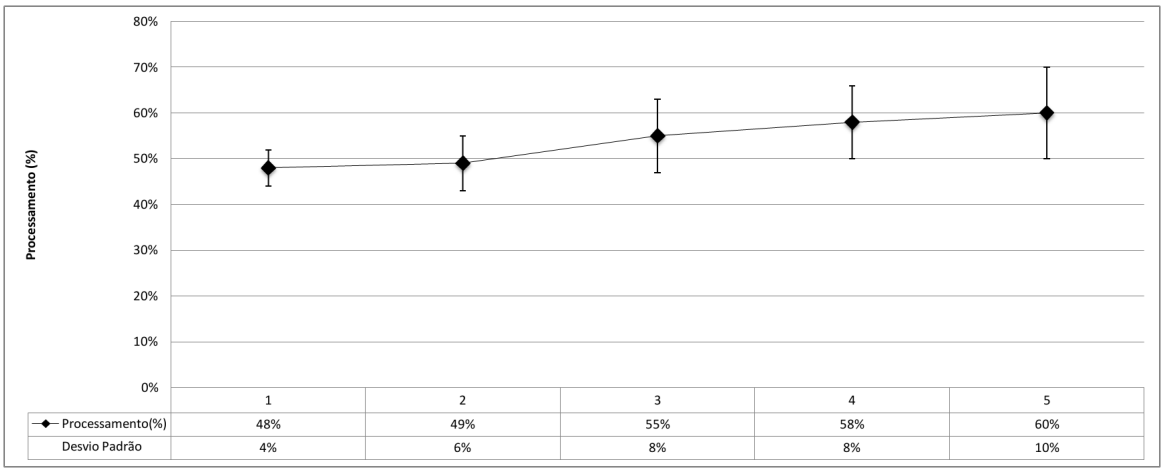


	1	2	3	4	5
Processamento(%)	35%	37%	39%	41%	44%
Desvio Padrão	2%	5%	6%	6%	7%

Figura 20 – Consumo de processamento para transformação de dados XPath

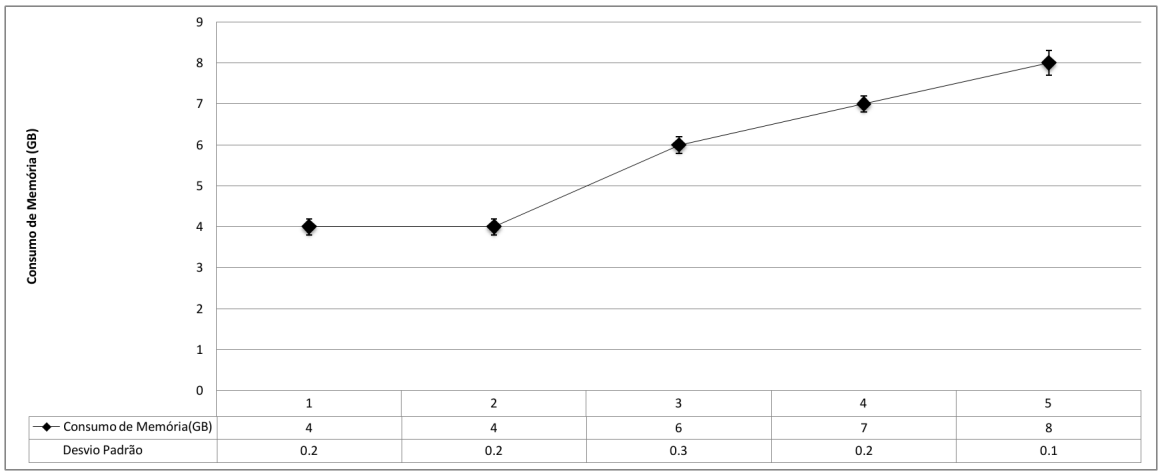
4.4.4 Avaliação do Componente de Alimentação de Dados RDF

O objetivo da avaliação do componente de armazenamento de triplas é determinar a taxa máxima de triplas inseridas em um repositório de triplas dado o mecanismo implementado. Este mecanismo, como descrito na subseção 4.3.3, faz a



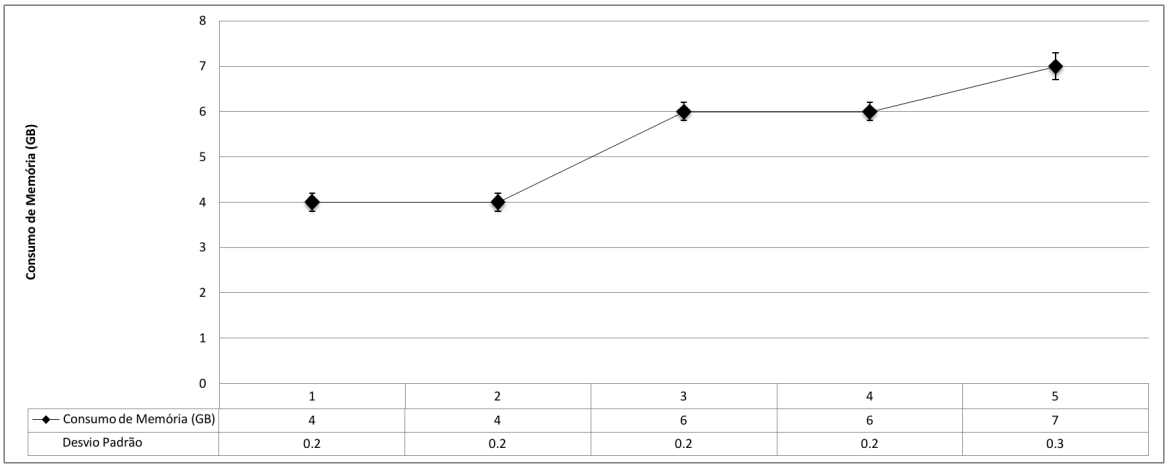
	1	2	3	4	5
Processamento(%)	48%	49%	55%	58%	60%
Desvio Padrão	4%	6%	8%	8%	10%

Figura 21 – Consumo de processamento para transformação de dados Javascript



	1	2	3	4	5
Consumo de Memória(GB)	4	4	6	7	8
Desvio Padrão	0.2	0.2	0.3	0.2	0.1

Figura 22 – Consumo de memória para transformação de dados Xpath



	1	2	3	4	5
Consumo de Memória (GB)	4	4	6	6	7
Desvio Padrão	0.2	0.2	0.2	0.2	0.3

Figura 23 – Consumo de memória para transformação de dados Javascript

inserção de blocos de triplas no repositório através da interface de acesso disponibilizada pelo mesmo. Quando um bloco de dados não é inserido, é feita a inserção partido-se o bloco de dados ao meio até que todas as triplas tenham sido armazenadas. As triplas

inválidas vão para um arquivo de log de controle.

Os resultados obtidos neste teste são dependente da capacidade do repositório de triplas utilizado e das conFigurações aplicadas a este repositórios. Para esta avaliação foi utilizada a versão 2.6.10 do repositório Sesame (Tabela 10), com mecanismo de serialização nativo. Os índices utilizados foram do tipo <sujeito,predicado,objeto>, pois refletem as consultas Sparql frequentemente feitas.

Para responder o GQM apresentado na Tabela 16 serão avaliados o número máximo de triplas que são inseridas por unidade de tempo em relação ao consumo de recursos computacionais (memória e processamento). Além disso, será avaliado o impacto que o percentual de triplas inválidas geram no desempenho do componente.

Tabela 16 – GQM para avaliação do componente de alimentação de dados RDF

Objetivo Propósito Problema Objeto	Analisar a capacidade de inserções no repositório de triplas RDF
Questão **Métricas**	Quantas triplas RDF podem ser inseridas por unidade de tempo? -Número de triplas inseridas por segundo -Consumo de memória -Consumo de processamento Desvio padrão
Questão **Métricas**	Qual o impacto que um determinado percentual de triplas inválidas causa no mecanismo de inserção? -Número de triplas inseridas por segundo -Percentual de triplas inválidas -Desvio padrão

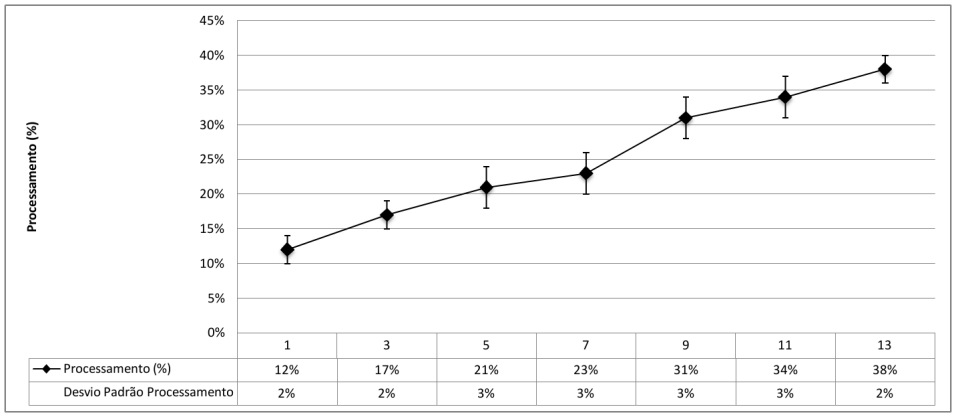
Os testes foram realizados com a inserção de quantidades diferentes de pacotes de triplas por unidade de tempo (cada pacote contendo 100 triplas RDF). Foram testadas a inserção de um, três, cinco, sete, nove, onze e treze pacotes. O número máximo de pacotes inseridos por segundo foi de treze (totalizando 1300 triplas) com desvio padrão de um.

Esta limitação foi devida ao repositório de triplas e não a limitações do componente de armazenamento. Os testes foram realizados durante o período de 30 minutos.

A Figura 24 apresenta o gráfico de processamento obtido no experimento. O eixo horizontal do gráfico representa o número de pacotes de triplas inseridas por

segundo. Pode-se observar que o componente apresentou comportamento linear. O consumo cresceu proporcionalmente ao número de pacotes inseridos.O desvio padrão não apresentou grandes variações, indicado que o comportamento deste componente é previsível para os valores observados. O valor inicial consumido foi de 12% do total disponível, terminando com 38% do total de processamento do computador onde o teste foi realizado.

O consumo de memória apresentou comportamento semelhante aos componentes avaliados anteriormente. O valor inicial obtido foi de 4GB para a inserção de apenas um único pacote de dados, consumindo toda memória disponível(8GB) para inserir 13 pacotes. O comportamento de consumo de memória pode ser visto no gráfico da Figura 25.

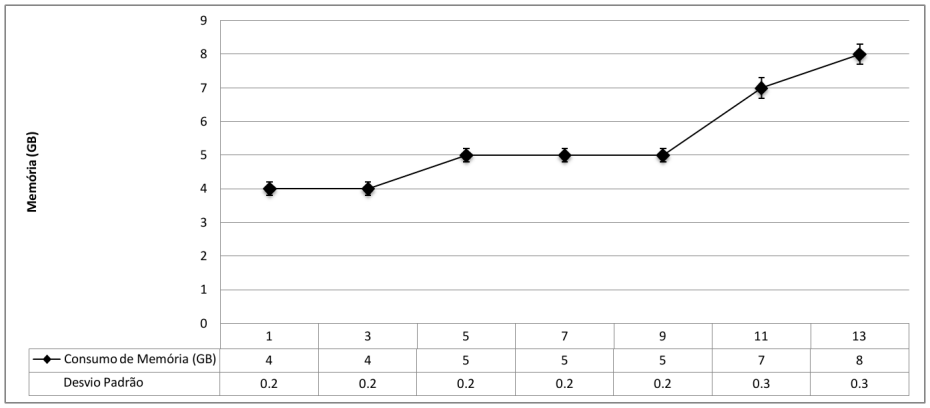


	1	3	5	7	9	11	13
Processamento (%)	12%	17%	21%	23%	31%	34%	38%
Desvio Padrão Processamento	2%	2%	3%	3%	3%	3%	2%

Figura 24 – Consumo de processamento para armazenamento de triplas

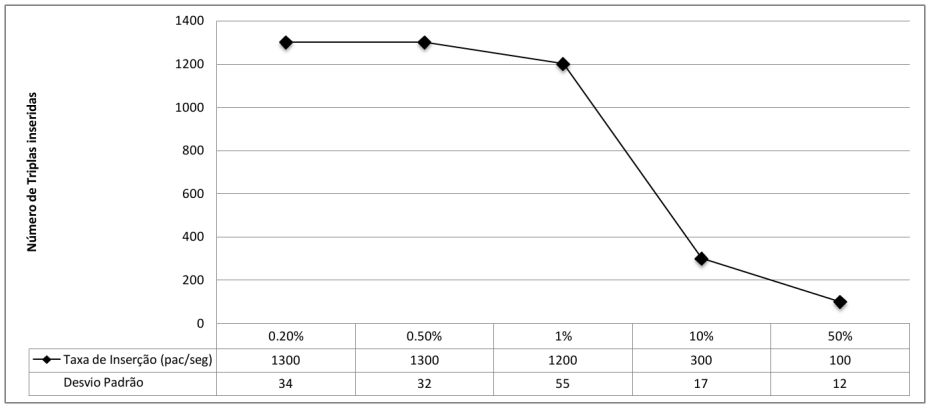
Para determinar o impacto sobre o mecanismo de inserção que determinados percentuais de triplas inválidas podem gerar, respondendo a segunda pergunta da Tabela 16, foram avaliadas as taxas de inserção de triplas em cinco cenários: 0,20% de triplas inválidas;0,50% de triplas inválidas;1% de triplas inválidas; 10% de triplas inválidas; e 50% de triplas inválidas. Estes percentuais são relativos ao todo, portanto, um mesmo pacote de inserção pode ter mais de uma tripla inválida. Esta avaliação foi executada durante o período de 30 min, os valores obtidos representando médias deste período.

Como pode ser visto no gráfico da Figura 26, percentuais pequenos, menores que 1% de triplas inválidas, não impactam o funcionamento do componente. Altos percentuais (superiores a 10%) por sua vez, reduzem muito o desempenho do componente, sendo, entretanto, são raros.



	1	3	5	7	9	11	13
Consumo de Memória (GB)	4	4	5	5	5	7	8
Desvio Padrão	0.2	0.2	0.2	0.2	0.2	0.3	0.3

Figura 25 – Consumo de memória para armazenamento de triplas



	0.20%	0.50%	1%	10%	50%
Taxa de Inserção (pac/seg)	1300	1300	1200	300	100
Desvio Padrão	34	32	55	17	12

Figura 26 – Impacto de triplas inválidas no mecanismo de inserção

4.4.5 Avaliação do Componente de Reconciliação de Entidades

O componente de Reconciliação de Entidades, como visto anteriormente, tem como objetivo manter a corretude dos dados no repositório de triplas através da identificação de recursos iguais que possuem URIs distintas. Como a estratégia utilizada por este componente é baseada em um método de avaliação tardia, os fatores considerados mais importantes para serem analisados são o tempo necessário para que as correções sejam aplicadas e o quanto diferentes percentuais de dados n-plicados impactam no funcionamento deste componente. Na Tabela 17 é apresentado o GQM utilizado na avaliação desse componente.

A avaliação foi realizada com um conjunto pré-definido de dados que foram produzidos especificamente pare este fim. Através desse conjunto de dados foi possível determinar o quanto e quais triplas RDF são esperadas no fim do processo e o percentual de duplicações de recursos presentes. Como o componente de Reconciliação de Entidades processa uma grande quantidade de triplas, é esperado que o consumo de memória e processamento sejam elevados.

Tabela 17 – GQM para avaliação do componente de reconciliação de entidades

Objetivo Propósito Problema Objeto	 Analisar a capacidade de manter a corretude do repositório de triplas
Questão **Métricas**	Uma vez que exista um recurso duplicado na base, quanto tempo é necessário para que este recurso seja reconciliado? -Quantidade de dados armazenados -Percentual de recursos n-plicados -Tempo até a correção -Desvio padrão
Questão **Métricas**	Qual o consumo de recursos computacionais durante o processo de reconciliação de entidades para diferentes cenários? -Memória consumida -Percentual de processamento consumido

Como existem dois parâmetros que afetam diretamente o funcionamento do componente de Reconciliação de Entidades, o tamanho do repositório de triplas e o percentual de triplas repetidas, estes parâmetros foram analisados separadamente.

Para analisar o impacto que o percentual de triplas inválidas pode gerar no funcionamento do arcabouço, fixou-se o tamanho do repositório de triplas em 3 milhões de triplas. O percentual de triplas inválidas testadas foi de 0,2%, 0,5%, 1%, 10% e 50%. De acordo com a primeira questão apresentada na Tabela 16, o que está sendo verificado é o tempo necessário para que as duplicações no repositório sejam corrigidas. O gráfico da Figura 27 apresenta os resultados obtidos. Para baixos percentuais (menores que 1%) o tempo necessário para que as correções sejam aplicadas é inferior a um minuto. Percentuais maiores obrigam o componente a trabalhar durante períodos maiores de tempo, entretanto, os períodos não crescem tanto em relação aos períodos necessários para baixos percentuais de n-plicação.

Para satisfazer a primeira pergunta da Tabela 17, foi necessário avaliar o impacto que grandes bases de triplas ocasionam no mecanismo de Reconciliação de Entidades. Para esta avaliação fixou-se o percentual de triplas n-plicadas em 10%, variando então o volume de dados no repositório em um milhão, três milhões, dez milhões, cinquenta milhões e cem milhões de triplas. Nos resultados obtidos, que podem ser vistos no gráfico da Figura 28, houve um maior impacto nos tempos necessários para que o componente pudesse finalizar as correções no repositório de triplas. No último caso, onde a base possui cem milhões de triplas, foram necessários 21 minutos para completar

o processo de reconciliação.

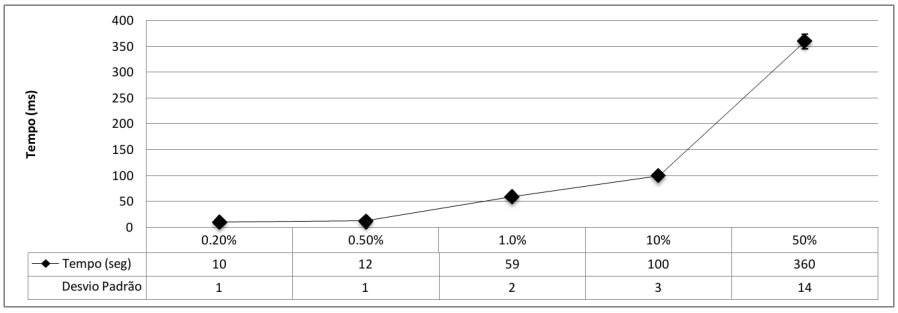


Figura 27 – Impacto de diferentes percentuais de triplas n-plicadas sobre o mecanismo de Reconciliação de Entidades

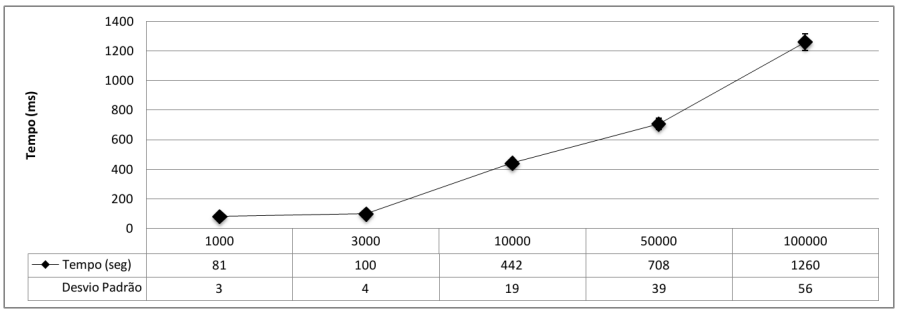


Figura 28 – Impacto de diferentes volumes de triplas RDF no repositório de triplas sobre o mecanismo de Reconciliação de Entidades

Respondendo a segunda questão da Tabela 17, o componente de Reconciliação de Entidades não apresentou variação do consumo dos recursos computacionais, pois, consumiu 6GB da memória disponível em todos os cenários testados, diferente do que ocorreu com os demais componentes do SemanticHub. O mesmo ocorreu com o percentual de processamento, que se manteve próximo a 50%. A implementação deste componente mostrou-se problemática devido ao recursos computacionais consumidos, entretanto, quando todas as peças do arcabouço estão em funcionamento, ela não interfere no funcionamento dos demais componentes. O funcionamento de todo o arcabouço será exemplificado no próximo capítulo, onde um cenário com dados de objetos reais será discutido.

5 ESTUDO DE CASO

No capítulo 4 foram apresentados a descrição dos componentes do SemanticHub, a conFiguração do arcabouço, uma implementação baseada na arquitetura e a avaliação desta implementação. Neste capítulo será discutido um cenário com objetos reais, que inclui sensores relacionados a parâmetros climáticos, como sensores de temperatura, umidade e gases. Para explicitar e estender a semântica dos dados foram conFiguradas regras de transformação que refletem as propriedades e conceitos da Semantic Sensor Network Ontology (SSNO) (NEUHAUS; COMPTON, 2009) e do vocabulário WGS84 (BRICKLEY, 2006) para dados geoespaciais. Por fim, estes dados serão armazenados e atualizados em um repositório de triplas e interligados com outros repositórios de triplas da nuvem de LOD. Também são apresentadas consultas Sparql sobre estes dados para demonstrar as potencialidades deste cenário.

5.1 DESCRIÇÃO DO CENÁRIO

As fontes de dados utilizados são provenientes da plataforma COSM (na verdade, após o desenvolvimento deste trabalho o nome da plataforma foi trocada para Xively). Cosm é uma plataforma em nuvem que provê a infraestrutura para publicação de objetos na Web. Os conceitos que a plataforma abrange podem ser vistos na Tabela 18.

Devido à disponibilidade de sensores ambientais funcionais, foi escolhido como localidade para o estudo de caso a cidade de Londres. O cenário escolhido para o teste foi o de clima, onde 350 objetos que efetuam atividade de sensoriamento foram selecionados. Dentre esses objetos, 342 executavam atividades de sensoriamento de temperatura, 21 de umidade e 9 de gases. O número de atividades de sensoriamento é superior ao número de dispositivos pois um mesmo dispositivo pode sensoriar mais de um parâmetro. O mapeamento entre os conceitos presentes no COSM e a ontologia/vocabulário utilizados está representado na Tabela 19.

5.2 CONFIGURAÇÃO

Como apresentado no capítulo 4, a primeira macro-etapa para o funcionamento do SemanticHub é a conFiguração das fontes de dados, definindo-se qual mecanismo

Tabela 18 – Descrição dos conceitos utilizados pela plataforma Cosm

Conceito	Descrição
Feed	Conjunto de canais (fluxos de dados) definidos para um dispositivo. Também abrange os metadados dos dispositivos, como localização, se o dispositivo é físico ou virtual, fixo ou móvel, interior ou exterior, etc
Fluxo de dados	Representa um atributo específico, unidade ou tipo de informação de um Feed. Por exemplo, leituras de um sensor ou comandos de uma aplicação
Produtos	Um produto é o mais alto nível de abstração na hierarquia do Cosm, representando um tipo específico de dispositivo real. A definição de um produto inclui o nome do produto, a sua descrição e um modelo de alimentação.
Gatilhos	Gatilhos, também referenciados como notificações, são interfaces, baseadas em Websockets, que enviam dados via HTTP POST. A diferença dos gatilhos para Websockets clássicos é que a aplicação cliente pode definir os parâmetros para envio de dados.

Tabela 19 – Quadro com mapeamento dos conceitos para vocabulários e ontologias utilizadas

Conceito	Mapeamento no Vocabulário/Ontologia
Produto	SSNO:Sensing Device
Feed	SSNO:Sensor Output
Datapoint	SSNO:Observation Value
Localização (metadado do produto)	SSNO:Platform e WGS84:point
ONDE: SSNO: Semantic Sensor Network Ontology DC: Dublin Core Vocabulary WGS84: WGS84 Vocabulary	

de captura de dados será utilizado (síncrono ou assíncrono), a taxa de atualização e as interfaces que devem ser consumidas.

Seguindo a descrição da Tabela 19, o primeiro recurso do Cosm a ser mapeado para o SemanticHub é o Produto. Produto corresponde a recursos estáticos, representando cada um dos 350 objetos que produzem dados. Na Tabela 20 é feita a conFiguração para um produto, representando uma placa de hardware da plataforma Arduino. Este produto apresenta como propriedades de dado o nome, a descrição e o seu estado, sendo este último uma propriedade dinâmica que é obtida pelo consumo de um serviço REST específico em intervalos de tempo de 60 segundos.

Produtos apresentam ligações externa com a DBPedia, através da conFiguração da propriedade de objetos owl:SameAs. Esta ligação provê o enriquecimento das informações do produto com descrição de restrições e formas de funcionamento dos dispositivos disponíveis em bases externas. Estas informações possibilitam a seleção de dispositivos, baseados nos requisitos das aplicações, de acordo com as restrições de funcionamento.

Tabela 20 – ConFiguração do SemanticHub para o estudo de caso

Mapeamento
Recursos **Tipo de recurso:** Estático **URI do recurso:** http://greco.ppgi.ufrj.br/resource#h5wT2IlOMrd09r-qd1jm **RDF-Type:** http://purl.oclc.org/NET/ssnx/ssn#Device
Propriedades de dados **Tipo de propriedade:** Estático **Propriedade:** http://purl.oclc.org/NET/ssnx/ssn#Name **Valor:** product example **Tipo de propriedade:** Estático **Propriedade:** http://purl.oclc.org/NET/ssnx/ssn#Description **Valor:** "this is a device on cosm" **Tipo de propriedade:** Dinâmico (único) **Propriedade:** http://purl.oclc.org/NET/ssnx/ssn#State **Fonte:** https://api.xively.com/v2/products/h5wT2IlOMrd09r-qd1jm **Frequência de consumo:** 60 segundos **Grupo:** data **Transformação:** data['state']
Propriedades de objetos **Sujeito:** http://greco.ppgi.ufrj.br/resource#h5wT2IlOMrd09r-qd1jm **Predicado:** http://www.w3.org/TR/owl-ref#sameAs **Objeto:** http://dbpedia.org/resource#ArduinoUno

Cada produto está associado a uma localização que é representada por um recurso do tipo WGS84:Point, o mapeamento deste recurso pode ser visto na Tabela 21. Como todos os produtos estão fixados em uma posição, esta é representada como um recurso estático.

O terceiro recurso a ser conFigurado são os feeds, que representam os sensores agregados em cada dispositivo. Como esses recursos não mudam ao longo do tempo, também foram conFigurados como recursos estáticos. Entretanto, possuem uma propriedade dinâmica que representa seu estado, mostrando se o sensor está ativo ou não. Esta

Tabela 21 – ConFiguração de uma localidade

Mapeamento
Recursos **Tipo de recurso:** Estático **URI do recurso:** http://greco.ppgi.ufrj.br/resource#1asdfasfd **RDF-Type:** http://www.w3.org/2003/01/geo/wgs84-pos#Point
Propriedades de dados **Tipo de propriedade:** Estático **Propriedade:** http://www.w3.org/2003/01/geo/wgs84-pos#lat **Valor:** "51.50318" **Tipo de propriedade:** Estático **Propriedade:** http://www.w3.org/2003/01/geo/wgs84-pos#long **Valor:**: -0.12701" **Tipo de propriedade:** Estático **Propriedade:** http://www.w3.org/2003/01/geo/wgs84-pos#alt **Valor:** "4"
Propriedades de objetos **Sujeito:** http://greco.ppgi.ufrj.br/resource#abc123 **Predicado:** http://purl.oclc.org/NET/ssnx/ssn#attachedSystem **Objeto:** http://greco.ppgi.ufrj.br/resource#1asdfasfd **Sujeito:** http://greco.ppgi.ufrj.br/resource#1asdfasfd **Predicado:** http://www.w3.org/TR/owl-ref#sameAs **Objeto:** http://linkedgeodata.org/triplify/node11018038

informação é obtida sincronamente a cada 60 segundos a partir do consumo do serviço REST específico para cada Feed. A ligação entre o sensor e a plataforma em que ele está implantado é feita através da propriedade de objetos SSNO:ObservationResult. Estas conFigurações e demais parâmetros podem ser vistos na Tabela 22.

As leituras obtidas dos sensores foram representadas como recursos dinâmicos do tipo SSNO:ObservationValue, sendo atualizadas através da interface assíncrona do Cosm e possuindo como informação além do próprio dado de sensoriamento o momento em que ele foi produzido. Exemplo de um pacote recebido no formato JSON pode ser visto na Figura 29. As transformações aplicadas são descritas na Tabela 23.

5.2.1 Resultados Obtidos

O cenário proposto permaneceu em execução pelo período de 30 dias ininterruptos, armazenando no repositório de triplas neste intervalo de tempo 1.254.427 triplas

Tabela 22 – ConFiguração de um Feed de Dados

Mapeamento
Recursos **Tipo de recurso:** Estático **URI do recurso:** : http://greco.ppgi.ufrj.br/resource#121601 **RDF-Type:** http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#SensorOutput
Propriedades de dados **Tipo de propriedade:** Dinâmico (único) **Propriedade:** http://purl.oclc.org/NET/ssnx/ssn#status **Fonte:** https://api.xively.com/v2/feeds/121601/ **Grupo:** data **Transformação:** data['state']
Propriedades de objetos **Sujeito:** http://greco.ppgi.ufrj.br/resource#121601 **Predicado:** http://purl.oclc.org/NET/ssnx/ssn#ObservedBy **Objeto:** http://greco.ppgi.ufrj.br/resource#h5wT2IlOMrd09r-qd1jm **Sujeito:** http://greco.ppgi.ufrj.br/resource#121601 **Predicado:** http://purl.oclc.org/NET/ssnx/ssn#observedProperty **Objeto:** http://purl.oclc.org/NET/ssnx/cf/cf-property#temperature

```
{
  "id": 121601,
  "title": "Demo",
  "private": "false",
  "feed": "https://api.xively.com/v2/feeds/121601.json", "status": "frozen",
  "updated␣": "2013-04-23T03:25:48.686462Z" ,
  "created␣": "2013-03-29T15:50:43.398788Z" ,
  "creator": "https://xively.com/users/calumbarnes", "version␣": "1.0.0" ,
  "datastreams␣": [
  {
    "id": "example",
    "current_value": "333",
    "at␣": "2013-04-23T01:10:02.986063Z" , "max_value": "333.0" ,
    "min_value␣": "333.0"} ,
  {
     "id": "key",
     "current_value": "value",
     "at": "2013-04-23T00:40:34.032979Z"}, {
     "id": "temp" }],
     "location␣": {
      "domain": "physical"
    }
 }
```

Figura 29 – Exemplo de JSON representando um feed na plataforma COSM

RDF. No gráfico da Figura 30 está apresentada a evolução do número de triplas ao longo do tempo, sendo o primeiro dia o que apresentou maior número de triplas inseridas dado que nele não só os recursos dinâmicos capturados foram armazenados mas também os

Tabela 23 – ConFiguração de um Fluxo de Dados

Mapeamento
Recursos **Tipo de recurso:** Dinâmico **URI do recurso:** http://greco.ppgi.ufrj.br/resource# **Fonte:** https://api.xively.com/v2/feeds/121601/ **Grupo:** data['datastream'] **RDF Type:** http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#ObservationValue
Propriedades de dados **Tipo de propriedade:** Dinâmico **Propriedade:** http://purl.oclc.org/NET/ssnx/ssn#hasValue **Fonte:** https://api.xively.com/v2/feeds/121601/ **Transformação:** data['value'] **Tipo de propriedade:** Dinâmico **Propriedade:** http://purl.oclc.org/NET/ssnx/ssn#created-at **Transformação:** data['at']
Propriedades de objetos **Sujeito:** http://greco.ppgi.ufrj.br/resource#121601 **Predicado:** http://purl.oclc.org/NET/ssnx/ssn#hasValue **Objeto:** http://greco.ppgi.ufrj.br/resource#

recursos estáticos. A partir daí o crescimento mostrou-se praticamente linear.

Durante o período do teste cada objeto gerou uma média de 3584 triplas, sendo a maior parte dessas triplas (93%) informações relativas a variações de temperaturas. Menos de 3% do total de triplas armazenadas representaram dados sobre gases ou umidade. Durante o teste a maior parte de sensores de gases e umidade tornou-se inoperantes segundo o Cosm.

O consumo de memória e processamento no computador onde o estudo de caso foi realizado mostrou-se baixo, muito próximo dos limites mínimos conFigurados, mostrando que seria possível adicionar maior número de dispositivos sem prejudicar a capacidade do arcabouço.

Devido à utilização de comunicação assíncrona através dos WebHooks do Cosm, o componente de Reconciliação de Entidades corrigiu apenas 123 triplas. Se o consumo dos dados de sensoriamento tivesse sido realizado de forma síncrona seria esperado que este valor fosse muito maior devido ao recebimento de dados repetidos.

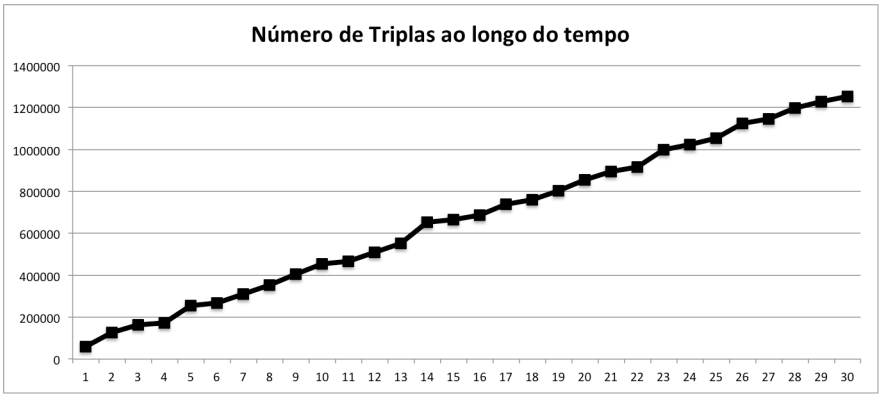


Figura 30 – Número de triplas armazenadas no repositório de triplas ao longo de 30 dias

5.2.2 Exemplo de Aplicação

Com o objetivo de mostrar as potencialidades do cenário utilizado, ainda que este cenário seja simples, foram feitas duas análises sobre os dados climáticos obtidos. Dadas as próprias características de LOD, as aplicações que permitem explorar o potencial do cenários criado são as que fazem agregação com o conjunto de dados capturados e que se aproveitam da ligação com outras bases. Apesar de não se a melhor abordagem para aplicações orientadas a eventos, onde a minimização do atraso entre a publicação do dado e a sua utilização pode ser mandatória, ainda assim seria possível utilizar esta abordagem se os requisitos desta aplicação permitirem atrasos na ordem de dezenas de segundos.

A primeira análise criada teve como objetivo a determinação da média de temperatura por localidade ao longo do intervalo de tempo do experimento. Na Figura31 está representada a consulta Sparql que responde a esta necessidade de informação. Na Tabela 26 está apresentada uma amostra com o resultado desta consulta. O ganho desta consulta em comparação à utilização apenas da API do COSM deve-se principalmente a capacidade de se utilizar dados históricos.

Uma análise análoga a anterior, mas com uma extensão das informações obtidas, seria não somente definir a média de temperatura por localidade, mas os tipos de construções próximas. Esta informação poderia ser utilizada para auxiliar, por exemplo, o planejamento urbano, definindo zonas onde a construção de determinadas estruturas são mais adequadas. As informações sobre as construções não estão disponíveis na base criadas, mas podem ser obtidas aproveitando-se as ligações com a base LikedGeoData. Para o exemplo de consulta apresentado na Figura 31, assume-se que existe algum mecanismo para realização de consultas federadas, permitindo a consulta entre bases distintas, de forma transparente. Os resultados para um único ponto podem ser vistos na Tabela 25.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix ssno: <http://purl.oclc.org/NET/ssnx/ssn#>.
@prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#>.

SELECT ?lat ,?long ,(AVG(?temp) AS ?temp) WHERE
{
?device rdf:type ssno:Device.
?device ssno : attachedSystem ?point ;
?point wgs84:long ?long.
?point wgs84: lat ?lat .
?device ssno:ObservedBy ?feed.
?feed ssno : HasValue ?data .
?feed ssno : observedProperty ssno : temperature . ?data ssno : value ?temp .
}
GROUPBY?lat ,?long
```

Figura 31 – Exemplo de consulta Sparql para obter a média de temperatura por localidade

Tabela 24 – Resuldado da Consulta Sparql apresentada na Figura 31

Latitude	**Longitude**	**Temperatura média (^{0}C)**
51.5624	-0.1760	12
51.5720	-0.1752	12
51.5296	-0.1657	15
51.5204	-0.3473	12
51.4198	-0.4016	14
51.5198	-0.1291	13
51.5235	-0.1380	12

Os dois exemplos apresentados são simples, mas evidenciam as potencialidades dos dados produzidos por objetos seguindo os princípios de LOD. Com esse cenário foi possível exemplificar como o SemanticHub permite a integração dos dados de sensoriamento utilizando as propostas mais recorrentes para integração de objetos na Web e o ganho que esta abordagem oferece para o consumo de dados produzidos por objetos. A implementação não exige modificações nas atuais infraestruturas que estão sendo utilizadas na integração de objetos na Web, sendo portanto, não invasiva do ponto de vista dos objetos.

Tabela 25 – Resuldado da Consulta Sparql apresentada na Figura 32

Latitude	**Longitude**	**Temperatura média (^{0}C)**	**Ponto**	**Tipo do Ponto**
51.5235	-0.1380	12	Fitzroy Tavern	Amenidade
51.5235	-0.1380	12	The Jeremy Bentham	Amenidade
51.5235	-0.1380	12	Euston	Ferrovia
51.5235	-0.1380	12	Woburn Cafe	Café
51.5235	-0.1380	12	Fitzrovia	Lugar

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix ssno: <http://purl.oclc.org/NET/ssnx/ssn#>.
@prefix wgs84:<http://www.w3.org/2003/01/geo/wgs84_pos#>.
@prefix geo : <http :// linkedgeodata . org#>.

SELECT ?lat ,?long ,(AVG(?temp) AS ?temp), ?point_label , ?point_type

WHERE
{
  ?device       rdf:type  ssno:Device.
  ?device       ssno:attachedSystem ?point;
  ?point        wgs84:long ?long.
  ?point        wgs84:lat  ?lat.
  ?device       ssno:ObservedBy ?feed.
  ?feed         ssno:HasValue ?data.
  ?feed         ssno:observedProperty ssno:temperature.
  ?data         ssno:value    ?temp.

  ?point_of_inteREST geo:geometry ?geo.
  ?point_of_inteREST rdfs:label ?point_label.
  ?point_of_inteREST rdfs:type ?point_type.

    FILTER (bif:st_intersects(?geo, bif:st_point(?lat ,?long),100))
}
GROUPBY?lat ,?long
```

Figura 32 – Exemplo de consulta Sparql para captura a média da temperatura por localidade relacionando com os pontos do LikedGeoData

6 CONCLUSÃO

Este trabalho apresentou um arcabouço, denominado SemanticHub, para integrar dados produzidos por objetos do mundo real na Web de Dados Interligados. A motivação principal para a criação deste arcabouço é: o grande volume e a grande variabilidade de formatos de dados que estão sendo gerados por objetos (volume este que tendem a crescer, segundo as previsões da União Internacional de Telecomunicações), irão causar problemas para sua manipulação e utilização.

A utilização dos dados de maneira transparente pelos usuários permitirá a utilização de uma grande variabilidade de fontes de dados e de grandes volumes através do processamento automatizado desses dados. Por utilização de maneira transparente entenda-se sem a necessidade de conhecimentos prévios sobre o domínio e sobre detalhes de implementação dos mecanismos de comunicação dos objetos.

O objetivo do SemanticHub é permitir que os dados produzidos por objetos sejam processáveis de maneira automatizada através da utilização de tecnologias semânticas. Para isso, o SemanticHub faz a explicitação dos significados dos dados e permite a extensão do domínio através da adição de descritores. Estas informações, que antes eram disponíveis apenas em documentos que somente seres humanos podem interpretar, passam a estar disponíveis para o processamento de máquinas.

Para o desenvolvimento do SemanticHub foram considerados os seguintes requisitos: permitir a comunicação com os objetos integrados na Web através de serviços Web do tipo REST e de métodos assíncronos de comunicação; permitir que os dados produzidos pelos objetos fossem convertidos para formatos adotados na Web Semântica; e permitir a consulta sobre os dados capturados.

Tendo em vista os requisitos e objetivos apresentados, o SemanticHub foi criado como um mediador entre as principais abordagens adotadas pela Web das Coisas e os princípios da Web de Dados. O SemanticHub realiza cinco funções básicas: capturar os dados dos objetos; transformar os dados recebidos para RDF; armazenar estes dados em uma base de dados apropriada; remover inconsistências destas bases; e prover mecanismos de acesso.

Um estudo de caso foi criado, mostrando o comportamento do arcabouço durante um mês para um total de 350 objetos que produzem dados relacionados a clima.

Os dados foram interligados com outras bases da nuvem de LOD e consultas mostraram a potencialidade do cenário, tanto na integração de dados de objetos através da utilização das tecnologias semânticas, quanto o ganho que se obtém quando estes dados estão interligados com outras bases.

Com base nos testes realizados e diferentemente dos trabalhos encontrados na literatura, o SemanticHub traz a capacidade de publicar dados de objetos utilizando as atuais propostas de integração de objetos na Web, transformando estes dados e fazendo com que sigam as propostas adotadas por tecnologias semânticas, ou seja, o SemanticHub é capaz de trabalhar com as abordagens que estão sendo empregadas, trazendo os benefícios do processamento automatizado sem que seja necessário modificações nas atuais abordagens de integração de objetos na Web.

O SemanticHub mostrou-se como uma solução adequada para integrar os dados produzidos por objetos na Web de Dados Interligados, desde que o volume de dados armazenados não seja superior à capacidade física de armazenamento disponível. Outro ponto importante é que não é necessário aplicar modificações nas aplicações que os objetos utilizam, permitindo que toda a estrutura desenvolvida com estes objetos seja mantida. Sob a perspectiva das propostas de LOD, a utilização de um repositório de triplas que permite o acesso navegacional e através de consultas Sparql permite que os métodos e tecnologias adotados possam ser utilizados, tornando os dados dos objetos apenas mais uma fonte de dados que pode ser utilizada.

Como trabalhos futuros, uma primeira possibilidade seria experimentar outras técnicas de exploração com diferentes métodos de comunicação, como por exemplo a comunicação direta com objetos através de interfaces físicas como conexões através de cabos seriais. Assim, objetos que não estivessem integrados diretamente na Web poderiam ser utilizados, provendo benefício duplo, tanto para a integração na Web quanto por utilizar técnicas que permitem a computabilidade de grandes volumes de dados. Outro trabalho futuro a ser explorado seria a formalização dos mecanismos de transformação de dados através da construção de um fluxo de transformação com componentes pré-definidos. Esta característica adotada por abordagens de data warehouse permite não só a programação visual, que facilita o processo de construção de transformações de dados, como a captura de proveniência dos dados. A proveniência é importante pois permite saber todos as etapas que deram origem a um determinado dado.

Um segundo trabalho futuro diz respeito aos mecanismos de reconciliação de entidades mais sofisticados, que poderiam ser testados para manter a unicidade da base de triplas RDF, além da avaliação de técnicas para armazenamento de grandes volumes

de dados em RDF de forma distribuída, contornando a limitação do volume de dados que podem ser inseridos, armazenados e consumidos.

Ainda um último trabalho futuro, mecanismos mais eficientes para ligar os dados com outros bases da nuvem de LOD podem ser explorados, permitindo o ganho de informação a partir das ligações entre dados. Estes mecanismos poderiam ser utilizados tanto após os dados terem sidos inseridos, quanto a partir da análise prévia dos dados gerados que irão ser inseridos. Durante o desenvolvimento do mecanismos de transformação de dados, percebeu-se que ainda que o tenha sido criado tem a capacidade de lidar com grande parte do cenário, a utilização de uma linguagem de programação, como Javascript, mostrou-se mais flexível do que uma linguagem de consulta como XPath. A padronização de uma única linguagem para transformação de dados e a formalização do processo, através da construção de um mecanismo de Extração-Transformação-Carregamento baseados nas ontologias e vocabulários que estão sendo empregados, poderia estender a capacidade do componente de transformação quanto simplificar este processo para o usuário que faz a conFiguração do arcabouço.

REFERÊNCIAS

ALLEMANG, D.; HENDLER, J. Semantic web for the working ontologist: Effective modeling in rdfs and owl. Morgan Kaufmann, 2008. Citado na página 25.

ALONSO, G. et al. *Web services: concepts, architectures and applications*. [S.l.]: Berlin, Germany: Springer-Verlag„ 2003. Citado na página 26.

ASHTON, K. That 'internet of things' thing. *RFiD Journal*, v. 22, n. 7, 2011. Citado 3 vezes nas páginas 17, 21 e 27.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer networks*, Elsevier, v. 54, n. 15, p. 2787–2805, 2010. Citado 4 vezes nas páginas 8, 15, 16 e 21.

BERNERS-LEE, T. Linked data-design issues. *http://www. w3. org/DesignIssues/LinkedData. html*, 2006. Citado na página 26.

BERNERS-LEE, T. Linked data-design issues (2006). *URL http://www. w3. org/DesignIssues/LinkedData. html*, v. 10, n. 11, 2011. Citado na página 24.

BERNERS-LEE, T.; FIELDING, R.; FRYSTYK, H. *Hypertext transfer protocol–HTTP/1.0*. [S.l.], 1996. Citado na página 22.

BERNERS-LEE, T.; FIELDING, R. T.; MASINTER, L. Uniform resource identifier (uri): Generic syntax. 2005. Citado na página 23.

BERNERS-LEE, T. et al. The semantic web. *Scientific american*, v. 284, n. 5, p. 34–43, 2001. Citado 3 vezes nas páginas 16, 22 e 24.

BIZER, C. The emerging web of linked data. *IEEE intelligent systems*, IEEE, v. 24, n. 5, p. 87–92, 2009. Citado na página 16.

BRAY, T. et al. Extensible markup language (xml). *World Wide Web Journal*, v. 2, n. 4, p. 27–66, 1997. Citado na página 23.

BREITMAN, K. *Web semântica: a internet do Futuro. Rio de Janeiro: LTC, 2006*. [S.l.], 2006. Citado na página 16.

BRICKLEY, D. Basic geo (wgs84 lat/long) vocabulary. *Documento informal escrito en colaboración*, 2006. Citado na página 64.

BRÖRING, A. et al. Semantic challenges for sensor plug and play. In: SPRINGER. *International Symposium on Web and Wireless Geographical Information Systems*. [S.l.], 2009. p. 72–86. Citado na página 32.

CALDIERA, V.; ROMBACH, H. D. The goal question metric approach. *Encyclopedia of software engineering*, v. 2, n. 1994, p. 528–532, 1994. Citado na página 49.

CYGANIAK, R.; JENTZSCH, A. Linking open data cloud diagram, 2011. *URL http://lod-cloud. net*, 2011. Citado na página 24.

DECKER, S. et al. A query and inference service for rdf. In: *QL'98-The Query Languages Workshop*. [S.l.: s.n.], 1998. v. 96. Citado na página 24.

DINU, V.; NADKARNI, P. Guidelines for the effective use of entity–attribute–value modeling for biomedical databases. *International journal of medical informatics*, Elsevier, v. 76, n. 11, p. 769–779, 2007. Citado na página 24.

DUQUENNOY, S.; GRIMAUD, G.; VANDEWALLE, J.-J. The web of things: interconnecting devices with high usability and performance. In: IEEE. *Embedded Software and Systems, 2009. ICESS'09. International Conference on*. [S.l.], 2009. p. 323–330. Citado na página 22.

FERNÁNDEZ, J. D.; MARTÍNEZ-PRIETO, M. A.; GUTIERREZ, C. Compact representation of large rdf data sets for publishing and exchange. In: SPRINGER. *International Semantic Web Conference*. [S.l.], 2010. p. 193–208. Citado na página 25.

FIELDING, R. T.; TAYLOR, R. N. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine Doctoral dissertation, 2000. Citado na página 27.

FRANÇA, T. C. *Infraestrutura de Software Baseada na Web das Coisas para Integração de Redes de Sensores Sem Fio à Web*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2012. Citado 2 vezes nas páginas 15 e 22.

FRIEDLANDER, D.; PHOHA, S. Semantic information fusion for coordinated signal processing in mobile sensor networks. *The International Journal of High Performance Computing Applications*, Sage Publications Sage UK: London, England, v. 16, n. 3, p. 235–241, 2002. Citado na página 31.

GRAY, A. J. et al. A semantically enabled service architecture for mashups over streaming and stored data. In: SPRINGER. *Extended Semantic Web Conference*. [S.l.], 2011. p. 300–314. Citado na página 33.

GUINARD, D. Towards opportunistic applications in a web of things. In: IEEE. *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. [S.l.], 2010. p. 863–864. Citado 2 vezes nas páginas 21 e 22.

GUINARD, D.; TRIFA, V. M.; WILDE, E. Architecting a mashable open world wide web of things. 2010. Citado 2 vezes nas páginas 21 e 22.

KATASONOV, A. et al. Smart semantic middleware for the internet of things. *ICINCO-ICSO*, v. 8, p. 169–178, 2008. Citado na página 32.

LANTHALER, M.; GUTL, C. Aligning web services with the semantic web to create a global read-write graph of data. In: IEEE. *Web Services (ECOWS), 2011 Ninth IEEE European Conference on*. [S.l.], 2011. p. 15–22. Citado 3 vezes nas páginas 17, 26 e 37.

LE-PHUOC, D. et al. Live linked open sensor database. In: ACM. *Proceedings of the 6th International Conference on Semantic Systems*. [S.l.], 2010. p. 46. Citado na página 33.

LI, L.; TAYLOR, K. A framework for semantic sensor network services. *Service-Oriented Computing–ICSOC 2008*, Springer, p. 347–361, 2008. Citado na página 33.

LIU, J.; ZHAO, F. Towards semantic services for sensor-rich information systems. In: IEEE. *Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on*. [S.l.], 2005. p. 967–974. Citado na página 31.

MAYER, S. Deployment and mashup creation support for smart things. *Institute for Pervasive Computing Department of Computer Science ETH Zurich*, 2010. Citado na página 27.

MCGUINNESS, D. L.; HARMELEN, F. V. et al. Owl web ontology language overview. *W3C recommendation*, v. 10, n. 10, p. 2004, 2004. Citado na página 25.

NEUHAUS, H.; COMPTON, M. The semantic sensor network ontology. In: *AGILE workshop on challenges in geospatial data harmonisation, Hannover, Germany*. [S.l.: s.n.], 2009. p. 1–33. Citado na página 64.

PAUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F. Restful web services vs. big'web services: making the right architectural decision. In: ACM. *Proceedings of the 17th international conference on World Wide Web*. [S.l.], 2008. p. 805–814. Citado 2 vezes nas páginas 26 e 29.

PEREIRA, B. *Resolução de Entidades Nomeadas utilizando Recursos em Linked Data*. Tese (Doutorado) — Dissertação de mestrado, PPGI-Programa de pósgraduação em Informática-IM-Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2012. Citado na página 17.

PFISTERER, D. et al. Spitfire: toward a semantic web of things. *IEEE Communications Magazine*, IEEE, v. 49, n. 11, p. 40–48, 2011. Citado na página 33.

PRUD'HOMMEAUX, E.; SEABORNE, A. *SPARQL query language for RDF. W3C Recommendation (January 15, 2008)*. 2011. Citado na página 23.

RICHARDSON, L.; RUBY, S. *RESTful web services*. [S.l.]: "O'Reilly Media, Inc.", 2007. Citado na página 22.

SEMANTICWEB. 2009. <https://www.w3.org/standards/semanticweb/>. Accessed: 2013-07-20. Citado 3 vezes nas páginas 8, 23 e 29.

SUNDMAEKER, H. et al. Vision and challenges for realising the internet of things. *Cluster of European Research Projects on the Internet of Things, European Commision*, v. 3, n. 3, p. 34–36, 2010. Citado na página 15.

TILKOV, S. A brief introduction to rest. *InfoQ, Dec*, v. 10, 2007. Citado 2 vezes nas páginas 27 e 30.

TRIFA, V. et al. Web messaging for open and scalable distributed sensing applications. *Web Engineering*, Springer, p. 129–143, 2010. Citado na página 38.

UNION, I. T. The internet of things—executive summary. *ITU Internet Reports*, 2005. Citado na página 15.

WEERAWARANA, S. et al. *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. [S.l.]: Prentice Hall PTR, 2005. Citado na página 26.

WEI, W.; BARNAGHI, P. Semantic annotation and reasoning for sensor data. In: SPRINGER. *European Conference on Smart Sensing and Context*. [S.l.], 2009. p. 66–76. Citado na página 32.

WHITEHOUSE, K.; ZHAO, F.; LIU, J. Semantic streams: A framework for composable semantic interpretation of sensor data. *Wireless Sensor Networks*, Springer, p. 5–20, 2006. Citado na página 31.