UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

MAYKE FERREIRA ARRUDA

Um modelo ontológico e um serviço de gerenciamento de dados de apoio à privacidade na Internet das Coisas

Goiânia
2019

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do material bibliográfico: **[X] Dissertação** **[ ] Tese**

2. Identificação da Tese ou Dissertação:

Nome completo do autor: Mayke Ferreira Arruda

Título do trabalho: Um modelo ontológico e um serviço de gerenciamento de dados de apoio à privacidade na Internet das Coisas

3. Informações de acesso ao documento:

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

Assinatura do(a) autor(a)[2]

Ciente e de acordo:

Assinatura do(a) orientador(a)[2]

Data: 08 / 02 / 2019

[1]Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.

Casos de embargo:
- Solicitação de registro de patente
- Submissão de artigo em revista científica
- Publicação como capítulo de livro
- Publicação da dissertação/tese em livro

[2]A assinatura deve ser escaneada.

MAYKE FERREIRA ARRUDA

Um modelo ontológico e um serviço de gerenciamento de dados de apoio à privacidade na Internet das Coisas

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. Renato de Freitas Bulcão Neto

Goiânia
2019

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Ferreira Arruda, Mayke
Um modelo ontológico e um serviço de gerenciamento de dados de apoio à privacidade na Internet das Coisas [manuscrito] / Mayke Ferreira Arruda. - 2019.
CXLVII, 147 f.: il.

Orientador: Prof. Dr. Renato de Freitas Bulcão Neto.
Dissertação (Mestrado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação, Goiânia, 2019.
Bibliografia. Apêndice.
Inclui siglas, abreviaturas, símbolos, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Internet das Coisas. 2. Privacidade. 3. Ontologia. 4. Serviço de Gerenciamento. 5. Web Semântica. I. de Freitas Bulcão Neto, Renato, orient. II. Título.

CDU 004

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UFG

ATA Nº 01/2019

ATA DA SESSÃO DE JULGAMENTO DA DISSERTAÇÃO DE MESTRADO DE MAYKE FERREIRA ARRUDA

Aos oito dias do mês de fevereiro de dois mil e dezenove, às oito horas e trinta minutos, na sala 150 do Instituto de Informática da Universidade Federal de Goiás, Campus Samambaia, reuniu-se a banca examinadora designada na forma regimental pela Coordenação do Curso para julgar a dissertação de mestrado intitulada **"Um modelo ontológico e um serviço de gerenciamento de dados de apoio à privacidade na Internet das Coisas"**, apresentada pelo aluno Mayke Ferreira Arruda como parte dos requisitos necessários à obtenção do grau de Mestre em Ciência da Computação, área de concentração Ciência da Computação. A banca examinadora foi presidida pelo orientador do trabalho de dissertação, Professor Doutor Renato de Freitas Bulcão Neto (INF/UFG), tendo como membros os Professores Doutores Cássio Vinicius Serafim Prazeres (DCC/UFBA) e Rita Cristina Galarraga Berardi (DAINF/UTFPR). Os professores Cássio Vinicius Serafim Prazeres e Rita Cristina Galarraga Berardi (DAINF/UTFPR) participaram à distância por webconferência. Aberta a sessão, o candidato expôs seu trabalho. Em seguida, o aluno foi arguido pelos membros da banca e:

(X) tendo demonstrado suficiência de conhecimento e capacidade de sistematização do tema de sua dissertação, a banca concluiu pela **aprovação** do candidato, sem restrições.

( ) não tendo demonstrado suficiência de conhecimento e capacidade de sistematização do tema de sua dissertação, a banca concluiu pela **reprovação** do candidato.

Os trabalhos foram encerrados às 11:15 horas. Nos termos do Regulamento Geral dos Cursos de Pós-Graduação desta Universidade, lavrou-se a presente ata que, lida e julgada conforme, segue assinada pelos membros da banca examinadora.

Prof. Dr. Renato de Freitas Bulcão Neto Renato Bulcão Neto

Prof. Dr. Cássio Vinicius Serafim Prazeres Cássio V. S. Prazeres

Profa. Dra. Rita Cristina Galarraga Berardi Rita Cristina Galarraga Berardi

Mayke Ferreira Arruda

Bacharel e Mestrando em Ciências da Computação pelo Instituto de Informática da Universidade Federal de Goiás (INF-UFG). Sua pesquisa foi baseada em soluções semânticas para a privacidade em Internet das Coisas. Suas áreas de interesse incluem Web Semântica, Engenharia de Software e Internet das Coisas.

Dedico este trabalho à minha família, em especial à minha mãe, que me ensinou a importância dos estudos, me incentivando neste longo caminho até aqui.

Agradecimentos

Agradeço a Deus pela saúde e força concedidas para enfrentar e superar todas as dificuldades encontradas ao longo do caminho.

À minha família por ser a base da minha educação e por sempre me apoiarem em meus objetivos, em especial à minha mãe Ivanilda por todos os ensinamentos, exemplos e apoio incondicional, sem os quais eu não teria chegado até aqui.

Ao meu amor Karolaine, por me dar suporte nessa jornada e nos momentos em que mais precisei. Obrigado por estar ao meu lado sempre, fazendo os meus dias mais especiais.

Ao meu orientador, Prof. Dr. Renato de Freitas Bulcão Neto, por todo o incentivo, conselhos e ensinamentos, os quais me proporcionaram a capacitação necessária para a conclusão deste trabalho. Espero que possamos trabalhar juntos mais vezes!

A todos os meus amigos e colegas de faculdade pela ajuda e companhia. Por fim, agradeço a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado a todos vocês.

A ciência nunca resolve um problema sem criar pelo menos outros dez.

George Bernard Shaw.

Resumo

Arruda, Mayke Ferreira. **Um modelo ontológico e um serviço de gerenciamento de dados de apoio à privacidade na Internet das Coisas**. Goiânia, 2019. 146p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

No paradigma da Internet das Coisas (IoT), objetos do mundo real equipados com recursos de identificação, detecção, rede e processamento comunicam-se e consomem serviços pela Internet para executar alguma tarefa em nome dos usuários. Devido à crescente popularização de dispositivos com capacidades de sensoriamento e ao consequente aumento da produção de dados oriundos desses dispositivos, a literatura afirma que o design de um modelo baseado em ontologias é um ponto de partida essencial para abordar os riscos de privacidade em IoT, uma vez que os dispositivos conectados são cada vez mais capazes de monitorar atividades humanas. Além disso, devido à complexidade e dinamicidade de ambientes IoT, enfatizamos a necessidade de ontologias de privacidade que combinem um vocabulário expressivo e extensível, mas que não sobrecarreguem o processamento de dados de privacidade. Diante deste problema, este trabalho apresenta o desenvolvimento de uma solução baseada em ontologias para a privacidade em IoT, composta por: i) IoT-Priv, uma ontologia de privacidade para a IoT, construída como uma camada leve sobre conceitos de IoT, importados de uma ontologia emergente, denominada IoT-Lite; e ii) IoT-PrivServ, um serviço de gerenciamento de privacidade, o qual fornece funcionalidades para os consumidores e/ou produtores que utilizam a ontologia IoT-Priv na modelagem de seus dados, abstraindo destes a complexidade de realizar tais tarefas. Como contribuições, os resultados da avaliação de IoT-Priv e IoT-PrivServ apontam que mantivemos a característica de leveza presente em IoT-Lite, a qual era um de nossos objetivos iniciais. Além disso, demonstramos que a IoT-Priv é expressiva e extensível, uma vez que seus conceitos possibilitam que cenários complexos sejam modelados, e caso seja necessário, os pontos de extensões incluídos na ontologia permitem que ela seja importada e estendida para atender necessidades mais específicas.

Palavras–chave

Internet das Coisas, Privacidade, Ontologia, Serviço de Gerenciamento, Web Semântica.

Abstract

Arruda, Mayke Ferreira. **An ontological model and a data management service for supporting privacy in the Internet of Things**. Goiânia, 2019. 146p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

In the Internet of Things (IoT) paradigm, real-world objects equipped with identification, detection, network, and processing capabilities communicate and consume services over the Internet to perform some task on behalf of users. Due to the growing popularization of devices with sensing capabilities and the consequent increase in data production from these devices, the literature states that the design of an ontology-based model is an essential starting point for addressing privacy risks in IoT, since the connected devices are increasingly able to monitor human activities. In addition, due to the complexity and dynamicity of IoT environments, we emphasize the need for privacy ontologies that combine expressive and extensible vocabulary but do not overload the processing of privacy data. Facing this problem, this work presents the development of an ontology-based solution for privacy in IoT, composed by: i) IoT-Priv, a privacy ontology for IoT, built as a light layer on IoT concepts imported from an emerging ontology, called IoT-Lite; and ii) IoT-PrivServ, a privacy management service, which provides functionalities for consumers and / or producers who use the IoT-Priv ontology in modeling their data, abstracting from them the complexity of perform such tasks. As contributions, the results of the evaluation of IoT-Priv and IoT-PrivServ indicate that we maintained the lightness characteristic present in IoT-Lite, which was one of our initial goals. In addition, we have demonstrated that IoT-Priv is expressive and extensible, since its concepts allow complex scenarios to be modeled, and if necessary, the extension points included in the ontology allow it to be imported and extended to meet more specific needs.

Keywords

Internet of Things, Privacy, Ontology, Management Service, Semantic Web.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Códigos de Programas

Lista de Siglas

API	*Application Programming Interface*
CMS	*Context Management Systems*
DL	*Description Logic*
GAPP	*Generally Accepted Privacy Principles*
IoT	*Internet of Things*
IRI	*Internationalized Resource Identifier*
JSON	*JavaScript Object Notation*
M2	*Machine-to-Machine*
M3	*Machine-to-Machine Measurement*
OWL	*Web Ontology Language*
PbD	*Privacy by Design*
QU	*Quantity Kinds and Units*
RDF	*Resource Description Framework*
RDFS	*Resource Description Framework Schema*
SN	*Sensor Network*
SSN	*Semantic Sensor Network*
SSO	*Stimulus-Sensor-Observation*
SPARQL	*Simple Protocol and RDF Query Language*
SQL	*Structured Query Language*
SWRL	*Semantic Web Rule Language*
Turtle	*Terse RDF Triple Language*
W3C	*World Wide Web Consortium*
WGS84	*World Geodetic System 1984*
WKT	*Well-Known-Text*
XML	*Extensible Markup Language*

CAPÍTULO 1

Introdução

Este capítulo está estruturado como segue: inicialmente, é contextualizado o tema de pesquisa e, a seguir, apresenta-se a motivação para a realização deste trabalho. Em sequência, descrevem-se o problema de pesquisa e os objetivos gerais e específicos deste trabalho. Por fim, são destacadas a metodologia utilizada e as principais contribuições deste trabalho.

1.1 Contextualização

> Elementos especializados de hardware e software, conectados por fios, ondas de rádio e de infravermelho, serão tão ubíquos que ninguém perceberá suas presenças. ([74], página 1, tradução própria)

Esta definição de *Computação Ubíqua* dada pelo cientista Mark Weiser vislumbrava que um dia a computação estaria tão difundida que as interações humano-computador poderiam se tornar transparentes [74]. Abowd et al. chamam a atenção para a popularização de interações naturais para humanos, entre elas, a utilização da fala, gestos, proximidade, ou até mesmo a presença no ambiente, substituindo a necessidade de dispositivos que requerem a entrada implícita de dados, tais como mouse ou teclado [3].

Dentre as vertentes da Computação Ubíqua, destaca-se a *Computação Sensível ao Contexto*[1] [2, 56], a qual foi descrita por Dey et al. [21] como a área que estuda os mecanismos para fornecimento e utilização de informações de contexto a fim de oferecer serviços e informações relevantes aos usuários e outras aplicações.

O termo sensível ao contexto foi utilizado pela primeira vez por Schilit and Theimer [57], desde então, outros trabalhos como os de Davies et al. [19] e Dey et al. [20] tentaram encontrar a melhor definição para o termo. Entretanto, as definições de *contexto* [2] e *sensibilidade a contexto* que são amplamente aceitas pela comunidade científica foram propostas por Abowd et al. [1]. Segundo os autores:

[1] do Inglês, *Context-aware*

[2] os termos *contexto* e *informação de contexto* no decorrer deste trabalho serão utilizados com o mesmo sentido, como sugerido no trabalho de Perera et al. [56]

> Contexto é qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, lugar ou objeto considerado relevante para a interação entre o usuário e a aplicação, incluindo o próprio usuário e a aplicação.
> ([1], página 3, tradução própria)

> Um sistema é sensível a contexto se ele usa o contexto para fornecer informações e / ou serviços relevantes para o usuário, onde a relevância é determinada em função da tarefa que será realizada pelo usuário.
> ([1], página 6, tradução própria)

As informações de contexto, em geral, tendem a serem submetidas a um ciclo de vida de contexto, que consiste nas etapas de aquisição, modelagem, raciocínio e disseminação de contexto [56], apresentado na Figura 1.1.

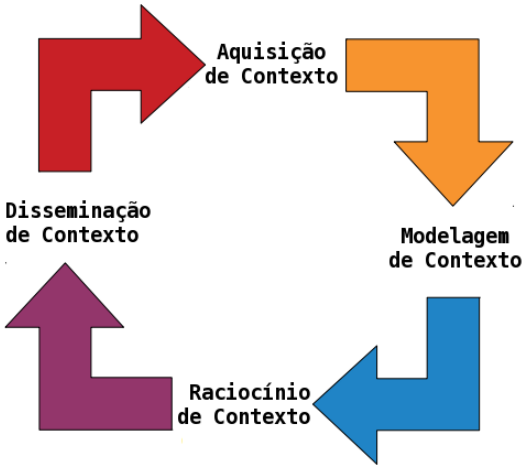

Figura 1.1: *Etapas essenciais em um ciclo de vida de contexto. Figura adaptada de Perera et al. [56], página 13.*

1. **Aquisição**: é a etapa onde as informações de contexto são obtidas diretamente dos provedores de contexto (p.ex.: sensores físicos ou virtuais).
2. **Modelagem**: nesta etapa, os dados coletados são representados por meio de técnicas de modelagem de contexto, as quais agregam semântica ao contexto. Segundo Perera et al., a etapa de modelagem possui um papel central no ciclo de vida de contexto [56].
3. **Raciocínio**: nesta etapa, são executadas técnicas de inferência de contexto com o objetivo de derivar contexto de mais alto nível em relação ao contexto entregue na etapa de modelagem.
4. **Disseminação**: nesta etapa, o ciclo de vida de contexto é encerrado e as informações geradas nas etapas de modelagem e raciocínio podem ser entregues para os consumidores (p.ex.: uma aplicação médica de monitoramento de sinais vitais)

No entanto, na última década o grande volume de pesquisas relacionadas à computação sensível ao contexto está convergindo para a *Internet das Coisas* (IoT)[3], a qual representa uma evolução da Internet atual, que conecta pessoas, computadores e dispositivos móveis para uma Internet onde uma infinidade de objetos presentes ao nosso redor poderão se conectar à rede e serem capazes de comunicar-se através da Internet, formando assim uma Internet das Coisas [56]. A Figura 1.2 representa esta evolução.

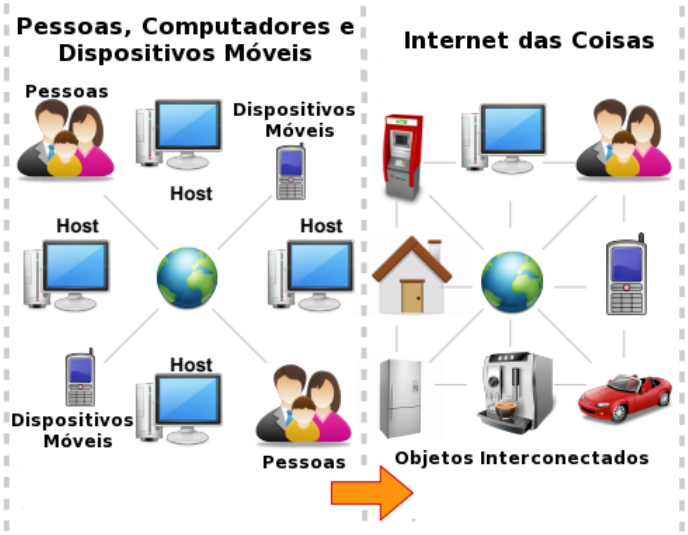

Figura 1.2: *Evolução da Internet atual para a Internet das Coisas. Figura adaptada de Perera et al. (2014) [56], página 3.*

1.2 Motivação

No paradigma da Internet das Coisas (IoT), objetos do mundo real equipados com recursos de identificação, detecção, rede e processamento comunicam-se e consomem serviços pela Internet para executar alguma tarefa em nome dos usuários [7].

Segundo Miorandi et al. [47] e Perera et al. [56], a privacidade é uma das maiores preocupações na IoT, pois espera-se que os sensores coletem e disponibilizem um grande volume de informações, as quais estão diretamente relacionadas aos usuários, por exemplo, a localização, preferências, dados do calendário, rotas para casa, ou até mesmo informações médicas restritas. Diversos autores concordam que a privacidade é um requisito primordial na IoT, mas além disso, também é considerada como um dos grandes desafios e tendências de pesquisa na IoT, como é relatado pelos autores Atzori et al. [7], Miorandi et al. [47], Gubi et al. [32], Borgia [12], Ziegeldorf et al. [77] e Yaqoob et al. [75].

[3] do Inglês, ***Internet of Things***

Como exemplo de um cenário de IoT, considere um lar de idosos em que roupas inteligentes equipadas com tecnologia de sensoriamento monitoram o paradeiro, os sinais vitais e o nível de socialização de cada habitante idoso. Essas roupas são capazes de comunicar-se com os *smartphones* dos profissionais de saúde locais para informá-los sobre a saúde e o estado social dos idosos.

Uma das preocupações mais desafiadoras dos cenários de IoT, como o lar de idosos, é a proteção da privacidade das pessoas. Esses profissionais de saúde podem acessar dados de pacientes de fora do lar de idosos? Por quanto tempo seus dispositivos poderão acessar dados de pacientes? Como se pode garantir que o dispositivo de alguém não autorizado tenha acesso indevido aos dados do paciente? Estas são apenas algumas ameaças potenciais significativas à privacidade dos usuários que os esforços de pesquisa em IoT têm abordado recentemente.

1.3 Problema

O design do modelo de privacidade tem sido cada vez mais considerado como um ponto de partida crucial para uma solução prática para lidar com riscos de privacidade. Além disso, na literatura tem havido um consenso de que as ontologias, como o formalismo de modelagem de conhecimento, são essenciais para se alcançar um modelo de privacidade interpretável por máquina, como é relatado nos trabalhos de Garcia et al. [27], Foukia et al. [26] e Perera et al. [55].

Perera [55] ainda ressalta que os atuais modelos de privacidade baseados em ontologias incorporam requisitos básicos, tais como as informações e seus proprietários e/ou consumidores, o período de retenção destas informações, propósito de acesso, coleta de dados, entre outros. No entanto, apenas estes requisitos podem não serem suficientes para atender a demanda dos sistemas e aplicações nos mais diversos cenários.

No entanto, como os sistemas IoT podem monitorar e interpretar as atividades e os comportamentos dos usuários 24 horas por dia, 7 dias por semana, por meio de vários dispositivos conectados, identificamos um *gap* de pesquisa relacionado o quanto um modelo ontológico de privacidade pode sobrecarregar o processo de interpretação e/ou processamento do conhecimento sobre privacidade em ambientes de IoT. A partir do exposto é apresentado o problema de pesquisa deste trabalho:

> **Problema:** *como o design de modelos de privacidade baseados em ontologias para IoT podem equilibrar expressividade, extensibilidade e a leveza?* Onde:
>
> - Expressividade está diretamente ligada aos construtores utilizados na construção de uma ontologia [41]. Quanto mais construtores forem utili-

zados, mais expressiva é uma ontologia, e consequentemente, inferências mais complexas podem ser realizadas.

- Extensibilidade refere-se a possibilidade de que outras ontologias sejam agregadas à *IoT-Priv*, reutilizando seu conhecimento e estendendo o que não é detalhado por ela, aumentando assim a sua expressividade e possibilitando que a *IoT-Priv* se adéque a requisitos e necessidades específicas de domínio [34]. P.ex.: agregando ontologias que descrevem propósitos de acesso ou técnicas de criptografias à *IoT-Priv*.
- Leveza refere-se ao comportamento temporal necessário para processar dados representados com um determinado modelo [11]. Vale ressaltar que há uma relação direta entre expressividade e leveza, isto é, quanto mais expressivo um modelo é, menos leve ele tende a ser.

1.4 Objetivos

O objetivo geral deste trabalho consiste em elaborar um *modelo ontológico* para a Internet das Coisas, que seja *expressivo*, *extensível* e *leve* para apoiar o desenvolvimento de aplicações para a IoT.

Neste intuito, os objetivos específicos da pesquisa são:

1. Elaborar um modelo ontológico de privacidade, chamado *IoT-Priv*, que inclua preferências e políticas de acesso relacionadas aos dados de cada entidade na Internet das Coisas.
2. Desenvolver um serviço semântico de gerenciamento de privacidade, chamado *IoT-PrivServ*.
3. Integrar o modelo ontológico ao serviço de privacidade.
4. Validar e avaliar o modelo e o serviço propostos a partir de estudos de caso, avaliando se os requisitos de expressividade, extensibilidade e leveza foram atingidos.

1.5 Materiais e Métodos

Esta seção apresenta um resumo da metodologia utilizada para alcançar os objetivos desta pesquisa.

i. **Revisão bibliográfica**: após identificar a privacidade na IoT como um problema, foi conduzida uma revisão bibliográfica com o objetivo de identificar e analisar as pesquisas que exploravam o tema.
ii. **Definição do escopo**: através da revisão bibliográfica foi identificado o problema de pesquisa apontado na Seção 1.3.

iii. **Definição dos objetivos**: uma vez que o escopo da proposta foi decidido, os objetivos gerais e específicos da pesquisa foram determinados, como descrito na Seção 1.4.

iv. **Definição da abordagem de pesquisa e materiais necessários**: uma vez que o objetivo foi decidido, foram definidas as metodologias, tecnologias, ferramentas e padrões para a realização desta pesquisa.

- A modelagem baseada em *ontologias* foi a técnica escolhida para especificar e modelar a privacidade do usuário na IoT, pois possibilitam que o contexto possa ser representado com um alto grau de *formalismo* e *expressividade*. Além disso, as ontologias podem ser *extensíveis*, possibilitando que o modelo de contexto possa ser importado e incrementado, caso seja necessário [56].
- Para o desenvolvimento da *IoT-Priv* será adotada a Metodologia de Desenvolvimento de Ontologias 101 [50], pois fornece uma abordagem iterativa, incremental e bem documentada para o desenvolvimento de ontologias.
- Como é recomendado por Noy e McGuinness [50], a ferramenta Protégé[4] foi adotada como editor de ontologias, pois fornece vários recursos úteis para gerenciar e projetar ontologias, como ferramentas de visualização e extração de métricas [50].
- Para representar as características básicas e necessárias para a IoT, tais como suporte a redes de sensores, suporte a localização e alta expressividade, a ontologia *IoT-Lite* [11] foi adotada como base para representação de contexto, pois possibilita que cenários IoT sejam representados com um alto grau de expressividade. Além disso, a IoT-Lite ainda conta com duas outras características desejáveis para ambientes IoT, sendo elas a extensibilidade e leveza de processamento.
- Foram adotados padrões já consolidados da Web Semântica, entre eles, as linguagens de desenvolvimento de ontologias *Resource Description Framework Schema* (RDFs) e *Ontology Web Language* (OWL), além da linguagem de consulta *Simple Protocol and RDF Query Language* (SPARQL) [76].

v. **Definição dos requisitos** : foram identificados na revisão bibliográfica um total de 27 requisitos de privacidade, a serem modelados na ontologia *IoT-Priv*.

vi. **Desenvolvimento, validação e avaliação da ontologia *IoT-Priv***

- Os termos técnicos relacionados à privacidade utilizados nestas etapas estão de acordo com os Princípios de Privacidade Geralmente Aceitos (GAPP)[5] [5].
- Para a validação, a IoT-Priv foi aplicada em dois cenários de domínios distintos, a saber: compartilhamento de fotos [28] e assistência médica domiciliar [4].

[4] Disponível para download em http://protege.stanford.edu.
[5] do Inglês, ***G**enerally **A**ccepted **P**rivacy **P**rinciples*

- A avaliação foi realizada comparando as propriedades estruturais e lógicas das ontologias *IoT-Priv* e IoT-Lite, utilizando o *plugin Ontology Metrics* da ferramenta *Protégé*.

vii. **Desenvolvimento e avaliação do serviço *IoT-PrivServ***:

- Nesta etapa foi desenvolvido o Serviço de Gerenciamento de Privacidade, o qual disponibiliza funções baseadas na ontologia *IoT-Priv*, tais como a validação e armazenamento de modelos de contexto, inferência de novos fatos a partir dos já instanciados e filtros temporais e espaciais para consultas.
- A avaliação do serviço foi realizada com relação ao comportamento temporal do serviço, onde foram analisados os tempos de respostas para executar as funcionalidades por ele fornecidas.

1.6 Estrutura do Documento

O restante deste documento está estruturado como segue: o Capítulo 2 apresenta os fundamentos teóricos e tecnologias que norteiam esta pesquisa, tais como Internet das Coisas, Privacidade, Web Semântica e e Ontologias. O Capítulo 3 descreve a ontologia *IoT-Priv*, detalhando os requisitos para construção da mesma, o processo de desenvolvimento da ontologia, além de cenários de aplicação e uma avaliação da ontologia proposta. A seguir, o Capítulo 4 detalha o serviço de gerenciamento de privacidade *IoT-PrivServ*, descrevendo sua arquitetura e funcionalidades fornecidas, e consequentemente, uma avaliação do serviço proposto. Por sua vez, o Capítulo 5 discorre sobre os trabalhos relacionados a esta proposta. Por fim, o Capítulo 6 discute as conclusões e trabalhos futuros.

CAPÍTULO 2

Fundamentação Teórica e Tecnologias

Este capítulo discute conceitos que fundamentam esta pesquisa, que incluem Internet das Coisas e Privacidade, bem como aspectos teóricos e tecnologias associados à Web Semântica. Em suas considerações finais, este capítulo relaciona o objetivo desta pesquisa a toda essa bagagem teórica e tecnológica apresentada.

2.1 Internet das Coisas

A Internet das Coisas (IoT) é um paradigma emergente na computação, o qual representa uma evolução do mundo atual, no qual os objetos presentes no ambiente poderão se conectar à rede e serão capazes de comunicar-se através da Internet, ou diretamente entre si, através da comunicação Máquina-a-Máquina (M2M). Uma definição de IoT bastante aceita pela comunidade científica é a de Vermesan et al. [63], em que:

> a Internet das Coisas possibilita que pessoas e "coisas" sejam conectadas a qualquer hora, em qualquer lugar, a qualquer "coisa" e a qualquer pessoa, idealmente, utilizando qualquer rede.
>
> ([63], pág 4, tradução própria)

Ao analisar esta definição, percebe-se que para IoT ser amplamente utilizada, são necessárias pelo menos três características essenciais:

1. **Comunicação**: os objetos que compõem a IoT devem ter a capacidade de comunicarem entre si, ou com os sistemas que os controlam, idealmente, utilizando mecanismos de comunicação sem fio;
2. **Identificação**: os objetos precisam ser reconhecidos por uma identidade digital única; e,
3. **Interação**: os objetos precisam ser capazes de interagir com qualquer ambiente e/ou outros objetos, através de recursos de detecção e atuação.

No entanto, a IoT é um paradigma complexo, e consequentemente abrange outras características, as quais usualmente são tratadas como problemas de pesquisa. Segundo os autores Perera et al. [56], Ngu et al. [49] e Yaqoob et al. [75], entre estas características, destacam-se:

- **Heterogeneidade de Dispositivos**: a IoT é composta por uma infinidade de dispositivos, os quais possuem especificações e características distintas, e consequentemente, as soluções IoT devem ser capazes de suportar esta grande diversidade de dispositivos. Por exemplo, um sensor de temperatura corporal pode usar protocolos e terminologias diferentes de um atuador utilizado para controlar o brilho de luz.
- **Interoperabilidade semântica**: para possibilitar a comunicação entre dispositivos heterogêneos, a interoperabilidade semântica se faz necessária, para isso, todas as entidades devem ter o mesmo entendimento do significado dos dados, através de um vocabulário único e consensual [45]. Sendo assim, as ontologias se mostram uma solução plausível para possibilitarem a interoperabilidade semântica, as quais são apresentadas na Seção 2.3.
- **Grande escala**: é previsto que até o ano de 2020 existirão mais de 50 bilhões de dispositivos conectados à Internet [22], totalizando uma média de aproximadamente 7 dispositivos conectados por pessoa, e a tendência é que estes números continuem a crescer ainda mais. Semelhante ao número de objetos, o número de interações entre esses objetos, bem como a a quantidade de dados produzidos por eles também tendem a aumentarem significativamente. Sendo assim, a IoT precisa de soluções que sejam capazes de suportar este grande volume de informações e interações, ou seja, soluções escaláveis.
- **Inteligência**: as aplicações da IoT são capazes de gerar conhecimento através dos dados brutos. A transformação destes dados brutos em conhecimento é feita através da modelagem e do raciocínio sobre o contexto, o que é um procedimento complexo, e consequentemente, caro computacionalmente.
- **Privacidade**: a IoT necessita de conectividade e acesso global, o que significa que qualquer entidade (pessoa ou dispositivo inteligente) pode ser capaz de acessar os dados produzidos por outras entidades, a qualquer momento e de qualquer lugar. Embora exista um enorme potencial na IoT, a privacidade representa um componente crítico para possibilitar a ampla aceitação de tecnologias e aplicativos da IoT, pois espera-se que as aplicações e/ou sensores coletem informações sobre as atividades diárias de entidades, as quais podem resultar em informações confidenciais, logo devem ser devidamente protegidas.

2.2 Privacidade

Com a popularização de sistemas que coletam informações e realizam a prestação de serviços, o valor da informação coletada e a necessidade de gerenciá-la com responsabilidade cresceram drasticamente [15].

A rápida inovação e a crescente complexidade dos sistemas apresentam grandes desafios para a privacidade informacional. Neste sentido, os usuários tem demonstrado uma preocupação significativa com a sua privacidade, tais como o roubo de identidade e a divulgação inadequada de suas informações pessoais, especialmente registros confidenciais, tais como dados financeiros ou de saúde. Sendo assim, os usuários esperam que sua privacidade seja respeitada e que suas informações pessoais sejam protegidas pelas organizações com as quais fazem negócios [16].

2.2.1 Conceitos Básicos

A privacidade é definida nos Princípios de Privacidade Geralmente Aceitos (GAPP) como "os direitos e obrigações de indivíduos e organizações em relação à coleta, uso, retenção, divulgação e descarte de **informações pessoais**" [15].

O termo informação pessoal refere-se a qualquer informação que possa ser vinculada a um indivíduo ou usada para identificá-lo, direta ou indiretamente, por exemplo: nome, endereço residencial, endereço de *e-mail*, número de identificação (por exemplo, CPF), características físicas ou até mesmo o histórico de compras.

No entanto, algumas informações pessoais são consideradas **confidenciais**, e consequentemente, exigem um nível extra de proteção, como por exemplo, exigir o consentimento explícito para a coleta e uso destas informações, isto é, a entidade que terá suas informações confidenciais coletadas deve, explicitamente, permitir e concordar com esta coleta antes que ela aconteça [15]. São exemplos de informações pessoais confidenciais: condições médicas ou de saúde, localização, condenações criminais, opiniões políticas, opção sexual, religião, entre outras.

Caso informações pessoais precisem ser divulgadas, em geral, devem ser **anonimizadas**. Este processo consiste na remoção ou ofuscação de qualquer informação secundária que possa ser usada para identificar um indivíduo específico, de modo que a identidade do indivíduo em questão não poderá ser determinada a partir das informações divulgadas. Por exemplo, suponha que informações médicas de um determinado paciente precisem ser divulgadas por algum motivo, para isso, as demais informações relacionadas a este paciente devem ser anonimizadas, tais como Nome, RG, CPF, cidade, entre outras, de modo que sua identidade não possa ser determinada.

2.2.2 *Framework* PbD

Cavoukian argumenta que a privacidade, por se tratar de um problema complexo e de grande importância para os envolvidos, deve ser abordada em todas as etapas do desenvolvimento de um sistema, sendo incorporada a modelos, padrões, protocolos e aos processos de um sistema [16].

Neste sentido, o *framework* conceitual PbD[1] foi desenvolvido, estabelecendo uma estrutura universal, organizada em 7 diretrizes, as quais podem ser aplicadas a tecnologias específicas, operações de negócios e até mesmo a modelos de governança [16]. As diretrizes do *framework* supracitado são descritas a seguir:

a. **Proativo e não Reativo**
   O *framework* PbD é caracterizado por medidas proativas e não reativas, pois tende a antecipar e evitar eventos invasivos de privacidade antes que eles aconteçam. O PbD não espera que os riscos de privacidade se materializem, nem oferece remédios para resolver infrações de privacidade uma vez que tenham ocorrido - ela visa impedir, de alguma maneira, que estas infrações ocorram. Em resumo, *PbD* trata as questões de privacidade antes de que qualquer fato tenha acontecido, e não depois.

b. **Privacidade como Padrão**
   O *framework* PbD procura oferecer o máximo grau de privacidade para sistemas, logo, recomenda que os seguintes princípios de privacidade sejam seguidos:

   - As **finalidades** para as quais as informações pessoais são coletadas, usadas, retidas e divulgadas devem ser comunicadas ao proprietário destas informações. As finalidades especificadas devem ser claras, limitadas e relevantes para as circunstâncias.
   - A **coleta** de informações pessoais deve ser justa, legal e limitada ao necessário para os fins especificados.
   - O **uso**, a **retenção** e a **divulgação** de informações pessoais devem ser limitados aos objetivos relevantes identificados para o indivíduo, para os quais foi consentido, exceto quando exigido por lei. Além disso, as informações pessoais devem ser retidas apenas pelo tempo necessário para cumprir as finalidades declaradas e, em seguida, destruídas com segurança.

c. **Privacidade incorporada ao design**
   O *framework* PbD sugere que a privacidade seja incorporada no *design* e na arquitetura de sistemas de TI. O resultado esperado é que a privacidade se torne um componente essencial da funcionalidade principal que está sendo entregue.

[1] do Inglês, ***P**rivacy **b**y **D**esign*

d. **Funcionalidade Total**
O *framework* PbD sugere que ao incorporar a privacidade em uma determinada tecnologia, processo ou sistema, isso deve ser feito de maneira que a funcionalidade total não seja prejudicada e, na medida do possível, todos os requisitos sejam otimizados.

e. **Proteção em Todo o Ciclo de Vida**
O *framework* PbD sugere que a privacidade seja incorporada em todo o ciclo de vida das informações, desde a coleta, modelagem, processamento, armazenamento, até a disseminação e destruição destas informações.

f. **Visibilidade e Transparência**
São princípios essenciais para estabelecer responsabilidade e confiança. Para isso, o *framework* PbD sugere que sejam aplicados os seguintes princípios:

- **Responsabilização** - A responsabilidade por todas as políticas e procedimentos relacionados à privacidade deve ser documentada e comunicada, conforme apropriado, e atribuída a um indivíduo específico.
- **Abertura** - Transparência é fundamental para a prestação de contas, neste sentido, informações sobre as políticas e práticas relacionadas ao gerenciamento de informações pessoais devem ser prontamente disponibilizadas aos indivíduos, se assim forem requisitadas.
- **Conformidade** - Mecanismos de reclamação e reparação devem ser estabelecidos, e informações devem ser comunicadas sobre eles aos indivíduos, incluindo como acessar o próximo nível de apelação.

g. **Respeito pela Privacidade do Usuário**
Permitir que os titulares de dados desempenhem um papel ativo na gestão dos seus próprios dados pode ser a verificação mais eficaz contra abusos e usos indevidos de privacidade e dados pessoais, para isso, sugere-se a utilização dos seguintes princípios:

- O consentimento do indivíduo é necessário para a coleta, uso ou divulgação de informações pessoais, exceto quando permitido por lei. O consentimento pode ser retirado em uma data posterior.
- Indivíduos terão acesso às suas informações pessoais e serão informados sobre seus usos e divulgações. Os indivíduos devem ser capazes de contestar a exatidão e integridade das informações e alterá-las conforme apropriado.

2.2.3 Privacidade em IoT

A privacidade é um assunto complexo para ser tratado, pois é conceitualmente denso e sujeito a variações culturais. Quando abordamos o tema no contexto de Internet das Coisas, o desafio é ainda maior, graças à grande variedade de dispositivos, os quais podem ser, ao mesmo tempo, produtores e consumidores de dados. Além disso, graças ao alcance da Internet, estes dispositivos podem estar localizados fisicamente em regiões que possuem diferentes legislações sobre a privacidade, o que dificulta ainda mais este processo.

Segundo Miorandi et al. [47] e Perera et al. [56], a privacidade é uma das maiores preocupações na IoT, pois espera-se que os dispositivos com capacidade de sensoriamento coletem e disponibilizem um grande volume de informações, as quais estão diretamente relacionadas as entidades sensoreadas. Miorandi et al. [47] revisaram a literatura, e apontaram as principais desafios associados à segurança em IoT, organizados em três principais grupos: confidencialidade dos dados, confiança e **privacidade**, como ilustra a Figura 2.1.

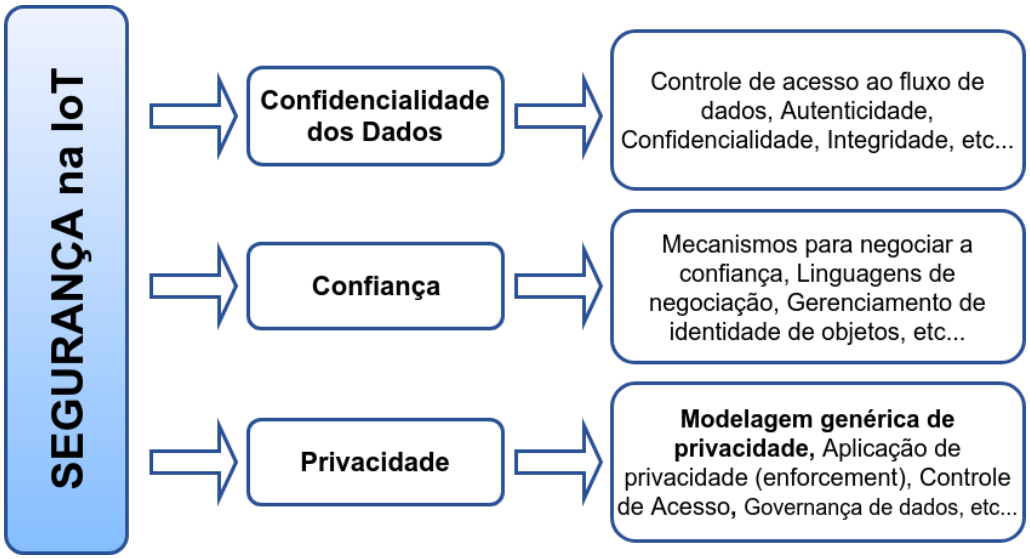

Figura 2.1: *Desafios relacionados à segurança em IoT. Figura adaptada de [47], página 12.*

Dentre os principais desafios associados à privacidade em IoT apontados por Miorandi et al. [47], destaca-se a definição de modelos genéricos de privacidade, isto porque um modelo genérico capaz de representar conceitos associados à privacidade em IoT, bem como seus relacionamento, beneficiariam qualquer tipo de aplicação que viesse a ser construída. Além disso, as implementações devem incluir mecanismos de aplicação de privacidade capazes de lidar com a escala e com a natureza dinâmica dos cenários de IoT.

Resumidamente, os principais desafios de pesquisa relacionados à privacidade em IoT, conforme é relatado na Figura 2.1, incluem a definição de um modelo genérico de privacidade para a IoT, o desenvolvimento de técnicas de aplicação da privacidade, bem como o desenvolvimento de técnicas de controle de acesso.

Além disso, em virtude da complexidade associada à IoT, a literatura tem recomendado a adoção do *framework* PbD [36]. Resumidamente, este *framework* aponta diretrizes e sugere que a privacidade deva ser tratada em todas as etapas na construção de um sistema IoT, tais como o planejamento, modelagem e desenvolvimento, de modo que a privacidade seja uma funcionalidade agregada em todas as etapas do ciclo de vida de contexto na IoT. Vale ressaltar que a etapa de modelagem do ciclo de vida de contexto é o enfoque deste trabalho, por isso este tópico será apresentado em maiores detalhes.

Com respeito à modelagem de privacidade em IoT, Perera et al. [55] revisaram a literatura e analisaram diversos trabalhos que realizam a modelagem de privacidade em IoT. Como resultados, os autores ressaltam que uma das tendências observadas é a crescente adoção de ontologias na modelagem do conhecimento de privacidade neste contexto. Uma outra vantagem apontada pelos autores, é que a utilização de ontologias permite que as políticas de privacidade sejam definidas tanto no nível de dados (através de instâncias específicas para cada caso), quanto no nível de classe (através de conceitos genéricos, comuns a qualquer domínio) [55].

Por fim, os autores ainda argumentam que as ontologias são o método mais apropriado de modelar o conhecimento sobre privacidade em IoT, isto devido à sua capacidade de permitir a modelagem de conceitos, bem como de relacionamentos entre conceitos. No entanto, a grande maioria dos trabalhos que utilizavam ontologias como técnica de modelagem eram específicos de domínio (p. ex.: saúde, transporte ou *e-commerce*) [55], o que impossibilita estas abordagens de serem amplamente aplicadas no cenário da IoT, corroborando a necessidade de modelos genéricos de privacidade em IoT, como foi apontado anteriormente por Miorandi et al. [47].

2.3 Ontologias

Segundo Gruber, "uma ontologia é uma *especificação formal* de uma conceitualização" [31]. Borst, por sua vez, complementou esta definição, afirmando que: "uma ontologia é uma especificação *explicita* e *formal* de uma *conceitualização compartilhada*" [13]. O termo *conceitualização* significa que as ontologias devem refletir um modelo abstrato de algum cenário ou fenômeno, bem como os conceitos relevantes desse cenário ou fenômenos. Consequentemente, o termo *explícita* enfatiza que os tipos de conceitos e as restrições utilizadas devem ser explicitamente definidos. Além disso, o termo *formal* ressalta que a ontologia deve ser representada sob uma linguagem legível por

sistemas de software. Por fim, o termo *compartilhada* reflete a noção de que uma ontologia representa o conhecimento consensual de um domínio [13].

Além disso, Gomés-Pérez [29] et al. ressaltam que as ontologias podem ser classificadas em dois grupos:

- Específicas de domínio (ou verticais): são aquelas que definem a semântica de conceitos e relações voltados para um domínio específico, p.ex. uma ontologia de vinhos.
- Independentes de domínio (ou horizontais): são aquelas cujo a semântica dos conceitos e relações independe de um domínio particular, p.ex. uma ontologia de privacidade.

Por fim, Bulcão Neto [14] aponta os benefícios associados a utilização de ontologias, dentre eles, destacam-se: prevenir diferentes interpretações; o conhecimento se torna independente da implementação; permite reusar e estender conhecimento; e a inferência de novos fatos, a partir dos já instanciados.

2.3.1 Metodologia para a Construção de Ontologias

Segundo Noy e McGuinness [50], não há uma maneira correta para modelar um domínio ou cenário específico, há sempre alternativas viáveis, e a melhor solução depende do que o projetista tem em mente e das extensões que ele é capaz de prever.

Consequentemente, não há um método ideal para construção de ontologias, no entanto, existem metodologias, as quais visam sistematizar a especificação e a construção da ontologia em questão [48]. Dentre as metodologias existentes, destaca-se a *Metodologia 101*, apresentada por Noy e McGuiness [50], a qual é dividida em 7 etapas, as quais são detalhadas a seguir:

1. **Determinar o domínio e o escopo da ontologia**: o desenvolvimento de uma ontologia deve ser iniciado pela definição de seu domínio e escopo. Isto é, respondendo a perguntas básicas, tais como:
   - Qual é o domínio que a ontologia cobrirá?
   - Para que a ontologia será utilizada?
   - Para quais tipos de perguntas as informações presentes na ontologia devem fornecer respostas? Isto é, quais questões de competência a ontologia deve ser capaz de responder.
2. **Considerar a possibilidade de reutilizar ontologias existentes**: consiste em considerar o que outra pessoa fez e verificar se é possível refinar e ampliar as fontes existentes para nosso domínio e tarefas específicas. Muitas ontologias já estão

disponíveis em formato eletrônico e podem ser importadas para um ambiente de desenvolvimento de ontologias, e consequentemente, reutilizadas e estendidas.

3. **Enumerar termos importantes na ontologia**: consiste em escrever uma lista de todos os termos que gostaríamos de fazer declarações, respondendo as seguintes perguntas:
   - Quais são os termos que pertencem a este domínio?
   - Quais propriedades esses termos possuem?
   - Quais são as relações e/ou relacionamentos entre estes termos?

   Por exemplo, uma lista de termos relacionados ao domínio de vinhos incluem *vinho*, *vinho branco*, *uva*, *vinícola*, *cor do vinho*, *sabor* e *teor de açúcar*.
4. **Definir as classes e a hierarquia de classes**: com a lista de termos enumerados, esta etapa consiste em identificar e especificar as classes, bem como a hierarquia das classes na ontologia.
5. **Definir as propriedades das classes**: apenas classes, em geral, não são capazes de fornecer informações suficientes para responder às questões de competência identificadas na etapa (1). Neste sentido, esta etapa consiste em especificar os conceitos que representam as relações entre essas classes.
6. **Definir as facetas das propriedades**: as facetas são, basicamente, restrições sobre as propriedades, as quais são divididas em três tipos:
   - **Domínio e imagem**: p.ex. domínio e imagem de *possuiCor* são, respectivamente, as classes *Vinho* e *Cor*, ou seja, a propriedade *possuiCor* liga um *Vinho* a sua respectiva *Cor*.
   - **Tipo de valor**: as propriedades que ligam uma classe a um valor literal devem restringir os valores literais que são aceitos. P.ex.: o valor da propriedade *nomeVinho* é uma *String*, enquanto o valor da propriedade *quantidadeAçúcar* é um *Decimal Positivo*.
   - **Cardinalidade**: a cardinalidade associada à uma propriedade também pode ser especificada.
     P.ex.: a propriedade *localizaçãoAdega* possui cardinalidade <*min=1*>, especificando que um determinado vinho é produzido por pelo menos uma adega, no entanto, podem haver outras adegas produzindo o mesmo vinho, enquanto a propriedade *possuiCor* deve possuir cardinalidade <*exactly=1*>, especificando que um vinho está associado à exatamente uma cor.

7. **Criar instâncias e Avaliar a Ontologia**: por fim, a ultima etapa da metodologia consiste em criar instâncias individuais das classes especificadas. Isso permite que sejam identificados erros na modelagem, ou até mesmo a necessidade de outras classes ou propriedades ainda não especificados. Neste caso, pode ser necessário voltar a etapa (4) para refinar a ontologia.

2.3.2 Lógica de Descrição

A lógica de descrição é uma linguagem formal de representação de conhecimento. No contexto da Web Semântica, a lógica de descrição é importante pois fornece um formalismo lógico para as ontologias, por exemplo, a Linguagem de Ontologia da Web (OWL, mais detalhes na Seção 2.4) é baseada em lógica de descrição [8].

A Lógica de Descrição é caracterizada pelos construtores que ela fornece, e consequentemente, lógicas mais expressivas podem ser obtidas através da extensão de lógicas menos expressivas. Por exemplo, a Lógica de Descrição $\mathcal{ALCQ}$ estende a linguagem base $\mathcal{AL}$ adicionando a negação de conceitos ($\mathcal{C}$) e restrições de cardinalidade qualificadas ($\mathcal{Q}$). Mais detalhes a respeito dos construtores da lógica de descrição são apresentados na Tabela 2.1.

Tabela 2.1: *Construtores da Lógica de Descrição e suas respectivas descrições. Adaptada de Kepler et al. [40]*

Construtor	Descrição
$\mathcal{AL}$	Esta é a linguagem base da lógica de descrição
$\mathcal{F}$	Propriedades funcionais
$\mathcal{U}$	Uniões de classes
$\mathcal{C}$	Negação de conceitos de classes
$\mathcal{H}$	Utilização de hierarquia de propriedades
$\mathcal{R}$	Propriedades complexas, tais como reflexividade ou composição; Também vale para as classes disjuntas.
$\mathcal{O}$	Restrições de valores (como indivíduos ou literais) que uma propriedade pode ter
$\mathcal{I}$	Propriedades inversas
$\mathcal{N}$	Restrições de cardinalidade
$\mathcal{Q}$	Restrições de cardinalidade qualificadas, por exemplo, cardinalidades máxima, mínima e exata.
$(\mathcal{D})$	Uso de propriedades cujos valores são literais
$\mathcal{S}$	Representa a abreviação de $\mathcal{ALC}$ com uso de propriedades transitivas.

Vale ressaltar que a lógica de descrição mínima possível é $\mathcal{AL}$, representando apenas a linguagem base, enquanto a máxima é $\mathcal{SROIQ}(\mathcal{D})$. Para fins de comparação, enquanto $\mathcal{AL}$ tem complexidade computacional na classe *PSPACE-complete*, $\mathcal{SROIQ}(\mathcal{D})$ é da classe *NExpTime-complete*, conforme é apontado pela ferramenta da Universidade de Manchester[2], que associa uma lógica de descrição a uma complexidade computacional.

2.3.3 Avaliação de Ontologias

De acordo com Slimani [58], o processo de avaliação de uma ontologia pode ser baseado em diferentes pontos de vista, entre eles por exemplo, a qualidade da ontologia projetada (ponto de vista técnico) e sua usabilidade no mundo real (visão prática). Adicionalmente, é possível avaliar uma ontologia baseada em modelos matemáticos (ponto de vista matemático).

Neste sentido, Gruber [30] aponta critérios para a avaliar a qualidade e usabilidade de ontologias, nos quais os seguintes pontos devem ser observados e avaliados:

- **Clareza**: uma ontologia deve comunicar efetivamente o significado pretendido dos termos definidos, sendo assim, todas as definições expressas na ontologia devem ser descritas de maneira objetivas e documentadas com linguagem natural.
- **Coerência**: uma ontologia deve ser coerente, isto é, deve ser consistentes com as definições do cenário que modela. A coerência também deve se aplicar aos conceitos definidos informalmente, como aqueles descritos em documentação e exemplos em linguagem natural.
- **Extensibilidade**: uma ontologia deve oferecer uma base conceitual para uma série de tarefas antecipadas, e a representação deve ser elaborada de modo que se possa ser estendida e especializada, se necessário. Em outras palavras, os usuários devem ser capazes de definir novos termos para usos especiais baseados no vocabulário existente, de uma forma que não exija a revisão das definições existentes.
- **Compromisso ontológico**: uma ontologia deve fazer o menor número possível de afirmações sobre o mundo que está sendo modelado, permitindo que as partes comprometidas com a ontologia se especializem e instanciem a ontologia conforme necessário. Isto é, a modelagem deve fornecer conceitos e relações para que os indivíduos possam ser instanciados conforme a necessidade.

Além destas, existem abordagens matemáticas para avaliar a qualidade de ontologias, como a abordagem *OntoQA* proposta por Tartir et al. [59], cujo objetivo principal é fornecer métricas para avaliar o esquema ontológico como um todo, utilizando

[2] disponível para acesso em http://www.cs.man.ac.uk/~ezolin/dl/

métricas como a riqueza de relacionamento, riqueza de atributos, proporção de atributos por classes.

Por fim, uma outra métrica para avaliação de ontologias é a **expressividade da lógica de descrição**, utilizada como indicativo da dificuldade de se processar um conjunto de dados representando com uma determinada ontologia, como processar consultas ou realizar inferências [40].

2.4 Tecnologias

Após a Seção anterior ter discutido sobre um dos pilares da Web Semântica, as ontologias, esta Seção descreve as tecnologias e padrões que compõem a arquitetura da Web Semântica, e são parte fundamental deste conceito. Dentre o que será discutido ao longo desta Seção, incluem os padrões para intercâmbio de dados (RDF), construção de ontologias (OWL) e consulta a triplas RDF (SPARQL).

2.4.1 *Resource Description Framework* (RDF)

RDF é um padrão para intercâmbio de dados, construído para representar qualquer tipo de informação, de maneira universal, consensual e legível por máquinas [69]. O modelo de dados RDF é baseado em triplas, as quais são declarações constituídas por três elementos *(sujeito, predicado, objeto)*, e podem ser mapeadas diretamente para grafos dirigidos, nas quais:

- **sujeito**: representa uma IRI[3] de um *recurso* qualquer na web;
- **predicado**: representa uma IRI de uma *propriedade* de um recurso, ou o *relacionamento* entre dois recursos; Além disso, vale ressaltar que todos os predicados descritos em RDF são binários, ligando sempre um sujeito a um objeto.
- **objeto**: pode representar um *valor literal*, ou uma IRI de um *recurso*.

Por exemplo: se quisermos expressar o seguinte fato: “O carro de Augusto é Palio”, em RDF, este fato seria uma tripla, como ilustra a Figura 2.2. Na Figura, percebe-se que tanto Augusto quanto Palio não estão representados como simples Strings, isto é, em forma textual, mas sim, por identificadores únicos (*http://example.org/Augusto* e *http://example.org/fiat/Palio*, respectivamente). Além disso, as IRIs de Augusto e Palio estão interligadas por um predicado, que também é uma IRI *(http://example.org/possuiCarro)*, indicando que Augusto possui um carro Palio.

[3] IRI é uma padrão para identificação de recursos na Web, definido na RFC 3987 [25]

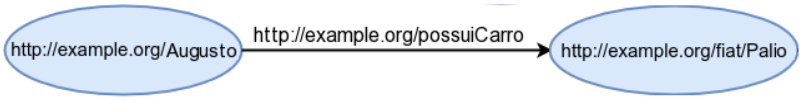

Figura 2.2: *Visualização em grafo de triplas RDF.*

No entanto, chama-se atenção também ao fato de que mesmo que grafos RDF sejam fáceis de visualizar, entender e programar, deve existir uma representação desses grafos para facilitar ainda mais o processamento e o intercâmbio de dados entre aplicações. Ou seja, necessita-se de sintaxes de serialização de dados RDF.

Dentre as várias sintaxes de serialização de triplas RDF, tais como N3 [65], N-Triples [70], RDF/XML [72] e JSON-LD [68], destaca-se a serialização Turtle (Terse RDF Triple Language) [71]. Turtle é uma sintaxe que permite que um grafo RDF seja representado sob uma forma textual compacta, simples e compatível com os padrões IRI e espaços de nomes XML. A sintaxe é direta e simples:

- Toda declaração é encerrada com um ponto final (.).
- A definição do espaço de nomes é realizada através da sintaxe:
  @prefix id: <namespace#> .
- As triplas são escritas em sequência, separadas apenas por um ou mais espaços em branco.
- Para várias triplas relacionadas a um mesmo sujeito, é possível descrever os dados em linhas diferentes, desde que: i) as linhas sejam finalizadas com ponto-e-vírgula, com exceção da última linha; ii) com exceção da primeira linha, o sujeito é omitido; e, iii) opcional, mas é recomendado que as demais linhas, com exceção da primeira sejam escritas com recuo à direita.

O Código 2.1 ilustra uma representação RDF serializa na sintaxe Turtle, o qual é detalhado a seguir.

Código 2.1 Exemplo de serialização RDF em *Turtle*.

```
1  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
3  @prefix xsd: <http://w3.org/2001/XMLSchema#> .
4  @prefix : <http://example.org/foaf-instance#> .
5
6  :pessoa1 a foaf:Person ;
7      foaf:name "Joao"^^xsd:string ;
8      foaf:age "30"^^xsd:positiveInteger;
9      foaf:knows :pessoa2.
10
11 :pessoa2 a foaf:Person.
```

- As linha de 1 a 4 associam um prefixo a um espaço de nomes, o qual é definido através de uma IRI.
- A linha 6 instancia a entidade *pessoa1* e atribui a ela o tipo *foaf:person*.
- A linha 7 define o nome da entidade, associando a string “Joao” a entidade *pessoa1*, através da propriedade *foaf:name*.
- A linha 8 define a idade da entidade, associando o número positivo inteiro “30” a entidade *pessoa1*, através da propriedade *foaf:age*.
- A linha 9 define um conhecido da entiade, associando a outra entidade *pessoa2* a entidade *pessoa1*, através da propriedade *foaf:knows*
- Por fim, a linha 11 instancia a entidade *pessoa2* e atribui a ela o tipo *foaf:person*.

2.4.2 *Ontology Web Language* (OWL)

OWL (*Web Ontology Language*) [66] é uma linguagem para construção de ontologias, a qual estende o padrão RDFS[4], incorporando construtores e propriedades que possibilitam a especificação de domínios mais complexos [35], e também deduzir novos conhecimentos a partir de fatos instanciados, o que é denominado *inferência ontológica*. A linguagem OWL fornece uma vasta gama de construtores e propriedades, a seguir, serão detalhados aqueles que foram utilizados na construção da ontologia *IoT-Priv*, proposta neste trabalho.

- ***owl:inverseOf*** – se há um relacionamento em uma direção, então há outro na direção inversa, como é exemplificado no Código 2.2, onde existem duas propriedades, *ex:temFilho* e *ex:temPai*, além disso, foi definido que a propriedade *ex:temFilho* é inversa da propriedade *ex:temPai*. A seguir, foi instanciado o fato de que (*:PessoA, ex:temFilho, :PessoaB*), como *ex:temFilho* é uma propriedade inversa de *ex:temPai*, a ontologia consegue inferir que (*:PessoaB, ex:temPai, :PessoaA*).

Código 2.2 Exemplo da propriedade *owl:inverseOf*

```
1 % Definição da propriedade:
2     ex:temFilho a    owl:Property.
3     ex:temPai   a    owl:Property.
4     ex:temFilho owl:inverseOf   ex:temPai.
5 % Instanciação de fatos:
6     :PessoaA    ex:temFilho   :PessoaB.
7 % Fato inferido:
8     :PessoaB    ex:temPai     :PessoaA.
```

[4] RDFS é um padrão para construção de vocabulários e taxonomias minimalista, especificado em [73]

- **owl:minCardinality**: determina que uma classe deve possuir, pelo menos, N valores semanticamente distintos para a propriedade em questão. Um exemplo deste tipo de restrição é ilustrado no Código 2.3, a propriedade *ex:fala* interliga uma *ex:Pessoa* a um *ex:Idioma*, e para cada *ex:Pessoa*, deve haver pelo menos uma propriedade *ex:fala* associada a ela, denotando que essa pessoa é capaz de falar pelo menos um idioma.

Código 2.3 Exemplo da restrição *owl:minCardinality*

```
% Definição das classes e propriedades:
    ex:Pessoa  a   owl:Class.
    ex:Idioma  a   owl:Class.
    ex:fala    a   owl:Property;
% Definindo as restrições:
    ex:restricao1    rdf:type   owl:Restriction ;
        owl:onProperty  ex:fala ;
        owl:minCardinality  1^^xsd:nonNegativeInteger .
```

- **owl:maxCardinality**: determina que uma classe deve possuir, no máximo, N valores semanticamente distintos para a propriedade em questão. Este tipo de restrição é ilustrado no Código 2.4, cujo objetivo é restringir que todo indivíduo pode ter, no máximo, dois pais vivos.

Código 2.4 Exemplo da restrição *owl:maxCardinality*

```
% Definição das classes e propriedades:
    ex:Pessoa  a   owl:Class.
    ex:temPaisVivos    a   owl:Property;
% Definindo as restrições:
    ex:restricao2    rdf:type   owl:Restriction ;
        owl:onProperty  ex:temPaisVivos ;
        owl:maxCardinality  2^^xsd:nonNegativeInteger .
```

- **owl:cardinality**: determina que uma classe deve possuir, exatamente, N valores semanticamente distintos para a propriedade em questão. Este tipo de restrição é ilustrado no Código 2.5, cujo objetivo é restringir que todo indivíduo deve ter, exatamente, dois pais.

Código 2.5 Exemplo da restrição *owl:cardinality*

```
1 % Definição das classes e propriedades:
2     ex:Pessoa a owl:Class.
3     ex:temPais a owl:Property;
4 % Definindo as restrições:
5     ex:restricao3 rdf:type owl:Restriction ;
6         owl:onProperty ex:temPais ;
7         owl:cardinality 2^^xsd:nonNegativeInteger .
```

Estes são alguns exemplos que demonstram a aplicabilidade da linguagem OWL, a qual se mostra mais expressiva em relação ao padrão RDFS, o que a possibilita de ser utilizada para representar e modelar domínios mais complexos, como por exemplo, a *privacidade em IoT*.

2.4.3 *Simple Protocol and RDF Query Language* (SPARQL)

SPARQL é uma linguagem para escrita de consultas sobre triplas RDF [67], permitindo que sejam escritas quatro formas de consulta, cada qual com sua devida finalidade, sendo: *SELECT, CONSTRUCT, ASK* e *DESCRIBE*. Estas formas de consultas serão exemplificadas a seguir, com base no conjunto de dados RDF ilustrado na Tabela 2.2.

Tabela 2.2: *Fonte de dados RDF*

Sujeito	Predicado	Objeto
:p1	rdf:type	foaf:Person
:p1	foaf:firstName	"Joao"^^xsd:string
:p1	foaf:mbox	"joao@mail.com"^^xsd:string
:p1	foaf:age	19^^xsd:integer
:p2	rdf:type	foaf:Person
:p2	foaf:firstName	"Andre"^^xsd:string
:p2	foaf:mbox	"andre@mail.com"^^xsd:string
:p2	foaf:age	15^^xsd:integer

- ***SELECT*** – são similares às consultas SQL tradicionais, retornam os dados estruturados no formato de tabela. O Código 2.6 ilustra uma consulta *SELECT*, a qual é explicada a seguir.
  - Linha 5: os atributos listados após a cláusula *SELECT* são as informações consultadas, ou seja, os dados que se desejam obter, os quais correspondem aos títulos das colunas da Tabela 2.3;

– Linhas 6 a 12: a cláusula *WHERE* especifica um *subgrafo*, escrito sob a forma de triplas RDF em sintaxe Turtle. Este subgrafo representa o conjunto de informações que deverão ser buscadas na fonte de dados RDF. Além disso, na linha 11 existe a cláusula *FILTER*, que garante que os resultados só serão retornados, se uma determinada condição for satisfeita, neste caso, apenas se a idade for maior ou igual a 18.

Código 2.6 Consulta em SPARQL, utilizando a cláusula SELECT.

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX xsd: <http://w3.org/2001/XMLSchema#>
3  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
4
5  SELECT ?pessoa ?nome ?email ?idade
6  WHERE{
7     ?pessoa rdf:type foaf:Person;
8             foaf:firstName ?nome;
9             foaf:mbox ?email;
10            foaf:age  ?idade.
11    FILTER(?idade >= 18^^xsd:integer)
12 }
```

Em linhas gerais, o processador SPARQL localiza subgrafos na fonte de dados RDF, os quais são comparados com o subgrafo expresso na cláusula *WHERE*. Caso não seja encontrada nenhuma combinação entre o subgrafo e as triplas armazenadas na fonte de dados, o retorno da consulta é nulo. Caso seja encontrada alguma combinação, o resultado é apresentado, conforme ilustra a Tabela 2.3.

Tabela 2.3: *Resultado da consulta ilustrada no código 2.6*

pessoa	nome	email	idade
:p1	“Joao”^^xsd:string	“joao@mail.com”^^xsd:string	19^^xsd:integer

Além disso, SPARQL possui cláusulas que alteram o resultado de uma consulta, como FILTER, OPTIONAL e ORDER BY, além daquelas que permitem realizar agrupamentos através de GROUP BY, como HAVING, SUM, AVG, etc. O Código 2.7 demonstra a utilização da cláusula AVG, a qual é exemplificada a seguir.

– Linha 1: a informação consultada, neste caso, é dada pela média das idades, obtida através da cláusula AVG(?idade). O resultado desta média é atribuída a variável ?mediaIdade.

– Linhas 2 a 5: a cláusula *WHERE* especifica um *subgrafo*, especificando que para um indivíduo possa entrar na contagem da média de idades, precisa pertencer à classe *foaf:Person* e ter a propriedade *foaf:age* preenchida.

Código 2.7 Consulta em SPARQL, utilizando a cláusula SELECT com AVG.

```
1 SELECT (AVG(?idade) AS ?medialdade)
2 WHERE{
3   ?pessoa rdf:type foaf:Person;
4           foaf:age  ?idade.
5 }
```

Consequentemente, o resultado da consulta referente ao Código 2.7 é 19^^xsd:integer.

- ***CONSTRUCT*** – retorna um grafo RDF, contendo um conjunto de triplas, as quais podem ser originadas da própria fonte de dados, ou triplas novas, construindo um novo conhecimento.
  O Código 2.8 ilustra uma consulta *CONSTRUCT*, cujo objetivo é realizar a transformação de indivíduo modelado através da ontologia *FOAF*, para um indivíduo modelado através da ontologia *VCARD*, o qual é detalhado a seguir:
  – Linhas 3 a 6 – a cláusula *CONSTRUCT* descreve o padrão de triplas que deve ser retornado. No caso, o objetivo é associar a uma *pessoa* um *nome* através da propriedade *vcard:FN* e um *email* através da propriedade *vcard:hasEmail*;
  – Linhas 7 a 11 – a cláusula *WHERE* descreve condição de retorno, que especifica o padrão de subgrafo que será transformado.

Código 2.8 Consulta em SPARQL, utilizando a cláusula *CONSTRUCT*.

```
1  PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>
2
3  CONSTRUCT{
4      ?pessoa vcard:FN ?nome;
5              vcard:hasEmail ?mail.
6  }
7  WHERE{
8      ?pessoa a foaf:Person;
9               foaf:firstName ?nome;
10              foaf:mbox ?mail.
11 }
```

O Código 2.9 ilustra o grafo resultante desta consulta, serializado em formato *Turtle*, no qual as propriedades FOAF (*foaf:firstName* e *foaf:mbox*) foram convertidas para propriedades semelhantes da ontologia *VCard* (*vcard:FN* e *vcard:hasEmail*).

Código 2.9 Resultado da consulta ilustrada no Código 2.8.

```
1 @prefix : <http://example.org/foaf-instance#>.
2
3 :p1 vcard:FN "Joao"^^xsd:string ;
4     vcard:hasEmail "joao@mail.com"^^xsd:string .
5
6 :p2 vcard:FN "Andre"^^xsd:string ;
7     vcard:hasEmail "andre@mail.com"^^xsd:string .
```

- ***ASK*** – testa se uma consulta tem ou não uma solução. A resposta é sempre um valor booleano (*True* ou *False*). O Código 2.10 ilustra esta consulta, o qual utilizando a cláusula ASK (linha 1) verifica se existe uma *pessoa* cujo nome é *Joao* (linhas 2 e 3)

Código 2.10 Consulta em SPARQL, utilizando a cláusula ASK.

```
1 ASK{
2     ?pessoa a foaf:Person;
3             foaf:firstName  "Joao".
4 }
```

- ***DESCRIBE*** – retorna um grafo, contendo "tudo" sobre um determinado recurso em uma fonte de dados. O Código 2.11 ilustra uma consulta deste tipo, cujo objetivo é obter todas as informações relacionadas à pessoa de nome "João".
  - Linhas 1 – a cláusula *DESCRIBE* determina qual variável deverá ser descrita;
  - Linhas 3 e 4 – a cláusula *WHERE* verifica se existe uma pessoa (*foaf:Person*) cujo nome é "João" (*foaf:firstName*). Caso exista, a consulta deve descrever todos os demais dados desta pessoa.

Código 2.11 Consulta em SPARQL, utilizando a cláusula DESCRIBE.

```
1 DESCRIBE ?pessoa
2 WHERE{
3     ?pessoa a foaf:Person;
4             foaf:firstName  "Joao".
5 }
```

O Código 2.12 ilustra o grafo resultante desta consulta, serializado em formato *Turtle*, no qual todas as informações registradas sobre a pessoa de nome "Joao" foram retornadas (isto é, o recurso :p1), contendo o nome (*foaf:firstName*), email (*foaf:mbox*) e idade (*foaf:age*).

Código 2.12 Consulta em SPARQL, utilizando a cláusula DESCRIBE.

```
1 @prefix : <http://example.org/instance#> .
2
3 :p1 a foaf:Person ;
4     foaf:firstName    "Joao" ;
5     foaf:mbox         "joao@mail.com" ;
6       foaf:age          "19"^^xsd:integer .
```

2.5 Modelos Ontológicos

Após terem sido discutidos os pilares da Web Semântica, como as ontologias (Seção 2.3) e suas tecnologias padrão (Seção 2.4), o intuito desta Seção é de apresentar, resumidamente, as principais ontologias que servem de base para a realização deste trabalho de pesquisa, por terem relacionamento direto com o tema de privacidade em IoT.

2.5.1 Vocabulário WGS84 (*Basic Geo Vocabulary* WGS84)

WGS84[5] é um vocabulário RDF minimalista, desenvolvido pelo *W3C Incubator Group* para representar informações de geolocalização [64]. O vocabulário WGS84 conta com duas principais classes, sendo elas *geo:SpacialThing* e *geo:Point*, como mostra a Figura 2.3.

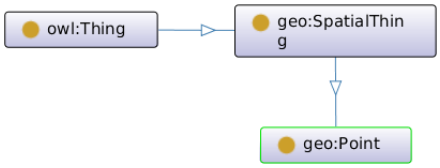

Figura 2.3: *Vocabulário WGS84.*

Em WGS84, *geo:Point* é uma subclasse de *geo:SpatialThing*, indicando que qualquer coordenada pode ser mapeada para algum "coisa" no espaço. A classe *geo:Point* é a parte central do vocabulário, permitindo representar a localização de um ponto no espaço, através das seguintes propriedades:

[5] WGS84 é um acrônimo para **W**orld **G**eodetic **S**ystem 19**84**, que é um sistema de coordenadas terrestres.

- ***geo:lat***: permite representar coordenada de latitude;
- ***geo:long***: permite representar coordenada de longitude;
- ***geo:alt***: permite representar coordenada de altitude.

O Código 2.13 exemplifica como utilizar o vocabulário WGS84 para representar coordenadas geográficas e associá-las a pontos espaciais.

Código 2.13 Exemplo de anotação semântica utilizando o vocabulário *WGS84*.

```
1 @prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
2 @prefix : <http://example.org/wgs84-instance#> .
3
4 :pontoA a geo:Point;
5   geo:lat "30.42"^^xsd:decimal;
6   geo:long "-151.85"^^xsd:decimal;
7   geo:alt "10"^^xsd:decimal.
```

- Linha 4: instanciação de um ponto geográfico (*:pontoA*), o qual pertence a classe *geo:Point*
- Linhas 5 a 7: anotação das propriedades associadas ao (*:pontoA*), que são respectivamente: latitude (*geo:lat*, com o valor 30.42), longitude (*geo:long*, com o valor -151.85) e altitude (*geo:alt*, com o valor 10).

2.5.2 Ontologia QU (*Quantity Kinds and Units*)

QU é uma ontologia desenvolvida para possibilitar que sejam representados tipos de quantidade e unidades de medidas [42]. As principais classes da QU são aquelas que nomeiam a ontologia: *QuantityKind* e *Unit*.

- ***QuantityKind***: utilizada para representar tipos de medidas (p. ex.: temperatura corporal, velocidade ou distância).
- ***Unit***: utilizada para representar unidades de medida (p. ex.: Graus Celsius, Quilômetros por hora, Metros).

Em virtude de suas características, a ontologia QU não permite expressar relacionamentos complexos, uma vez que é munida de pouquíssimas propriedades, no entanto, a riqueza da ontologia QU está na possibilidade de estendê-la facilmente, como ilustra o Código 2.14, o qual é detalhado a seguir:

Código 2.14 Exemplificando a possibilidade de extensão da ontologia *QU*.

```
1 @prefix qu: <http://purl.oclc.org/NET/ssnx/qu/qu#> .
2 @prefix : <http//www.example.org/qu-extension#> .
3
4 :Graus rdfs:subClassOf qu:Unit .
5 :GrausCelsius rdfs:subClassOf qu:Graus .
6 :GrausFahrenheit rdfs:subclassOf qu:Graus.
7
8 :Temperatura rdfs:subClassOf qu:QuantityKind .
9 :TemperaturaCorporal rdfs:subClassOf :Temperatura .
```

- Linha 4: é especificado que *:Graus* é uma subclasse de *qu:Unit*, estendendo a semântica da ontologia QU. Isto quer dizer que *:Graus* passa a representar uma unidade de medida.
- Linhas 5 e 6: são especificadas duas classes: *:GrausCelsius* e *:GrausFahrenheit*, neste caso, ambas foram definidas como subclasse de *:Graus*, indicando que ambas são especializações desta unidade de medida.
- Linha 8: desta vez é especificado que a classe *:Temperatura* é uma subclasse de *qu:QuantityKind*, estendendo novamente tipo de medição.
- Linha 9: por fim, determina que existe uma classe *:TemperaturaCorporal* que é uma especialização de *:Temperatura*. Isto é, toda medição de *:TemperaturaCorporal* também é uma medição de *:Temperatura*.

Esta abordagem de estender a ontologia QU não é algo novo. Diversas ontologias já publicaram suas extensões, dentre elas, destaca-se a taxonomia M3-Lite [23]. Esta taxonomia classifica dispositivos (tais como sensores de pressão, sensores de luminosidade ou interruptores de luz), domínios de interesses (tais como saúde, casa inteligente ou monitoramento ambiental), tipos de medição (tais como temperatura ou distância) e unidade de medidas (tais como graus Celsius ou quilômetros). Atualmente, a taxonomia M3-Lite conta com mais de 75 classes que estendem o conceito de *qu:QuantyKind* e mais de 85 classes que estendem o conceito de *qu:Units*.

2.5.3 Ontologia SSN (*Semantic Sensor Network*)

SSN é uma ontologia desenvolvida pela *W3C Incubator Group*, cujo objetivo é auxiliar no gerenciamento, consulta e combinação de sensores, bem como os dados resultantes do sensoriamento destes [18]. A ontologia SSN pode ser utilizada sob diferentes perspectivas, sendo elas:

- Do sensor, com foco sobre o que é sensoreado, como é sensoreado e o que é detectado;
- Da observação, com foco em aferições de dados;
- Do sistema, com foco nos sistemas de sensores e sua implantação;
- Das características e propriedades de interesse monitoradas, pelos sensores.

A ontologia SSN utiliza o padrão SSO[6] como ferramenta central para realizar a representação de sensoriamento. Senso assim, o padrão SSO é parte central da ontologia SSN, pois relaciona o sensor, a propriedade sensoreada, a entidade de interesse e o dado aferido, abrangendo as perspectivas acima descritas [18]. A Figura 2.4 ilustra o padrão SSO, o qual é a parte central da ontologia SSN, e é detalhado a seguir.

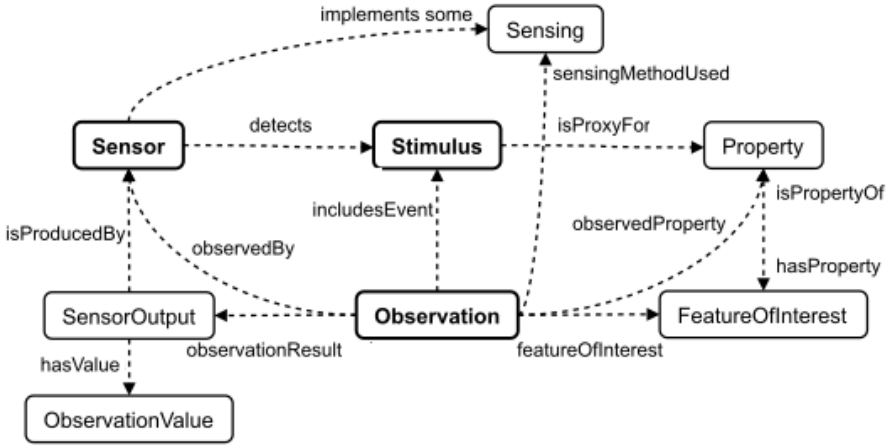

Figura 2.4: *Padrão SSO, parte central da ontologia SSN.*

- ***Stimulus***: representam as mudanças detectadas no ambiente. Os estímulos são o ponto de partida de cada medição realizada por um sensor.
- ***Sensor***: representa um dispositivo físico de sensoriamento, uma vez que um *sensor* é uma subclasse de um *dispositivo* (*ssn:Device*), além disso, um ou mais sensores compõe um sistema (*ssn:System*) e podem pode observar uma propriedade (*ssn:Property*) ou detectar estímulos no ambiente (*ssn:Stimulus*), o que resulta em uma observação (*ssn:Observation*). Feita a observação, um resultado é produzido e ligado ao sensor responsável por meio da classe *ssn:SensorOutput*, através da propriedade *isProducedBy*. Um *ssn:SensorOutput* também possui um valor, representado pela classe *ssn:ObservationValue* e ligado através da propriedade *ssn:hasValue*.

[6] do Inglês, ***S**timulus-**S**ensor-**O**bservation*

- ***Observation***: as *observações* atuam como a conexão entre um *estímulo* de entrada e o resultado de um *sensor*. Estão associadas a uma característica de interesse (*ssn:FeatureOfInterest*), a qual é observada, e produzem resultados (*ssn:SensorOutput*).

A seguir, o Código 2.15 ilustra um exemplo de modelagem utilizando a ontologia SSN, no qual um determinado sensor observa e registra a temperatura corporal de um paciente.

Código 2.15 Exemplo de utilização da ontologia SSN

```
1  @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
2  @prefix : <http//www.example.org/ssn-instance#> .
3
4  :paciente041n a ssn:FeatureOfInterest .
5
6  :temperaturaCorporal a ssn:Property .
7
8  :sensorVSO a ssn:SensingDevice ;
9        ssn:observes :temperaturaCorporal .
10
11 :observacaoTemperatura a ssn:Observation ;
12       ssn:observedBy :sensorVSO ;
13       ssn:featureOfInterest :paciente041n ;
14       ssn:observationResult :outputVSO ;
15       ssn:observedProperty :temperaturaCorporal .
16
17 :outputVSO a ssn:SensorOutput ;
18       ssn:hasValue :valorObservacao1 ;
19       ssn:isProducedBy ssn:sensorVSO .
20
21 :valorObservacao1 a ssn:ObservationValue ;
22       ssn:hasOutputValue "37.300"^^xsd:float .
```

- Linha 4: é expressada a característica de interesse, ou seja, o que deve ser observado. Neste exemplo, o IRI referente a um paciente (*:paciente041n*) é a *ssn:FeatureOfInterest*.
- Linha 6: a seguir, é expressado a propriedade que será observada no paciente. Neste exemplo, a temperatura corporal (*:temperaturaCorporal*) foi definida como a *ssn:Property*.
- Linhas 8 e 9: são expressos as definições relacionadas ao sensor. Em linhas gerais, o sensor *:sensorVSO* deve observar ou medir temperatura corporal, expresso pela tripla (*:sensorVSO, ssn:observes, :temperaturaCorporal*).

- Linhas 11 a 15: são descritas as informações a respeito da observação de temperatura realizada pelo sensor. Estas triplas expressam: i) que a observação foi produzida pelo sensorVSO, através da tripla (*:observacaoTemperatura, ssn:observedBy, :sensorVSO*); ii) que esta observação é referente ao paciente 041n, através da tripla (*:observacaoTemperatura, ssn:featureOfInterest, :paciente041n*); iii) que esta observação é referente à propriedade de temperatura corporal, através da tripla (*:observacaoTemperatura, ssn:observedProperty, :temperaturaCorporal*); e por fim iv) que os resultados desta observação estão disponíveis na *output* do sensor, através da tripla (*:observacaoTemperatura, ssn:observationResult, :outputVSO*).
- Linhas 17 a 19: detalha o que é a output do sensor (*:outputVSO*), gerado pela observação e citado anteriormente. Em linhas gerais, esta output é produzida por um sensor (*:outputVSO, ssn:isProducedBy, ssn:sensorVSO*) e consequentemente, produz um valor real, associado a medição (*:outputVSO, ssn:hasValue, :valorObservacao1*).
- Linhas 21 e 22: por fim, é expresso o valor da observação, citado anteriormente, através da tripla (*:valorObservacao1, ssn:hasOutputValue, "37.30"^^xsd:float*).

Neste exemplo, utilizando-se a semântica provida pelo ontologia SSN, conclui-se que a temperatura corporal do paciente041n foi aferida pelo sensorVSO com o valor de 37.30. No entanto, ainda não é possível responder algumas perguntas, tais como: onde o sensor está localizado? e o paciente? 37.30 é referente a qual unidade de medida, Graus Celsius ou Fahrenheit?

Além disso, em virtude da complexidade dos conceitos, relações e das diferentes visões que a ontologia *SSN* permite representar, a ontologia *SSN* possui uma alta complexidade lógica, do tipo $\mathcal{ALCRIQ}$, o que traduzindo em complexidade computacional pertence à classe *NExpTime-hard*, logo, é uma ontologia complexa para processar e realizar inferências. Devido a estes fatos, foi desenvolvida uma ontologia mais leve e adequada para modelagem e processamento de dados para a Internet das Coisas, chamada IoT-Lite, que é o assunto subsequente.

2.5.4 Ontologia IoT-Lite

Enquanto a grande maioria dos modelos semânticos para IoT tendem a descrever conceitos e relações detalhadamente, a ontologia IoT-Lite (Figura 2.5) representa apenas os conceitos mais importantes para o domínio IoT, como sensores, dados de sensores, localização e tipos de dados [11]. Isso abre o caminho para a criação de sistemas escaláveis, pois reduz o custo computacional de processamento e a memória necessária para executar consultas e inferências em aplicativos que utilizem a ontologia IoT-Lite.

A ontologia IoT-Lite estende a semântica da ontologia SSN, que define conceitos bem compreendidos para representação de sensores [43][18], e também reutiliza conhecimento de outras ontologias, como: i) Vocabulário WGS84, que possibilita representar informações sobre pontos espaciais, usando o padrão de coordenadas terrestre [64]; e ii) Ontologia QU, que fornece conceitos sobre tipos de medições e unidades de medida [42]. Uma breve descrição de seus conceitos e as relações entre eles é apresentada abaixo:

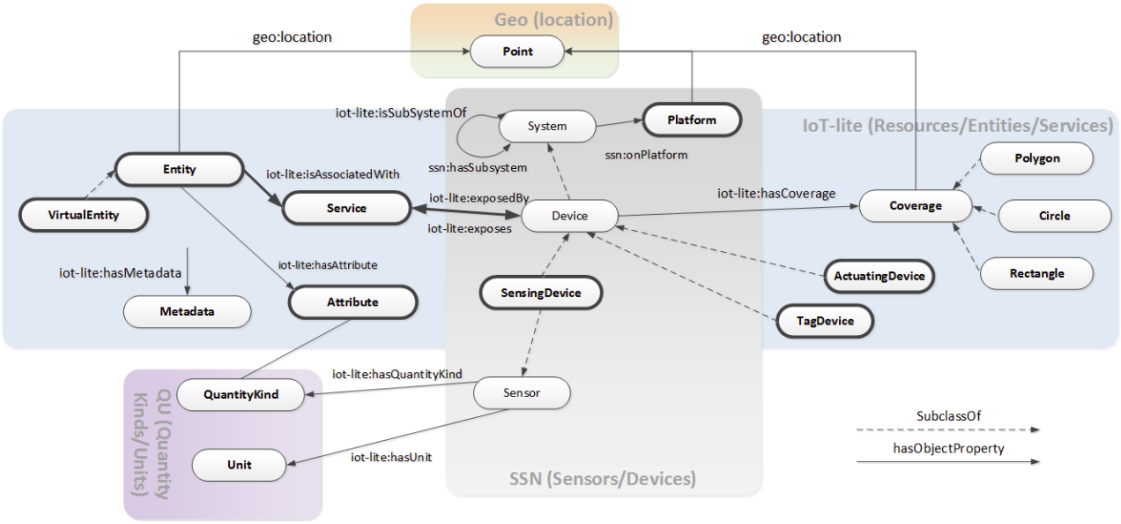

Figura 2.5: *Ontologia IoT-Lite.*

- ***iot-lite:Entity***: são as " coisas " que compõem o ambiente de IoT, equivalentes a objetos (*iot-lite:Objetos*). Os objetos (ou entidades) podem ter atributos associados à eles, expressos pela tripla *(iot-lite:Entity, iot-lite:hasAttribute, iot-lite:Attribute)*.
- ***iot-lite:Attribute***: atributos são valores que podem ser medidos. O tipo de um atributo é expresso através da classe *qu:QuantityKind*, herdada da ontologia QU, como na tripla (*iot-lite:Attribute, iot-lite:hasQuantityKind, qu:QuantityKind*).
- ***ssn:System***: é um conceito definido na ontologia SSN, representando qualquer sistema, incluindo os sistemas de sensores.
- ***ssn:Sensor***: também é um conceito definido na ontologia SSN, representando um tipo específico de dispositivo. Este conceito é estendido na IoT-Lite, de modo que um sensor passa a estar associado ao tipo de medição e à unidade de medida que sensoreiam. Estes conceitos são expressos, respectivamente, pelas classes *qu:QuantityKind* e *qu:Unit*, ambas herdadas da ontologia QU.
- ***ssn:Device***: também é um conceito definido na ontologia SSN, representando dispositivos em geral. Esse conceito é estendido na IoT-Lite, adicionando uma área de cobertura espacial para cada dispositivo, como em *(ssn:Device, iot-lite:hasCoverage, iot-lite:Coverage)*.

- ***iot-lite:Coverage***: as áreas de cobertura são representadas por coordenadas geográficas, usando a classe (*geo:Point*), que é um conceito reutilizado da ontologia WGS84. A área de cobertura geográfica está vinculada às suas coordenadas através da propriedade *iot-lite:hasPoint*.
- ***iot-lite:Service***: um dispositivo (ou sistema) é exposto na forma de um serviço, permitindo que tenham seus recursos ou dados acessados e/ou consumidos através da Web. A associação de um dispositivo (ou sistema) a um serviço ocorre na forma da tripla *(ssn:Device, iot-lite:exposedBy, iot-lite:Service)*.

A seguir, o Código 2.16 ilustra um exemplo de modelagem utilizando a ontologia IoT-Lite, no qual é descrito um sensor de temperatura corpórea em um hipotético cenário de monitoramento de sinais vitais.

Código 2.16 Exemplo de utilização da ontologia IoT-Lite

```
@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/iot-lite/iot-lite#> .
@prefix m3: <http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite> .
@prefix : <http://example.org/iot-lite-instance#> .

:paciente001 rdf:type iot-lite:Entity ;
    iot-lite:hasAttribute :temperaturaCorporal ;
    iot-lite:isAssociatedWith :servicoDeTemperatura001 .

:temperaturaCorporal rdf:type iot-lite:Attribute ;
    iot-lite:hasQuantityKind m3:BodyTemperature .

:sensor001 rdf:type ssn:Sensor ;
    iot-lite:hasQuantityKind m3:BodyTemperature ;
    iot-lite:hasUnit m3:DegreeCelsius ;
    geo:hasLocation :localizacaoSensor001 ;
    iot-lite:exposedBy :servicoTemperatura001 .

:localizacaoSensor001 rdf:type geo:Point ;
    geo:long "-49.255"^^xsd:float ;
    geo:lat "-16.6799"^^xsd:float ;
    iot-lite:altRelative "2o andar"^^xsd:string .

:servicoTemperatura001 rdf:type iot-lite:Service ;
    iot-lite:endpoint
        "http://example.org/servico/endpoint/P001-temp"^^xsd:anyURI .
    iot-lite:interfaceDescription
        "http://example.org/servico/interface/P001-temp"^^xsd:anyURI .
```

- Linhas 5 a 7: o paciente é modelado como uma entidade na IoT, através da tripla (*:paciente, rdf:type, iot-lite:Entity*). Este paciente tem um atributo de interesse, no caso, temperatura corporal, expresso pela tripla (*:paciente001, iot-lite:hasAttribute, :temperaturaCorporal*). Além disso, também é expresso que o paciente está associado a um serviço, através da tripla (*:paciente001, iot-lite:isAssociatedWith, :servicoTemperatura001*).
- Linhas 9 e 10: é realizada a definição do atributo *:temperaturaCorporal*, através da tripla (*:temperaturaCorporal, iot-lite:hasQuantityKind, m3:BodyTemperature*). Esta tripla expressa que o atributo *:temperaturaCorporal* é relacionado à temperatura corporal, para isso, foi utilizado a classe da taxonomia *M3-Lite* que representa este conceito (*m3:BodyTemperature*).
- Linhas 12 a 16: a seguir, é realizada a definição do sensor. Foi definido que o *:sensor001* observa medições de temperatura corporal utilizando Graus Celsius como unidade de medida, representado pelas triplas (*:sensor001, iot-lite:hasQuantityKind, m3:BodyTemperature*) e (*:sensor001, iot-lite:hasUnit, m3:DegreeCelsius*), respectivamente. Além disso, ao sensor também são associadas as informações de localização (*:sensor001, geo:hasLocation, :localizacaoSensor001*) e do serviço que expõe este sensor na web (*:sensor001, iot-lite:exposedBy, :servicoTemperatura001*).
- Linhas 18 a 21: é especificado a localização do sensor, para isso, foram utilizadas as propriedades da ontologia WGS84, especificando as coordenadas de longitude e latitude, como nas triplas (*:localizacaoSensor001, geo:long, "-49.255"^^xsd:float*) e (*:localizacaoSensor001, geo:lat, "-16.6799"^^xsd:float*), respectivamente. Além disso, é expresso que o sensor em questão está localizado no segundo andar, através da tripla (*:localizacaoSensor001, iot-lite:altRelative, "2o andar"^^xsd:String*).
- Linhas 23 a 27: por fim, é realizada a especificação do serviço que expõe o *:sensor001*. Para isso, foram especificados o endpoint do serviço, isto é, a URI pela qual o serviço em questão pode ser acessado e/ou consumido, através da propriedade *iot-lite:endpoint*. Além disso, também foi especificada a URI onde a interface do serviço é descrita, por meio da propriedade *iot-lite:interfaceDescription*.

A partir deste cenário, é possível perceber que utilizando a ontologia IoT-Lite é possível construir um modelo mais expressivo, se comparado à modelagem anterior, realizada com a ontologia SSN, uma vez que a modelagem realizada com a ontologia IoT-Lite foi capaz de responder as "perguntas" que ficaram em aberto na modelagem anterior, relacionadas à localização e unidades de medidas associadas ao sensor.

Além disso, graças as decisões de projeto da *IoT-Lite*, onde apenas os principais conceitos e relações necessárias para compor o cenário da IoT foram disponibilizados, isto garante uma ontologia concisa, expressiva e de fácil utilização, o que impactou diretamente em suas métricas, com principal destaque para a complexidade lógica, do tipo $\mathcal{ALUI}(\mathcal{D})$,

isto é, de complexidade computacional *PSpace-complete*, o que representa um ganho em relação as suas antecessoras, como a *SSN*, cumprindo seu propósito original, de ser uma ontologia leve para a IoT.

2.6 Considerações Finais

Este capítulo apresentou o conceito de Internet das Coisas, onde a ideia principal é permitir que os objetos ou "coisas" do dia-a-dia possam se conectar à Internet e comunicar entre si. No entanto, este conceito traz consigo diversos problemas e desafios de pesquisa, entre os quais destaca-se a **privacidade**, pois espera-se que os sensores coletem e disponibilizem um grande volume de informações, as quais estão diretamente relacionadas aos usuários.

A privacidade já é um assunto conceitualmente denso, quando incluída no contexto de IoT, o problema se torna ainda mais complexo. Deve-se levar em conta fatores como a heterogeneidade de dispositivos e protocolos de comunicação, legislações específicas de cada país, características de desempenho, entre outros. Além disso, a privacidade tange todas as fases de um sistema, e deve estar incorporada em todas elas, como é sugerido pelo *framework* PbD.

Dentre as fases de um sistema que a privacidade pode ser incorporada, incluem a coleta, modelagem, processamento, retenção, e divulgação, as quais compreendem o ciclo de vida de contexto. Este trabalho aborda principalmente a etapa de modelagem, mas também tange a etapa de processamento, neste sentido, optamos por adotar as tecnologias já conhecidas da Web Semântica, com destaque para as ontologias, pois o formalismo e a linguagem consensual das ontologias possibilita que dispositivos heterogêneos tenham uma interpretação única e consensual de um mesmo conceitos, ou conjuntos de conceitos, no caso, da privacidade em IoT.

Por fim, vale ressaltar não existe um método considerado correto para avaliar ontologias, o que existem são métodos mais adequados para determinados objetivos. Como nossa proposta era desenvolver uma ontologia de privacidade para a internet das coisas, que preservasse as características de leveza da ontologia IoT-Lite, por nós importada, optamos por realizar a avaliação, principalmente, com base na métrica de complexidade lógica, a qual é um indicativo da dificuldade de se trabalhar com determinado ontológico, ou seja, da dificuldade de se processar dados anotados com uma determinada ontologia.

CAPÍTULO 3

Ontologia *IoT-Priv*

A *IoT-Priv* foi projetada com o objetivo de fornecer conceitos e relações, os quais possibilitam que cenários e/ou aplicações IoT expressem suas definições, restrições e políticas de privacidade utilizando um método livre de ambiguidade, graças ao formalismo fornecido pelas ontologias [6]. Além disso, a *IoT-Priv* deve ser expressiva, extensível e leve.

3.1 Requisitos de Privacidade

Com base nos trabalhos de Perera et al. [55] [53], Garcia et al. [27], He et al. [33] e no *framework* PbD [16], uma série de requisitos de privacidade para a IoT foram identificados e compilados na Tabela 3.1. Em seguida, algumas observações sobre esses requisitos são destacadas:

Tabela 3.1: *Compilação de requisitos de Privacidade para a Internet das Coisas.*

Req.	Descrição	Fonte
#1	O modelo deve permitir a identificação de proprietários de dados.	[33]
#2	O modelo deve permitir a identificação de consumidores de dados.	[33]
#3	O modelo deve permitir que os dados consumidos sejam identificados.	[33]
#4	O modelo deve permitir que um provedor de dados especifique um conjunto de finalidades com as quais o acesso às suas informações é permitido.	[33]
#5	O modelo deve permitir que um consumidor de dados informe o propósito de acesso aos dados que serão adquiridos.	[33]
#6	O modelo deve permitir que um provedor de dados apresente um conjunto de obrigações associadas ao acesso e retenção de seus dados.	[33]
#7	Um provedor de dados deve poder consentir com a coleta de seus dados antes que isso aconteça.	[27]

Tabela 3.1: continuação da página anterior.

Req.	Descrição	Fonte
#8	O modelo deve permitir que um provedor de dados descreva como o dado foi coletado.	[27]
#9	O modelo deve permitir que um provedor de dados especifique o tempo máximo de retenção de seus dados.	[27]
#10	O modelo deve permitir que um provedor de dados informe se o consumidor pode completar, corrigir e / ou atualizar seus dados quando adquiridos.	[27]
#11	O modelo deve permitir que um provedor de dados solicite registros sobre como seus dados foram usados pelos consumidores.	[27]
#12	O modelo deve permitir o registro de informações temporais quando um consumidor requisita acesso a um serviço.	contribuição do autor, baseado em [16] e [55]
#13	O modelo deve permitir o registro de informações de localização quando um consumidor requisita acesso a um serviço.	contribuição do autor, baseado em [16]
#14	O modelo deve permitir que um provedor de dados mantenha uma lista de acesso com consumidores autorizados para cada serviço próprio.	[55]
#15	As listas de serviços de dados devem ser configuráveis.	[55]
#16	Os provedores de dados e os consumidores de dados devem concordar com uma recompensa que os primeiros possam receber devido aos riscos de compartilhar dados.	[55]
#17	Os tipos de recompensa associados ao compartilhamento de dados devem ser explicitamente identificados quando necessário.	[55]
#18	O modelo deve permitir o uso de mecanismos de reclamação e reparação em caso de violação de privacidade.	[16] [55]
#19	O modelo deve permitir a identificação da entidade de destino da divulgação de informações pessoais.	[16] [55]
#20	O modelo deve permitir o registro de informações privadas divulgadas a um sistema juntamente com a frequência de divulgação.	[16] [55]
#21	O modelo deve permitir que um provedor de dados defina intervalos de tempo em que seus dados devem ser excluídos.	[55]
#22	O modelo deve permitir que um provedor de dados escolha como anonimizar suas informações pessoais antes de compartilhar dados.	[53]

Tabela 3.1: continuação da página anterior.

Req.	Descrição	Fonte
#23	O modelo deve permitir que um provedor de dados escolha como usar a criptografia no armazenamento de seus dados privados.	[53]
#24	O modelo deve permitir que um provedor de dados use o chamado mecanismo de respostas seguras, no qual a divulgação de informações privadas oculta os dados brutos.	[53]
#25	O modelo deve permitir que um consumidor de dados registre suas ações como um complemento ao requisito #11.	[53]
#26	O modelo deve usar os padrões da indústria.	[53]
#27	O modelo deve possibilitar que os provedores de dados especifiquem áreas espaciais, nas quais os seus serviços podem ser acessados, em complemento ao requisito #13.	contribuição do autor, baseado em [16]

- Requisitos #1 a #3 abordam a identificação de fornecedores de dados, os consumidores de dados, bem como os dados consumidos.
- As requisições de consumo controlam o consumo de dados e encapsulam as seguintes informações: finalidade de uso, instante e local em que uma solicitação é feita (requisitos #5, #12 e #13, respectivamente).
- Segue um conjunto de requisitos destinados a permitir a especificação de políticas de privacidade:
  - Requisitos #4 e #6 permitem que um proprietário de dados especifique normas para a concessão ou negação de acesso a seus dados, bem como obrigações para aqueles que ganharam acesso.
  - Requisitos #9, #10 e #21 a #24 permitem que os provedores de dados descrevam políticas relacionadas às suas informações pessoais, por exemplo, técnicas de criptografia e anonimização e o tempo máximo de retenção de informações.
  - Requisito #8 permite a definição do método de coleta de dados associado a um sistema ou serviço.
  - Requisitos #16 e #17 permitem especificar um tipo de recompensa (por exemplo, um valor monetário) para provedores de dados, pois compartilham dados. Por sua vez, o requisito #18 aborda a necessidade de uma política de reparação em caso de violação de privacidade.
  - Tanto um sistema quanto um serviço podem capturar e disseminar as informações pessoais dos usuários. Neste sentido, os requisitos #19 e #20 abordam a identificação da entidade de destino e a frequência de disseminação.

- Requisitos #14 e #15 permitem que provedores de dados associem listas de acesso específicas a consumidores de dados, por exemplo, aqueles com acesso concedido ou negado, ou forçados a uma restrição de qualidade de serviço.
- Requisitos #11 e #25 tratam do registro de uso de informações e consumo de serviços para fins de auditoria.
- Requisito #7 destina-se a permitir que um proprietário de dados consinta que uma entidade adquira e consuma suas informações.

Finalmente, para proteger adequadamente a privacidade em IoT, há também uma demanda por mecanismos para monitorar, avaliar e verificar a conformidade dos sistemas e consumidores com as especificações das políticas de privacidade. Por exemplo, considere um cenário em que um provedor de dados solicita o registro de uso de seus dados (requisito #11); isso deve levar os consumidores a produzirem e disponibilizarem esses registros (requisito #25). No entanto, monitorar e determinar se os consumidores produzem e disponibilizam esses registros não faz parte do escopo deste trabalho.

A próxima seção apresenta a ontologia *IoT-Priv* construída com base nos requisitos supracitados.

3.2 Desenvolvimento da Ontologia *IoT-Priv*

A *IoT-Priv* é uma ontologia de privacidade da IoT, desenvolvida na Web Ontology Language (OWL) de acordo com a Metodologia de Desenvolvimento de Ontologias 101 [50], a qual foi detalhada no Capítulo 2.

De acordo com a metodologia, o primeiro passo é definir os requisitos ou questões de competência, os quais orientarão a construção da ontologia. Assim, a *IoT-Priv* foi construída tendo como base na lista de requisitos da Seção 3.1.

O segundo passo da metodologia é identificar a possibilidade de reutilizar ontologias existentes. Como a IoT-Priv deve ser uma ontologia de privacidade para a Internet das Coisas, optamos por reutilizar uma ontologia que modelasse esse cenário, como a ontologia IoT-Lite [11], que ainda possui características como: alta expressividade, extensibilidade, flexibilidade e baixa complexidade lógica, como já foi demonstrado em [60] [61].

O terceiro passo é enumerar termos importantes na ontologia. De acordo com os requisitos da Seção 3.1, foi possível identificar diversos termos para o domínio de privacidade, entre eles: Proprietário de Dados, Política de Privacidade, Consumidor, Requisições de Consumo, Divulgação de Informação, Listas de Acesso, Disputa, Anonimização, Criptografia, Valores de Serviço, entre outros. Bermudez et al. também afirmam que os principais termos relacionados ao domínio da IoT são Sistemas, Serviços e Objetos (ou

Coisas) [11], conceitos estes que serão o ponto de ligação entre a ontologia IoT-Lite e a camada de privacidade desenvolvida pela *IoT-Priv*.

Os demais passos são orientados à implementação, e descrevem como: i) especificar classes; ii) especificar propriedades; e iii) especificar restrições.

A Figura 3.1 apresenta a camada de privacidade fornecida pela *IoT-Priv* e suas ligações com os conceitos da ontologia IoT-Lite. Na Figura, tem-se a seguinte legenda:

- os retângulos sólidos representam classes independentes de domínio, sem necessidade de serem instanciadas como um novo indivíduo;
- os retângulos tracejados são classes genéricas, mas extensíveis, uma vez que permitem que indivíduos específicos do domínio sejam instanciados. Neste caso, ontologias externas também podem ser acopladas para aumentar a expressividade, por exemplo, uma ontologia para descrever propósitos de acesso para fins médicos;
- círculos são propriedades de dados, representando dados literais;
- uma seta representa uma propriedade (ou relação), com ou sem cardinalidade;
- finalmente, uma seta pontilhada representa uma subclasse.

As classes, propriedades e respectivas definições de cardinalidade da *IoT-Priv* são detalhadas a seguir:

1. **PrivacyPolicy**: este é o conceito central da ontologia *IoT-Priv*, à qual todos os outros estão ligados. Existem dois tipos de políticas de privacidade: uma genérica e obrigatória, representada pela classe *GeneralPrivacyPolicy*; e um opcional relacionado ao armazenamento de dados (a classe *StoragePrivacyPolicy*). Se necessário, a classe *PrivacyPolicy* também permite a associação com novos tipos de políticas de privacidade, por exemplo, uma política de privacidade relacionada à inferência de dados. Usando o conceito de proteção de privacidade, as políticas de privacidade são anexadas a um serviço IoT.

   (a) **GeneralPrivacyPolicy**: esse conceito está vinculado à classe *PrivacyPolicy* por meio da propriedade *hasGeneralPrivacyPolicy*, a qual é uma propriedade obrigatória, ou seja, toda política de privacidade precisa de uma única descrição das políticas genéricas, pois a cardinalidade dessa propriedade é <*exactly 1*>. Esta classe permite especificar políticas de finalidade geral, incluindo propósito de acesso (*iot-priv:AccessPurpose*), obrigações (*iot-priv:Obligations*), configurações de anonimização (*iot-priv:AnonymizationTechniques*), valores de serviço (*iot-priv:ServiceValue*) e mecanismos de reparação (*iot-priv:Disputes*), tempo máximo de retenção de dados (*iot-priv:maxRetentionPeriod*), a localização geográfica de onde os dados podem ser acessados (*accessFrom*), dentre outras.

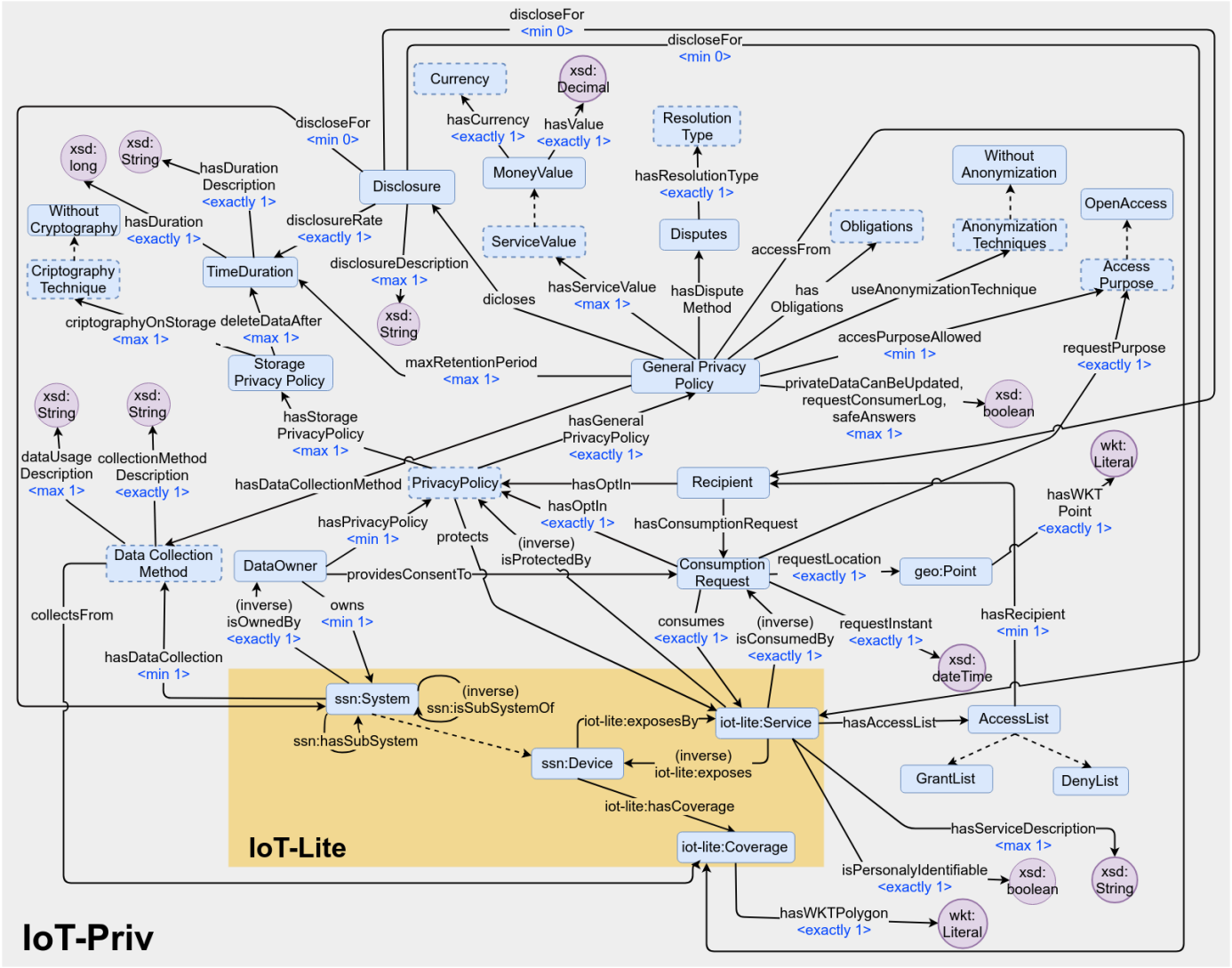

Figura 3.1: *Uma visão geral da ontologia IoT-Priv.*

   (b) **StoragePrivacyPolicy**: vinculada à classe *PrivacyPolicy* através da propriedade *hasStoragePrivacyPolicy*, a qual é uma propriedade opcional, sendo assim, não é necessário que uma política de armazenamento seja expressa, mas caso seja, apenas uma única é permitida, daí a cardinalidade associada desta propriedade é <*max1*>. Esta classe permite especificar políticas de privacidade relacionadas ao armazenamento de dados, como o uso de técnicas de criptografia para armazenamento de dados (classe *CryptographyTechniques*), ou a exclusão de dados armazenados após um intervalo de tempo decorrido (classe *TimeDuration* através da propriedade *deleteDataAfter*).

2. **DataOwner**: representa uma entidade que possui um ou mais Sistemas (*ssn:System*), representado através da propriedade *iot-priv:owns*, a qual tem cardinalidade <*min1*>, ou seja, um *DataOwner* deve ser proprietário de ao menos um Sistema. Devido à herança de classes, um proprietário de dados também pode ser o proprietário de ao menos um dispositivo (*ssn:Device*).
   Um proprietário de dados também deve ter pelo menos uma política de privacidade vinculada a ele por meio da propriedade *hasPrivacyPolicy*, daí esta tem cardinalidade <*min1*>. Além disso, um proprietário de dados também é capaz de fornecer consentimento a uma solicitação de consumo a seus dados por meio da propriedade *providesConsentTo*.
3. **Recipient**: em geral, os destinatários recebem dados ou usam recursos disponibilizados por um serviço (*iot-lite:Service*). Para torná-lo auditável, sempre que um destinatário tentar acessar um serviço, uma solicitação de consumo deverá ser registrada por meio da propriedade *hasConsumptionRequest*.
4. **ConsumptionRequest**: é produzida por destinatários (*iot-priv:Recipient*) interessados em consumir um serviço (iot-lite:Service). Esta classe permite o registro das seguintes informações: o instante de tempo e a localização geográfica a partir da qual uma solicitação é gerada (*iot-priv:requestInstant* e *iot-priv:requestLocation*, respectivamente), o propósito de acesso de uma requisição (*iot-priv:requestPurpose*), o serviço a ser consumido (*iot-priv:consumes*) e o acordo explícito com a política de privacidade que protege o serviço a ser consumido (*iot-priv:hasOptIn*). Todas essas informações são obrigatórias, logo, tem cardinalidade <*exactly1*>.
5. **AccessPurpose**: representa os possíveis propósitos do acesso a dados. É uma classe genérica e pode ser estendida para atender necessidades específicas, por exemplo, pode ser vinculada a uma ontologia ou taxonomia descrevendo tipos de propósitos de acesso. Atualmente, a classe *AccessPurpose* tem apenas a subclasse *OpenAccess*. Um propósito de acesso é expresso através da propriedade *accessPurposeAllowed* e vinculado à classe *GeneralPrivacyPolicy*, sendo que pelo menos um propósito de acesso é preciso ser especificado para cada política de privacidade, daí a propriedade

accessPurposeAllowed tem cardinalidade <*min1*>.

6. **Obligations**: representa as obrigações que um consumidor de dados deve seguir. Semelhante ao *AccessPurpose*, é genérico e extensível para atender às necessidades específicas do domínio. Uma obrigação é conectada a uma *General Privacy Policy* por meio da propriedade *hasObligations*, a qual não tem restrição de cardinalidade.
7. **AnonymizationTechnique**: anonimização pode ser necessária quando existe alguma informação capaz de identificar uma entidade. Esta classe é genérica e extensível e define se um tipo de técnica de anonimização deve ou não ser aplicado, por exemplo, ofuscação. Uma configuração de anonimização de dados vincula-se a uma *GeneralPrivacyPolicy* por meio da propriedade *useAnonymizationTechnique*, a qual não tem restrições de cardinalidade.
8. **DataCollection**: vinculada a uma política geral de privacidade por meio da propriedade *hasDataCollectionMethod*, a qual não tem restrição de cardinalidade. Essa classe genérica e extensível descreve os métodos de coleta e uso de dados. Por sua vez, a propriedade *hasDataCollection* transmite a configuração de como um sistema ou dispositivo coleta e usa dados, na forma de descrições textuais (propriedades *collectionMethodDescription* e *dataUsageDescription*, de uso obrigatório, ambas com cardinalidade <*exactly1*>).
9. **ServiceValue**: determina um valor opcional do serviço protegido por uma política de privacidade, logo tem cardinalidade <*max1*>. Como uma classe genérica e extensível, *Service Value* permite a definição de múltiplos tipos de "valores". No entanto, a priori, há apenas uma subclasse chamada *MoneyValue*, cujas propriedades *hasValue* e *hasCurrency* descrevem a recompensa monetária em virtude do compartilhamento de dados. O primeiro representa o valor monetário, enquanto o segundo vincula-se à respectiva moeda (*Currency*). Essa decisão de modelagem acomoda a grande variedade de moedas, por exemplo, *AmericanDollar* ou *BrazilianReal*. Vale ressaltar que tanto *hasValue* como *hasCurrency* são obrigatórias para descrever um valor monetário, já que possuem cardinalidade <*exactly1*>.
10. **Disputes**: representa possíveis métodos de resolução para as violações da política de privacidade. Esta classe possui a propriedade *hasResolutionType*, a qual se conecta à classe *ResolutionType*, utilizada para descrever métodos de resolução de problemas, por exemplo, ações judiciais, mediação ou conciliação. Em virtude de eventuais necessidades específicas de domínio, a classe *ResolutionType* também é apresentada como genérica e extensível. Vale ressaltar que a propriedade *hasResolutionType* é de uso obrigatório, ou seja, para descrever um método de disputa é necessário determinar um métodos de resolução de problemas, logo a cardinalidade da propriedade é <*exactly1*>.

11. **Disclosure**: Esta classe apresenta as razões para a divulgação de dados, a frequência de divulgação e para quem os dados são divulgados através das propriedades *disclosureDescription*, *disclosureRate* e *discloseFor*, respectivamente. Os destinos da divulgação de dados incluem um *Recipient*, um *System* ou até um *Service*. A propriedade *disclosureDescription* tem cardinalidade <*max1*>, ou seja, seu uso não é obrigatório, mas caso seja utilizado, apenas uma é permitida. Por sua vez, a propriedade *disclosureRate* tem cardinalidade <*exactly1*>, pois é uma informação importante para determinar a frequência que as informações serão divulgadas, por isso, deve obrigatoriamente estar presente na classe *Disclosure*. Por fim, a propriedade *discloseFor* tem cardinalidade <*min0*>, isso porque em um determinado momento, é possível não querer divulgar informações para ninguém.
12. **TimeDuration**: essa classe representa instâncias de duração de tempo expressas pelas seguintes propriedades de dados: *hasDuration* e *hasDurationDescription*. *hasDuration* é ligada a um valor numérico inteiro e positivo, enquanto *hasDurationDescription* é ligada a uma descrição textual da unidade de tempo, a qual pode assumir os seguintes valores: {*milliseconds, seconds, minutes, hours, days, weeks, months, years*}. Tanto *hasDuration* quanto *hasDurationDescription* têm cardinalidade <*exactly1*>, isso porque é necessário que ambas informações estejam em conjunto para que uma duração de tempo seja descrita corretamente.
13. **CryptographyTechnique**: descreve as possíveis técnicas de criptografia no armazenamento de dados. É uma classe genérica e extensível para fins especiais de domínio e está vinculada à classe *StoragePrivacyPolicy* através da propriedade *cryptographyOnStorage*, que tem cardinalidade <max1>, indicando que é uma informação opcional, mas que no máximo uma instância de *CriptographyTechnique* é permitida.
14. **AccessList**: descreve as listas de acesso de um serviço específico. Um indivíduo de uma *AccessList* se associa a um ou mais destinatários através da propriedade *hasRecipient*, a qual tem cardinalidade <*min1*>, indicando que uma *AccessList* deve possuir pelo menos um *Recipient*. Uma *AccessList* se conecta a um *Service* através da propriedade *hasAccessList*. Atualmente, há duas subclasses de *AccessList*, sendo elas: *GrantList* e *DenyList*, que incluem configurações para conceder e negar acesso a um serviço, respectivamente. Como as listas de tipos podem variar de domínio para domínio, a classe *AccessList* é genérica, e se necessário, outras configurações de listas podem ser instanciadas, dependendo das necessidades da aplicação que for utilizá-la.

A ontologia *IoT-Priv* também engloba novas extensões que fizemos do vocabulário reutilizado da IoT-Lite, conforme descrito a seguir:

1. Novas informações são anexadas a um *iot-lite:Service*: uma descrição textual opcional (*hasServiceDescription*) e um valor *booleano* obrigatório que identifica se o serviço fornece ou manipula informações que identificam pessoalmente seu provedor (*isPersonalyIdentifiable*). A propriedade *hasServiceDescription* facilita a descoberta de serviços, enquanto a propriedade *isPersonalyIdentifiable* permite determinar se um serviço fornece, utiliza ou captura informações pessoais. Sabendo disso, os sistemas que gerenciam a privacidade podem tomar as providências cabíveis para evitar vazamento ou exposição destes dados pessoais.
2. Uma área de cobertura (*iot-lite:Coverage*) possui uma representação serializada no formato WKT (*Well-Known-Text*) para áreas poligonais [51] (*iot-priv:hasWKTPolygon*). Esse tipo de serialização é importante, pois diversas técnicas de processamento de dados espaciais necessitam que os dados estejam serializados em formato WKT para serem processados, como o *GeoSPARQL* [9].
3. Da mesma forma, um ponto geográfico (*geo:Point*) também possui uma representação serializada no formato WKT (*iot-priv:hasWKTPoint*).

3.3 Verificação e Avaliação da Ontologia *IoT-Priv*

Com o objetivo de demonstrar o uso da ontologia *IoT-Priv*, sua expressividade e extensibilidade, este capítulo propõe realizar a modelagem de dois cenários utilizando a ontologia *IoT-Priv*. Para isso, foram selecionados dois problemas de domínios distintos, a saber: i) compartilhamento de fotos; e ii) assistência médica domiciliar, os quais são detalhados nas Subseções 3.3.1 e 3.3.2, respectivamente. Por fim, este Capítulo apresenta uma avaliação da ontologia *IoT-Priv*, avaliando a sua complexidade lógica para fins de processamento e inferência.

Para compreensão do que se segue, apresentamos a notação utilizada na Figura 3.2, a qual é usada ao longo deste capítulo:

- Uma “caixa” representa um indivíduo. Este indivíduo é especificado em dois níveis: O retângulo superior representa as definições de tipos e nomes, enquanto o retângulo inferior representa as propriedades de dados literais.
- As *n-1* linhas do retângulo superior representam os tipos associados a este indivíduo. Um indivíduo pode ter mais de um tipo associado a ele, por exemplo, um indivíduo pode ser do tipo *Recipient* e *DataOwner* ao mesmo tempo. Por sua vez, a última linha do primeiro retângulo vem em negrito, representando a identificação do indivíduo, isto é, sua IRI.
- O retângulo inferior é composto por pares do tipo *propriedade : valor*. Neste campo são representadas as propriedades de dados literais, por exemplo, *collectionMethod-Description: "Collet GPS Data"*.

- Uma seta representa uma propriedade de objeto. Neste tipo de propriedade, são associados dois indivíduos (sujeito e objeto) por meio de uma propriedade (predicado).

Figura 3.2: *Exemplificação da notação utilizada neste capítulo.*

3.3.1 Verificação com Privacidade em Compartilhamento de Fotos

MyPhotos é um aplicativo para compartilhamento de fotos e recomendação de *hashtags* com base no contexto do usuário. Neste sentido, Gomes apresentou um cenário motivacional, onde explorava como a privacidade poderia ser aplicada na recomendação de *hashtags* para fotos [28]. A descrição do problema é apresentada a seguir:

> "Alexandre está na abertura dos Jogos Olímpicos do Rio 2016, na qual Alexandre e milhares de outros usuários estão publicando fotos e marcando-as com *hashtags* por meio de um aplicativo móvel, denominado *MyPhotos*. O aplicativo sugere *hashtags* de fotos com contexto semelhante (p.ex.: com local e instante semelhantes ao de novas fotos). [...] Quando Alexandre captura uma foto, são recomendadas as cinco *hashtags* mais utilizadas entre os usuários do aplicativo durante a festa de abertura. Depois do jogo, Alexandre vai a um shopping na cidade e continua utilizando o aplicativo, no entanto, por receio que outros usuários saibam a sua localização, Alexandre desativa o envio dos seus dados contextuais, dentre eles, a sua localização."
>
> Trecho adaptado de [28].

A Figura 3.3 apresenta a modelagem do cenário *MyPhotos*, conforme foi descrito acima, e a representação da Figura 3.3 utilizando a serialização TURTLE é apresentada no Apêndice A.1. As decisões de modelagem relacionadas a este cenário são detalhada a seguir.

Existe um sistema ou aplicativo, nomeado *:MyPhotos*, instanciado como um indivíduo do tipo *ssn:System*. O aplicativo MyPhotos tem um subsistema nomeado *:MyPhotos_Alexandre*, representando uma instância do aplicativo *MyPhotos* para o usuário *:Alexandre*. Este sistema é exposto na Web na forma de um serviço, nomeado *:MyPhotos_Alexandre_Service* do tipo *iot-lite:Service*, o qual tem a funcionalidade de recomendar *hashtags* com base no contexto do usuário, o que é descrito no serviço *:MyPhotos_Service* através da propriedade *iot-priv:serviceDescription*.

:Alexandre utiliza o aplicativo *:MyPhotos_Alexandre*, logo *:Alexandre* é o proprietário de seus dados privados, os quais são manuseados por este aplicativo,

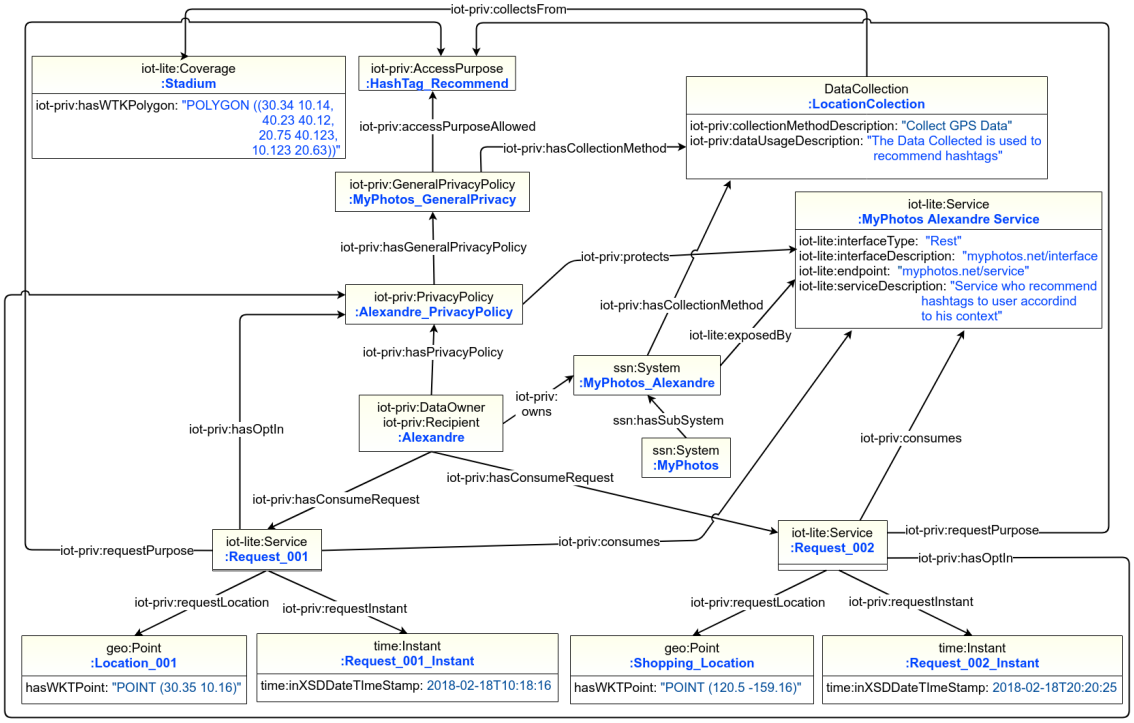

Figura 3.3: *Modelagem completa do aplicativo MyPhotos.*

sendo classificado como um indivíduo do tipo *DataOwner*. Como *:Alexandre* é um *DataOwner* no aplicativo *:MyPhotos_Alexandre*, ou seja, ele fornece dados contextuais para o aplicativo, consequentemente pode especificar sua própria política de privacidade, representada pelo indivíduo *:Alexandre_Privacy_Policy*, a qual está associada à sua configuração do aplicativo por meio da tripla (*:Alexandre_Privacy_Policy, iot-priv:protects, :MyPhotos_Alexandre_Service*).

Por padrão, o aplicativo *:MyPhotos* tem uma política de privacidade, a qual é definida no indivíduo *:MyPhotos_GeneralPrivacy*, instanciado com o tipo *iot-priv:GeneralPrivacyPolicy*, política esta que *:Alexandre* deve utilizar na sua definição de privacidade. Esta política especifica informações básicas para o sistema, por exemplo, o propósito de acesso com o qual o acesso ao serviço *:MyPhotos_Service* é permitido, no caso, o propósito *:HashTag_Recommend*, do tipo *iot-priv:AccessPurpose*, é único. Também especifica que o sistema coleta dados de localização através do indivíduo *:LocationCollection*, do tipo *iot-priv:DataCollection*.

No entanto, como é dito na especificação do cenário, *:Alexandre* pode complementar a política de privacidade padrão e adicionar informações, por exemplo, especificar lugares “seguros”, onde ele aceita que haja a coleta de seus dados, através da propriedade *iot-priv:collectsFrom*. No caso, foi especificado que os seus dados contextuais de localização podem ser coletados quando *Alexandre* estiver na área do estádio, o qual é definido como um indivíduo do tipo *iot-lite:Coverage*, com a propriedade *iot-lite:hasWKTPolygon* especificando a área do estádio, serializado em formato WKT.

:Alexandre também é um consumidor, pois ele também utiliza o serviço de recomendação de *hashtags* (*:MyPhotos_AlexandreService*), sendo assim, *:Alexandre* também foi classificado como um *iot-priv:Recipient*. As requisições ao serviço realizadas por *:Alexandre* são registradas em dois momentos distintos:

1. **:Request_001**: é um indivíduo do tipo *iot-priv:ConsumptionRequest*, representando o momento inicial, onde a localização de Alexandrew, no instante da requisição, estava dentro da área do estádio, pois a propriedade *iot-priv:requestLocation* apontava para "POINT (30.35 10.16)". Como esta requisição especifica o propósito de acesso igual ao permitido (*iot-priv:accessPurpose* do tipo *:HashTag_Recommend*) e concorda com a própria política de privacidade (*iot-priv:hasOptIn*), o acesso ao serviço e, consequentemente, o compartilhamento de dados contextuais são autorizados.
2. **:Request_002**: também é um indivíduo do tipo *iot-priv:ConsumptionRequest*, representando o momento posterior, onde a localização de Alexandre, no instante da requisição, está fora da área do estádio (*:Request_002 iot-priv:requestLocation :Shopping_Location*). Similar a *:Request_001*, como o propósito de acesso e o consentimento com a política de privacidade estão devidamente especificados e de acordo com a política, o acesso ao serviço também é autorizado. No entanto, difere da *:Request_001*, pois o compartilhamento de dados contextuais não acontece, devido ao que foi especificado na propriedade *iot-priv:collectsFrom*, que restringe o compartilhamento de dados contextuais somente quando dentro da área do estádio.

Por fim, vale ressaltar que não foi necessário estender a *IoT-Priv* para atender as demandas de privacidade deste aplicativo, mas apenas criar indivíduos de suas classes, isto porque, as demandas deste cenário, basicamente, giram em torno de privacidade baseada em localização.

3.3.2 Verificação com Privacidade em Assistência Médica Domiciliar

Ahamed et al. [4] descrevem um conjunto de situações que envolvem problemas de proteção de privacidade no domínio de assistência médica inteligente. Aqui, modelamos essas situações usando o conhecimento de privacidade disponível na ontologia *IoT-Priv*.

O principal objetivo é demonstrar não apenas o uso da ontologia, mas também sua expressividade e extensibilidade, quando aplicados em um cenário com maior complexidade, como o descrito a seguir:

> Andrew está sofrendo de pressão alta, e recentemente, ele sofreu um leve derrame também. Os médicos que o tratam acham que ele ainda não está fora de perigo e precisa estar em repouso sob supervisão médica

> por algum tempo, enquanto ele estiver gradualmente voltando à sua vida normal em casa.
> Andrew instalou um sistema de monitoramento no servidor central de sua casa. O sistema monitora suas atividades e seus parâmetros fisiológicos. Com o apoio do sistema, Andrew pode aproveitar sua vida em casa e, ao mesmo tempo, estar sob os cuidados de médico, enfermeiro e psiquiatra. O sistema de saúde tem a capacidade de coletar dados sobre o comportamento e a condição física de Andrew e interpretar esses dados para tirar conclusões sobre seu estado de saúde. Esses dados são registrados localmente em seu histórico médico, os quais podem ser acessados pelos médicos e enfermeiros de Andrew, quando necessário.
> Quando o sistema deduzir que Andrew está sofrendo um problema de saúde, ele emitirá um alarme e notificará os médicos e parentes.
> Andrew é o proprietário das informações de seu histórico médico e tem o direito de fornecer acesso a outras pessoas. Ele não tem o direito de modificar essa informação; esse direito é do sistema e dos médicos responsáveis.
> Andrew não quer compartilhar suas informações médicas com outras pessoas e está preocupado com a capacidade de seus visitantes residenciais também acessarem seu histórico médico, já que seu sistema de monitoramento de saúde permite que novos dispositivos sejam integrados ao ambiente, sendo assim, qualquer pessoa que entre em sua casa com um dispositivo em mãos poderia ter acesso às suas informações confidenciais, se estas não estiverem devidamente protegidas, então Andrew quer que o sistema aplique políticas de autenticação e controle de acesso e peça aprovação, antes de conceder acesso a conteúdo e serviços em sua casa.
> Embora sistemas ubíquos tenham contribuído com mudanças radicais nas unidades de saúde, também foram levantadas preocupações sobre possíveis violações de privacidade, pois informações médicas confidenciais são reunidas em pontos de armazenamento específicos (o servidor ou rede do paciente, o hospital) e podem estar sujeitas ao ponto único de ataques de falha, como por exemplo, o serviço que disponibiliza as informações. Além disso, uma vez que esta informação está em formato digital, pode ser copiada e transmitida praticamente a custo zero. Finalmente, o indivíduo não tem percepção direta do uso dessa informação, embora esteja diretamente relacionada a ele.
>
> Trecho adaptado de [4], tradução própria.

Por se tratar de um cenário mais complexo, foi necessário estender a ontologia *IoT-Priv*, descrevendo conceitos apropriados para o cenário em questão. Neste sentido, utilizando uma abordagem de duas camadas tradicional, onde importamos a ontologia *IoT-Priv* e a estendemos, construindo a ontologia que chamamos de *IoT-Priv-M*, isto é, a ontologia *IoT-Priv* estendida e adaptada para o cenário médico, onde foram adicionados os seguintes conceitos:

- ***iot-priv:AnonymizationTechnique***: como mostra a Figura 3.4, o conceito foi estendido, de modo que passa a existirem duas subclasses que derivam de *iot-priv:AnonymizationTechnique*, sendo elas: *iot-priv:WithoutAnonymization*, a qual já

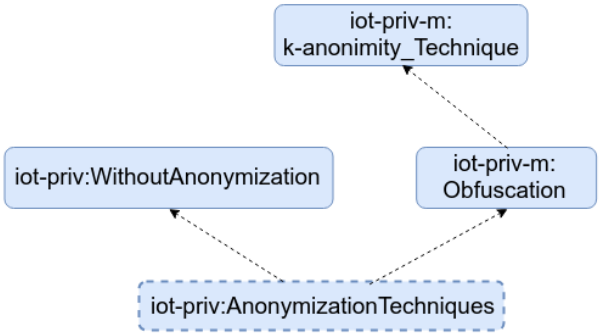

Figura 3.4: *Extensão do conceito iot-priv:AnonymizationTechnique.*

era definida na própria ontologia *IoT-Priv*; e a classe *Obfuscation*, a qual foi definida na *IoT-Priv-M*, representando um conjunto de técnicas de anonimização baseada em ofuscação. Além disso, *k-anonimity_Technique* também foi criada como uma subclasse de *Obfuscation*, descrevendo que*k-anonimity* é uma técnica de anonimização de ofuscação.

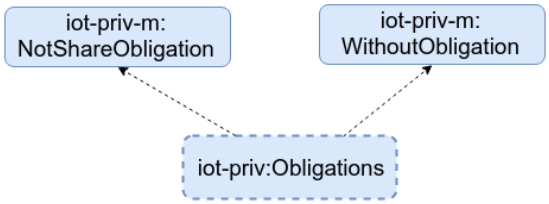

Figura 3.5: *Extensão do conceito iot-priv:Obligations.*

- ***iot-priv:Obligations***: como mostra a Figura 3.5, o conceito foi estendido, de modo que passa a existirem duas subclasses que derivam de *iot-priv:Obligations*, sendo elas: *iot-priv:WithoutObligation*, a qual já era definida na própria ontologia *IoT-Priv*; e *NotShareObligation*, representando a obrigação de não compartilhar ou repassar os dados privados obtidos.

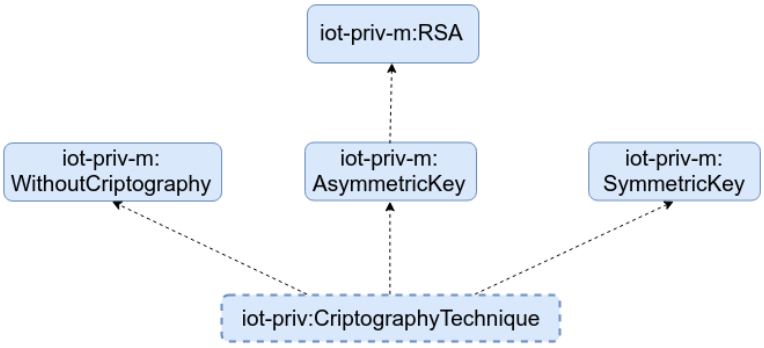

Figura 3.6: *Extensão do conceito iot-priv:CriptographyTechnique.*

- ***iot-priv:CriptographyTechnique***: como mostra a Figura 3.6, o conceito foi estendido, de modo que passa a existirem três subclasses que derivam de *iot-*

priv:CriptographyTechnique, sendo elas: *iot-priv:WithoutCriptography*, a qual já era definida na própria ontologia *IoT-Priv*; *iot-priv-m:SymmetricKey*, representando as técnicas de criptografia que utilizam chave simétrica; e *iot-priv-m:AsymmetricKey*, representando as técnicas de criptografia que utilizam chave assimétrica. Por fim, também foi criada a classe *iot-priv-m:RSA*, a qual foi definida como uma subclasse de *iot-priv-m:AsymmetricKey*, especificando que RSA é uma técnica de criptografia assimétrica.

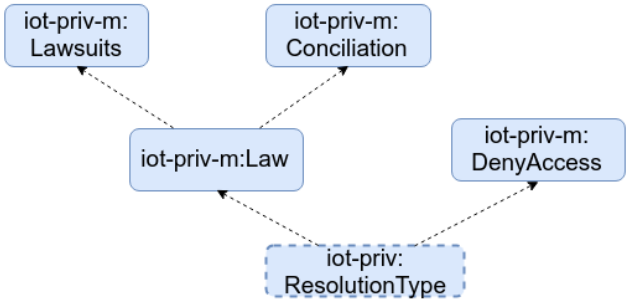

Figura 3.7: *Extensão do conceito iot-priv:ResolutionType.*

- ***iot-priv:ResolutionType***: como mostra a Figura 3.7, o conceito foi estendido, de modo que passa a existirem três subclasses que derivam de *iot-priv:ResolutionType*, sendo elas: *iot-priv-m:DenyAccessList*, representando que os conflitos serão resolvidos adicionando o infrator na lista de acesso que nega acesso ao serviço; e *iot-priv-m:Law*, representando que os conflitos serão resolvidos na lei. Por fim, também foram criadas duas classes, *iot-priv-m:Lawsuits* e *iot-priv-m:Conciliation*, indicando que na lei, os conflitos poderão ser resolvidos por meio de ações judiciais ou conciliação, respectivamente.

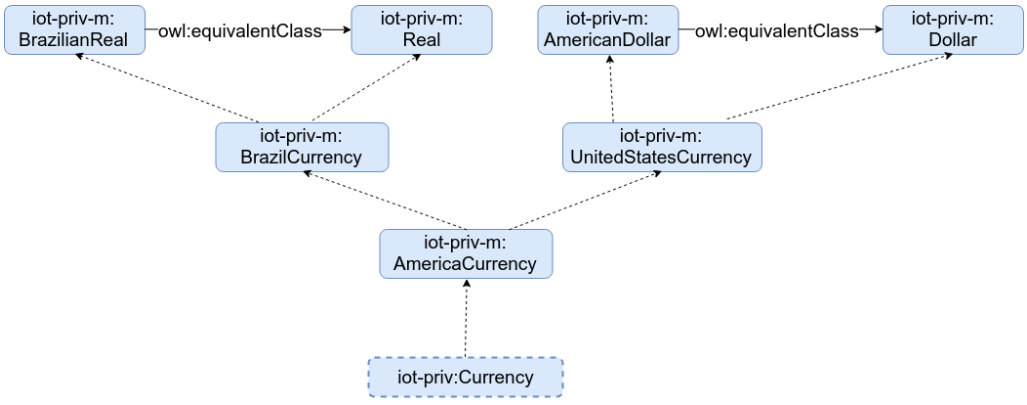

Figura 3.8: *Extensão do conceito iot-priv:Currency.*

- ***iot-priv:Currency***: como mostra a Figura 3.8, o conceito foi estendido, de modo que passa a existir uma subclasse que deriva de *iot-priv:Currency*, sendo ela a classe *iot-priv-m:AmericaCurrrency*, representando as moedas do continente americano. Além disso, foram criadas duas classes, as quais derivam de *iot-priv-m:AmericaCurrency*,

sendo elas: *iot-priv-m:BrazilCurrency*, representando a moeda do Brasil; e *iot-priv-m:UnitedStatesCurrency*, representando a moeda dos Estados Unidos. Além disso, temos duas classes: *iot-priv-m:BrazilianReal* e *iot-priv-m:Real*, as quais derivam da classe *iot-priv-m:BrazilCurrency*, representando a moeda atual do Brasil, e como representam a mesma moeda, são interligadas pela propriedade *owl:equivalentClass*, como na tripla (*iot-priv-m:BrazilianReal, owl:equivalentClass, iot-priv-m:Real*). Da mesma forma, as classes *iot-priv-m:AmericanDollar* e *iot-priv-m:Dollar* derivam da classe *iot-priv-m:UnitedStatesCurrency*, e também são interligadas pela propriedade *owl:equivalentClass*.

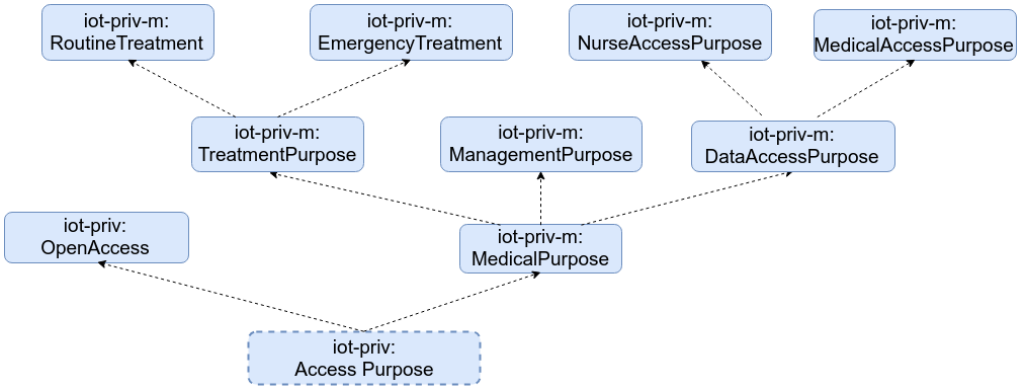

Figura 3.9: *Extensão do conceito iot-priv:AccessPurpose.*

- ***iot-priv:AccessPurpose***: como mostra a Figura 3.9, o conceito foi estendido, de modo que passa a existirem duas subclasses que derivam de *iot-priv:AccessPurpose*, sendo elas: *iot-priv:OpenAccess*, a qual já era definida na própria ontologia *IoT-Priv*; e *iot-priv-m:MedicalPurpose*, representando propósitos médicos de forma geral. A seguir foram criadas três outras classes, as quais derivam de *iot-priv-m:MedicalPurpose*, sendo elas: *iot-priv-m:ManagementPurpose*, representando o propósito de gestão hospitalar; *iot-priv-m:TreatmentPurpose*, representando o propósito de tratar pacientes, o qual é dividido em duas subclasses, sendo elas o tratamento de rotina (*iot-priv-m:RoutineTreatment*) e o tratamento de emergência (*iot-priv-m:EmergencyTreatment*); e, por fim, *iot-priv-m:DataAccessPurpose*, representando o propósito de acessar os dados médicos de um paciente, o qual também é dividido em duas subclasses, sendo elas o acesso por enfermeiras (*iot-priv-m:NurseAccessPurpose*) e o acesso por médicos (*iot-priv-m:MedicalAccessPurpose*).

Uma vez que estendemos a ontologia *IoT-Priv*, foi possível modelar o cenário descrito. Por se tratar de um cenário mais complexo, o mesmo é dividido e exemplificado por partes, as quais são detalhadas a seguir. Após isso, o modelo completo resultante também é apresentado na Figura 3.21.

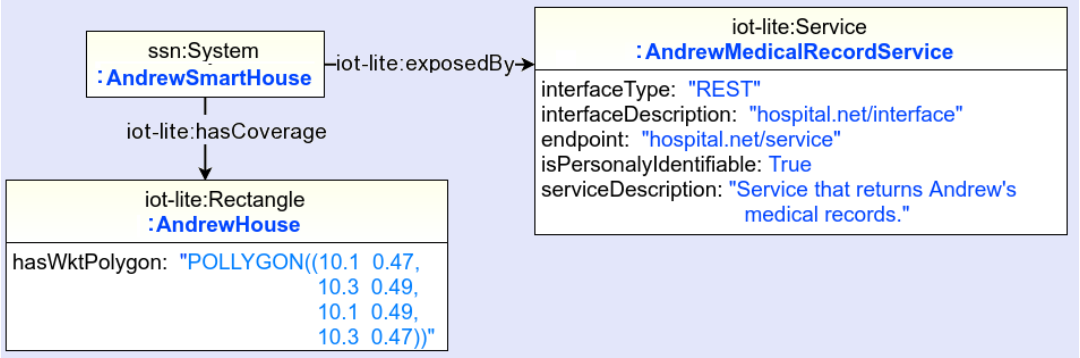

Figura 3.10: *Modelagem do cenário de assistência médica domiciliar – Parte (1)*

1. Como mostra a Figura 3.10, é realizada uma especificação básica de um sistema IoT com a configuração do sistema, do serviço exposto e da área de cobertura do sistema. Neste caso, a casa de Andrew (*:AndrewHouse*) é o sistema que expõe um serviço (*:AndrewMedicalRecordService*) cuja função é enviar os dados de saúde de Andrew aos médicos e enfermeiros. Além disso, neste serviço são determinadas URL's para acesso à descrição da interface (*iot-lite:interfaceDescription*) e ao *endpoint* do serviço (*iot-lite:endpoint*) e também é especificado que o serviço manipula dados pessoais (*iot-priv:isPersonallyIdentifiable*). Este sistema tem uma área de cobertura, no caso, a própria casa de Andrew (*:AndrewHouse*). Por fim, uma serialização poligonal baseada no formato WKT é associada à casa de Andrew, utilizando a propriedade *iot-priv:hastWKTPolygon*.

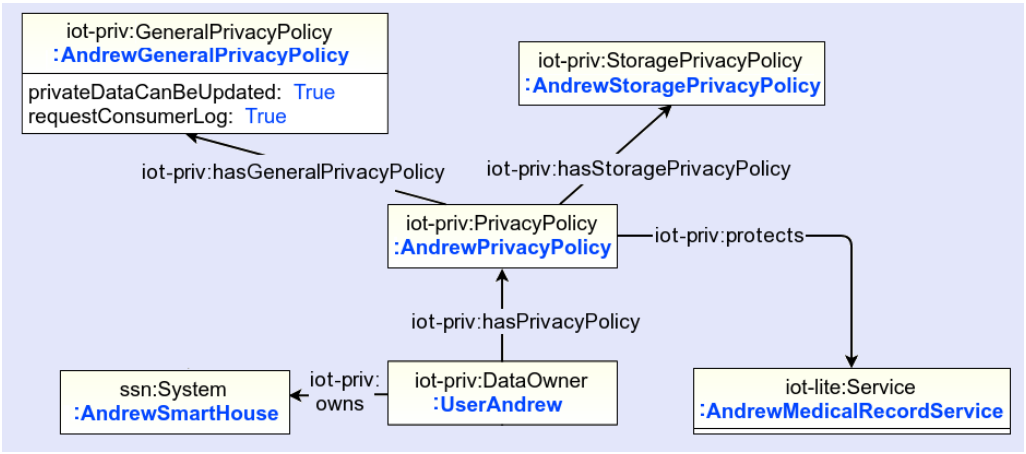

Figura 3.11: *Modelagem do cenário de assistência médica domiciliar – Parte (2)*

2. A Figura 3.11 apresenta a modelagem do proprietário de um sistema, de sua política de privacidade e a proteção de serviços por esta política. No cenário apresentado, a tripla *(:UserAndrew, iot-priv:owns, :AndrewSmartHouse)* modela a propriedade do sistema. Por sua vez, a tripla *(:UserAndrew, iot-priv:hasPrivacyPolicy, :AndrewPri-*

vacyPolicy) determina a política de privacidade de Andrew, a qual é composta de política de privacidade para propósitos gerais (*:AndrewGeneralPrivacyPolicy*) e a política com propósito de armazenamento (*:AndrewStoragePrivacyPolicy*). Por fim, a política de privacidade de Andrew protege o serviço que pode enviar seus dados de saúde para profissionais no hospital, representado pela tripla *(:AndrewPrivacyPolicy, iot-priv:protects, :Andrew MedicalRecordService)*.

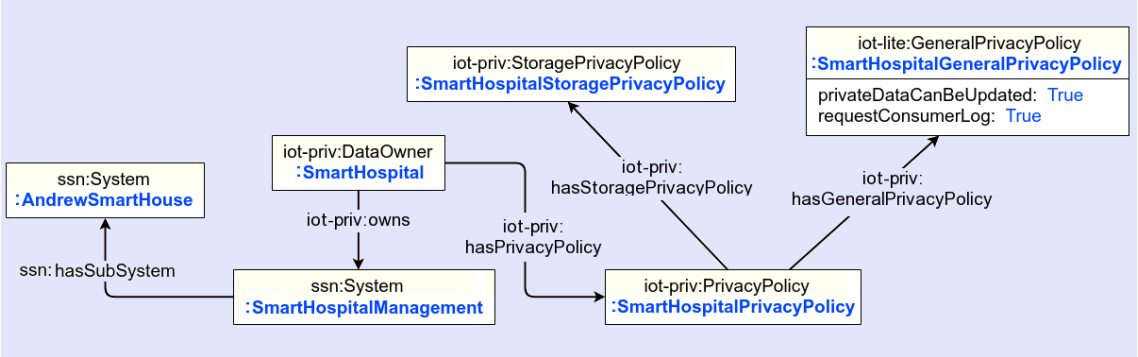

Figura 3.12: *Modelagem do cenário de assistência médica domiciliar – Parte (3)*

3. A Figura 3.12 ilustra um cenário, onde o *:SmartHospital* é o proprietário do sistema *:SmartHospitalManagement*. Também vemos que *:AndrewSmartHouse*, o qual foi exemplificado na Figura 3.10, é modelado como um subsistema do *:SmartHospitalManagement*, construindo a relação de composição entre os sistemas. Por sua vez, a tripla *(:SmartHospital, iot-priv:hasPrivacyPolicy, :SmartHospitalPrivacyPolicy)* determina a política de privacidade do *Smart Hospital*, a qual também é composta de políticas de privacidade para propósitos gerais (*:SmartHospitalGeneralPrivacyPolicy*) e as políticas com propósito de armazenamento (*:SmartHospitalStoragePrivacyPolicy*). A política de proposito geral ainda indica que os dados privados de Andrew podem ser atualizados (*iot-priv:privateDataCanBeUpdated*), mas para isso, são requisitados *log's* dos consumidores (*iot-priv:requestConsumerLog*).
4. A Figura 3.13 representa a especificação de técnicas de anonimização e de criptografia para as necessidades de Andrew. Uma instância da classe *iot-priv-m:k-anonymity_Technique*, que é uma extensão da classe *iot-priv:AnonymizationTechinques*, vincula-se a política de privacidade genérica de Andrew, como em *(:AndrewGeneralPrivacyPolicy, iot-priv:useAnonymizationTechnique, :k-anonymity_Technique)*. Da mesma forma, um indivíduo da classe *iot-priv-m:RSA*, que é uma extensão da classe *iot-priv:CriptographyTechnique*, vincula-se à política de privacidade de armazenamento de Andrew, como em *(:AndrewStoragePrivacyPolicy, iot-priv:cryptographyOnStorage, :RSA_Cryptography)*.

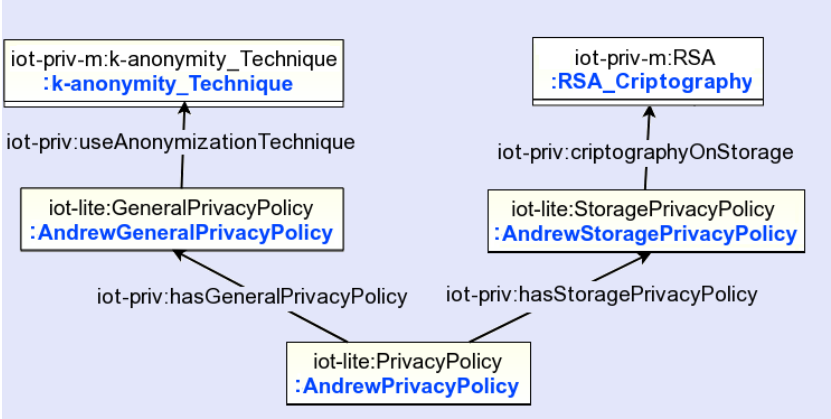

Figura 3.13: *Modelagem do cenário de assistência médica domiciliar – Parte (4)*

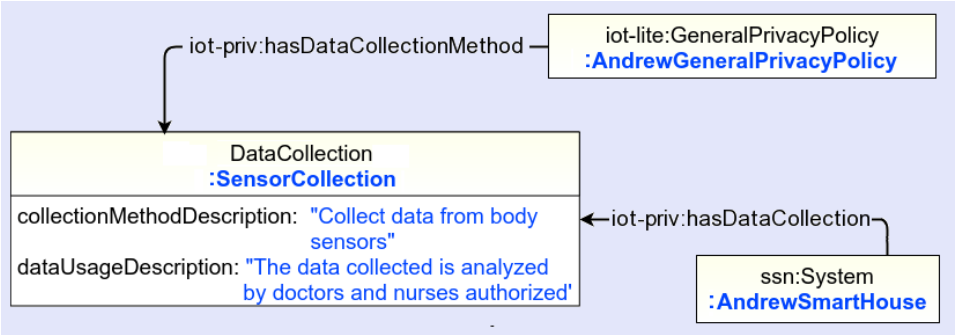

Figura 3.14: *Modelagem do cenário de assistência médica domiciliar – Parte (5)*

5. A Figura 3.14 demonstra como o sistema de Andrew coleta dados. Um indivíduo da classe *DataCollection* se conecta ao sistema de Andrew, bem como à sua política de privacidade genérica. Esta última informação é representada como (*:AndrewGenericPrivacyPolicy, iot-priv:hasDataCollectionMethod, :SensorCollection*).

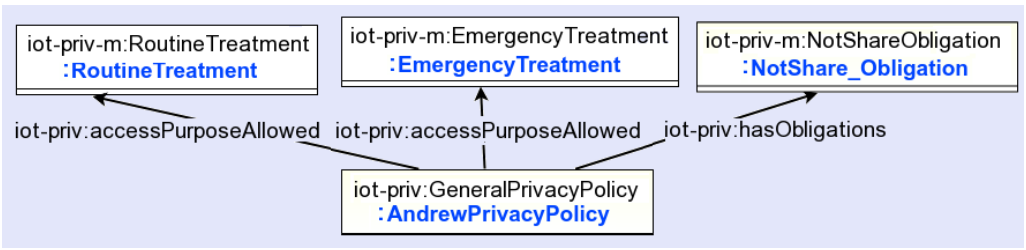

Figura 3.15: *Modelagem do cenário de assistência médica domiciliar – Parte (6)*

6. A Figura 3.15 ilustra como a a política geral de privacidade de Andrew se vincula a propósitos e obrigações de acesso. No cenário, identificamos dois tipos de propósito para acesso aos dados de Andrew: para tratamentos de rotina (*:RoutineTreatment*) e de emergência (*:EmergencyTreatment*t), os quais são do tipo *iot-priv-m:RoutineTreatment* e *iot-priv-m:EmergencyTreatment*, respectivamente, ou seja, são extensões da classe *iot-priv:AccessPurpose*. Além disso, os que têm acesso aos dados de Andrew não podem divulgá-los a terceiros; a classe *iot-priv-m:NotShareObligation* representa este tipo de obrigação, que é uma extensão da classe *iot-priv:Obligation*, e vincula-se a política de privacidade genérica de Andrew, como em (*:AndrewPrivacyPolicy, iot-priv:hasObligations, :NotShare_Obligation*).

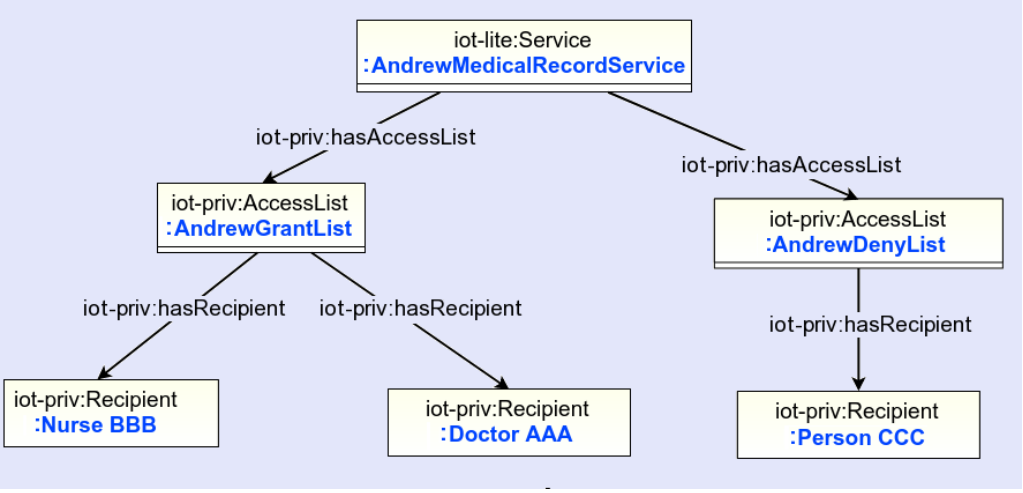

Figura 3.16: *Modelagem do cenário de assistência médica domiciliar – Parte (7)*

7. A Figura 3.16 representa a modelagem de consumidores ou destinatários cujo acesso ao serviço de Andrew (*:AndrewMedicalRecordService*) é concedido ou negado. *:AndrewGrantList* é um indivíduo da classe *iot-priv:AccessList*, que inclui aqueles que têm acesso ao serviço, como o enfermeiro e o médico particular de Andrew (*:Nurse_BBB* e *:Doctor_AAA*, respectivamente). Da mesma forma, *:AndrewDenyList* vincula-se a destinatários com acesso negado ao serviço de Andrew, por exemplo, uma pessoa em particular (*:Person_CCC*).
8. A Figura 3.17 apresenta informações sobre para quem se pode divulgar os dados de Andrew e com que frequência isso pode acontecer. No modelo, *:SmartHouseDisclose* é um indivíduo da classe *iot-priv:Disclosure*, vinculado à Política de Privacidade de Andrew através da propriedade *iot-priv:discloses*. No modelo apresentado, os indivíduos *:SmartHospitalManagement* e *:AndrewsWife* representam as entidades para quem os dados de Andrew são divulgados, enquanto o indivíduo *:DisclosureRate* transmite a frequência de disponibilização dos dados de Andrew, ou seja, uma vez por mês.

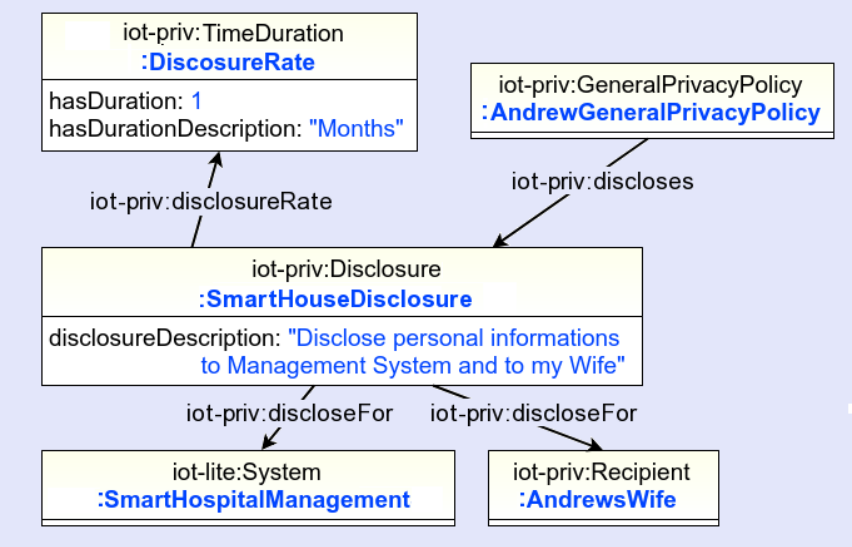

Figura 3.17: *Modelagem do cenário de assistência médica domiciliar – Parte (8)*

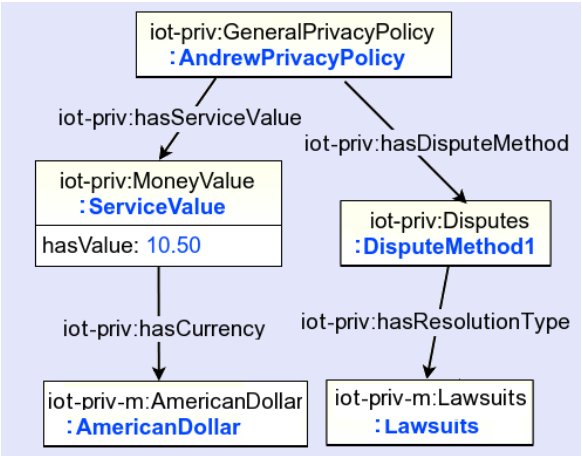

Figura 3.18: *Modelagem do cenário de assistência médica domiciliar – Parte (9)*

9. A Figura 3.18 representa um valor de recompensa fictício (por exemplo, 10,5 USD) para o serviço de registro médico de Andrew. Essa modelagem inclui *:ServiceValue* como instância da classe *iot-priv:MoneyValue* e *:AmericanDollar* como instância da classe *iot-priv-m:AmericanDollar*, ambas vinculadas à política de privacidade de Andrew. Vale ressaltar que *iot-priv-m:AmericanDollar* é uma extensão da classe *iot-priv:Currency*. Além disso, este trecho do modelo ainda mostra um procedimento de disputa modelado no caso da violação de privacidade de Andrew, através da tripla (*:DisputeMethod1, iot-priv:hasResolutionType, :Lawsuits*), onde *:Lawsuits* é um indivíduo do tipo *iot-priv-m:Lawsuits*, que é uma extensão da classe *iot-priv:ResolutionType*.
10. A Figura 3.19 descreve as políticas de privacidade do *Smart Hospital*. A política de armazenamento do hospital manipula dados confidenciais, por isso, uti-

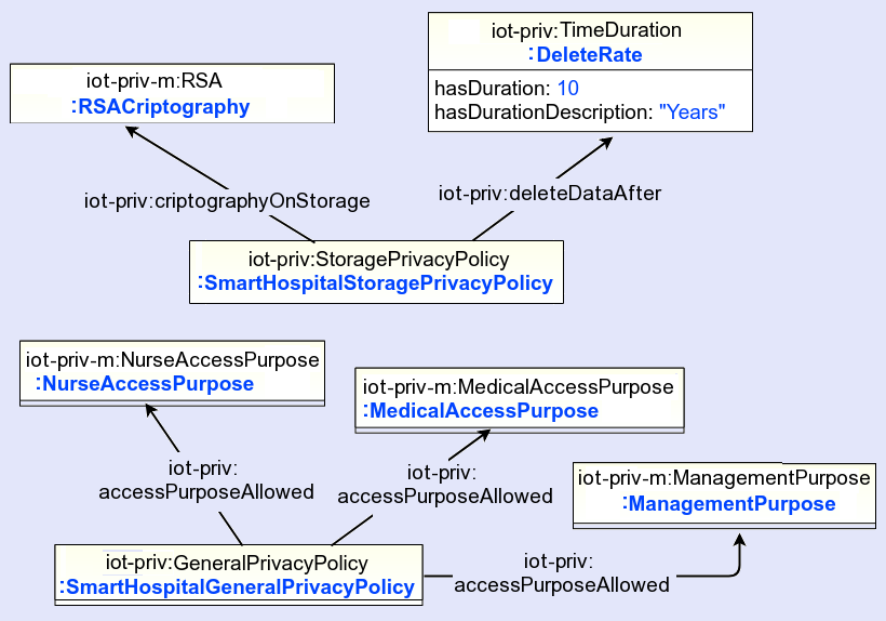

Figura 3.19: *Modelagem do cenário de assistência médica domiciliar – Parte (10)*

liza criptografia no armazenamento dos dados, e estes só podem ser excluídos após o período mínimo de 10 anos, representados pelos indivíduos *:RSACriptography* e *:DeleteRate*, respectivamente. Vale ressaltar que o indivíduo *:RSACriptography* é uma instância da classe *iot-priv-m:RSA*, que é uma extensão da classe *iot-priv:CriptographyTechnique*. Ainda neste cenário, a política de privacidade genérica define 3 propósitos de acesso, com os quais o acesso ao sistema do hospital é permitido, sendo eles, acesso de enfermeiras (*:NurseAccessPurpose*), de médicos (*:MedicalAccessPurpose*) e dos gerentes do hospital (*:ManagementPurpose*), instâncias das classes *iot-priv-m:NurseAccessPurpose*, *iot-priv-m:MedicalAccessPurpose* e *iot-priv-m:ManagementPurpose*, todas extensões da classe *iot-priv:AccessPurpose*.

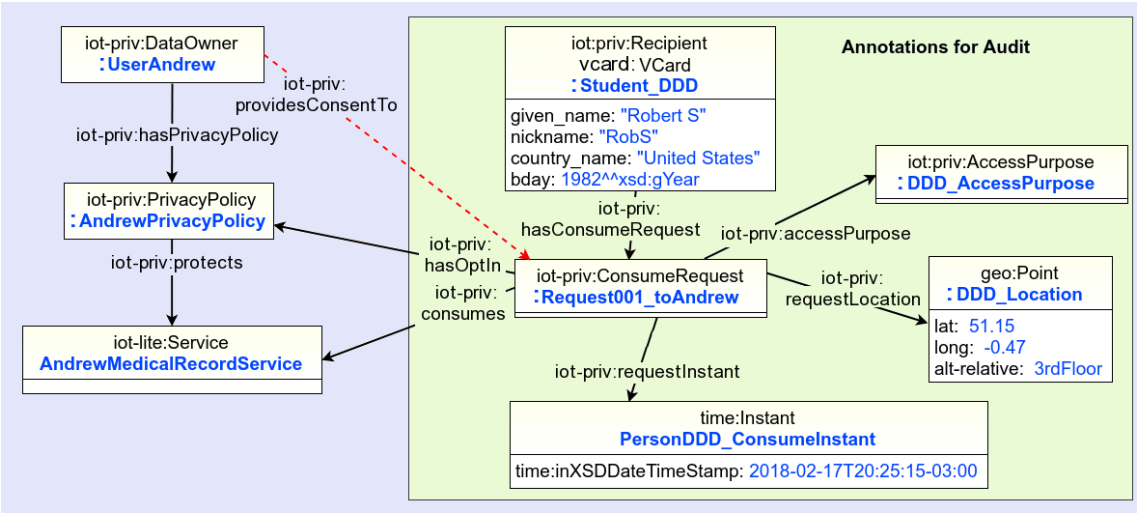

Figura 3.20: *Modelagem do cenário de assistência médica domiciliar – Parte (11)*

11. A Figura 3.20 descreve como Andrew consente com as solicitações de consumo de seu serviço de registro médico. Um destinatário específico (*:Student_DDD*) emite uma solicitação de consumo de dados (*:Request001_toAndrew*), a qual deve estar em conformidade com a política geral de privacidade que protege o serviço de Andrew através da propriedade *iot-priv:hasOptIn*. Por fim, a uma requisição de consumo são registradas informações para auditoria, incluindo o instante de tempo e a localização da geração da solicitação, representados pelos indivíduos *:PersonDDD_ConsumeInstant* e *:DDD_Location*, respectivamente. Observe ainda a representação das coordenadas geográficas do indivíduo *:DDD_Location*, se necessário, a serialização WKT também pode ser aplicada como mostrado no trecho da Figura 3.10. A solicitação deste indivíduo também encapsula o propósito de acesso (*:DDD_AccessPurpose*). Como um *Data Owner*, Andrew deve explicitamente consentir com esta solicitação através da propriedade *provideConsentTo*. Neste exemplo, o acesso não é permitido, já que o *:DDD_Access Purpose* não é um dos propósitos de acessos registrados na política de privacidade, como foi ilustrado na Figura 3.15.

A Figura 3.21 apresenta a modelagem completa do cenário de *Healthcare*, conforme foi descrito acima. Vale ressaltar que as mesmas considerações feitas para as Figuras 3.10 a 3.20 valem para a Figura 3.21, que nada mais é que a composição das demais Figuras já apresentadas. A representação da Figura 3.21 utilizando a serialização TURTLE é apresentada no Apêndice A.2.

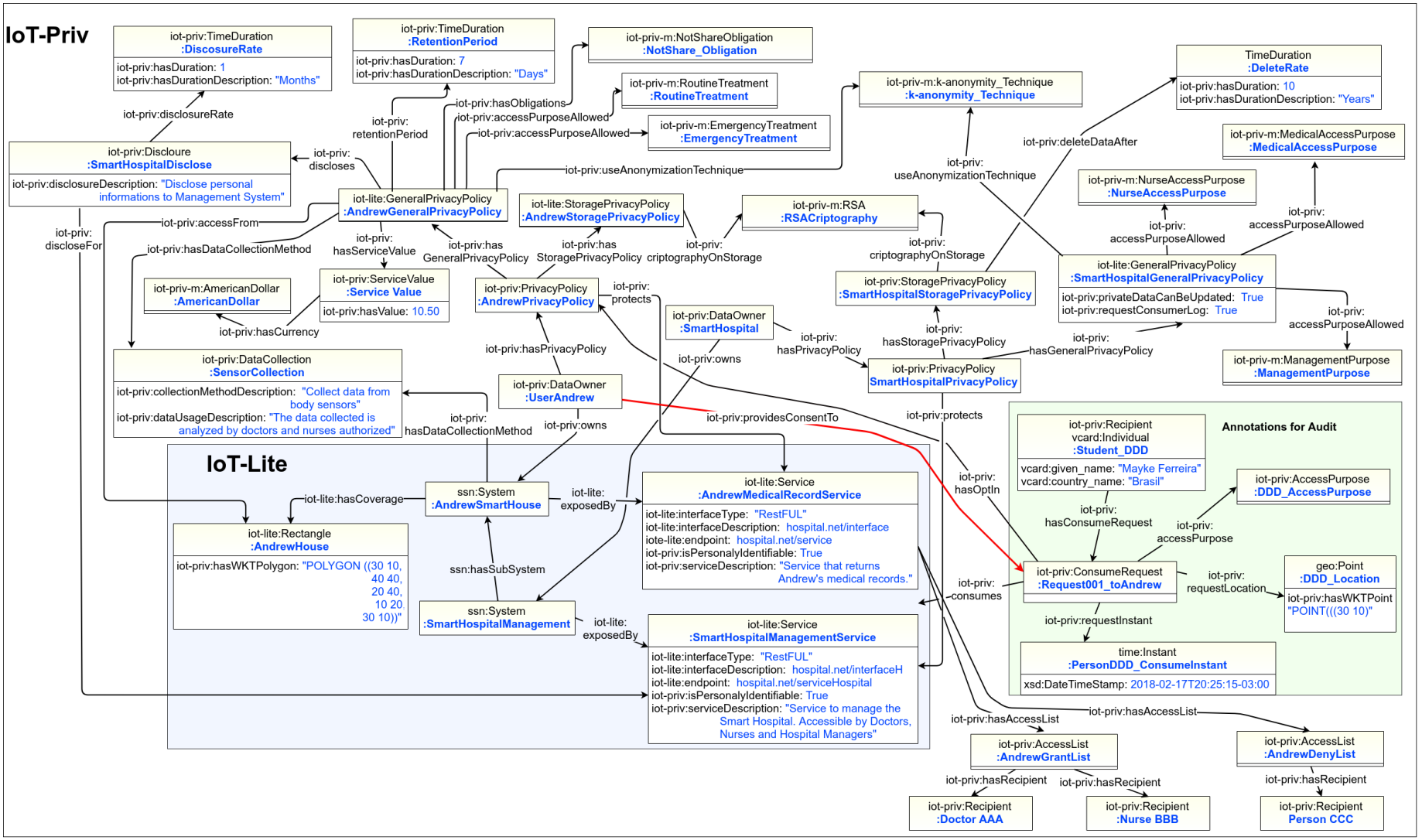

Figura 3.21: *Modelagem completa do cenário de assistência médica domiciliar.*

3.3.3 Avaliação da *IoT-Priv*

Como não há uma abordagem definitiva para a avaliação de ontologias, deve-se levar em consideração alguns fatores para orientar a escolha do método adequado: o aspecto da ontologia que se deseja avaliar, a aplicação que utiliza ou irá utilizar a ontologia e o propósito da avaliação. Como a ontologia *IoT-Priv* reutilizou a ontologia IoT-Lite e agregou uma camada de privacidade à mesma, umas das principais preocupações por trás do design e desenvolvimento da *IoT-Priv* é manter a propriedade de *leveza* fornecida pela ontologia reutilizada.

Para realizar esta avaliação, optamos por utilizar dois tipos de métricas, as quais são indicadores da dificuldade em uma aplicação trabalhar com uma determinada ontologia: i) as métricas estruturais, tais como quantidades de classes, propriedades e axiomas; ii) as métricas que refletem a complexidade da ontologia, como a lógica de descrição.

Neste sentido, optamos por utilizar a ferramenta *Protégé* com o *plugin Ontology Metrics*, o que foi suficiente para as nossas necessidades. Esta ferramenta se trata de um ambiente interativo para construção de ontologias, a qual oferece, dentre outras coisas, uma biblioteca de *plugins*, contendo uma vasta gama de funcionalidades [50], dentre elas, o *plugin Ontology metrics*, utilizado para realizar a extração das métricas supracitadas.

A Tabela 3.2 apresenta as métricas extraídas de ambas ontologias, as quais são explicadas a seguir:

- Axiomas: representa qualquer "conhecimento", isto é, fatos, regras ou uma declarações representadas ontologia. Estes conhecimentos são classificados em axiomas declarativos ou axiomas lógicos.
  - Axiomas Declarativos: representam a contagem da propriedades de anotações (*owl:AnnotationProperty*, aquelas que servem para documentar e descrever metadados da ontologia), das propriedades de dados (*owl:DatatypeProperty*, aquelas que interligam uma classe a um literal), das propriedades de objetos (*owl:ObjectProperty*, aquelas que interligam duas classes) e das classes (*owl:Class*).
  - Axiomas Lógicos: representam, entre outras coisas: as restrições de domínio e imagem (*rdfs:Domain e rdfs:Range*, respectivamente), as definições de hierarquia (rdfs:subClassOf e rdfs:subPropertyOf), as definições de cardinalidade (*owl:maxCardinality, owl:minCardinality e owl:cardinality*), além das especializações de propriedades (tais como *owl:FunctionalProperty, owl:InverseProperty*, entre outras).
- Indivíduos: representam a quantidade de indivíduos que foram instanciados na própria ontologia.

- **Lógica de Descrição**: representa a lógica de descrição associada à ontologia. Essa métrica é um indicativo da dificuldade de se processar ou realizar inferências sobre os dados anotados com uma determinada ontologia.

Tabela 3.2: *Métricas de tamanho e complexidade das ontologias IoT-Lite e IoT-Priv.*

Ontologias / Métricas	IoT-Lite	IoT-Priv	Diferenças entre as Ontologias	
			Quantidade	%
Triplas	297	894	597	201%
Axiomas Declarativos	55	136	81	147%
Axiomas Lógicos	53	226	173	314%
Indivíduos	0	0	0	0%
Lógica de Descrição	$\mathcal{ALUI}(\mathcal{D})$	$\mathcal{ALUIQ}(\mathcal{D})$	$\mathcal{Q}$	

Ao analisar as métricas da Tabela 3.2, percebemos o notável aumento no tamanho das ontologias, através da quantidade de triplas (de 297 para 894), indicando que a *IoT-Priv* é pelo menos 3 vezes mais densa conceitualmente que a IoT-Lite, uma vez que este aumento na quantidade de triplas é naturalmente refletido em um aumento na quantidade de axiomas.

Consequentemente, houve aumento na quantidade de axiomas declarativos (de 55 para 136), indicando que a IoT-Priv tem, em média, 2.5 vezes mais classes e propriedades quando comparada à IoT-Lite, o que é um resultado esperado, visto o notável aumento na quantidade de triplas. No entanto, o aumento na quantidade de axiomas lógicos (de 53 para 226) nos permite deduzir o quanto a *IoT-Priv* é aproximadamente 4 vezes mais rica conceitualmente, se comparada à IoT-Lite, uma vez que a riqueza de uma ontologia está nos seus relacionamentos e restrições.

No entanto, embora os axiomas descritivos e lógicos tenham uma influência significativa na complexidade lógica de uma ontologia, a complexidade lógica da ontologia *IoT-Priv* sofreu poucas alterações. Observe que a expressividade lógica variou de $\mathcal{ALUI}(\mathcal{D})$ para $\mathcal{ALUIQ}(\mathcal{D})$, indicando que o aumento na complexidade lógica da ontologia *IoT-Priv* em relação a IoT-Lite é causada pelo emprego de restrições de cardinalidade (do Inglês, ***Q**ualified cardinality restrictions*).

Usamos muito esse tipo de axiomas lógicos na ontologia *IoT-Priv*, restringindo o número de valores permitidos para uma propriedade, por exemplo, indicando que um proprietário de dados deve ter exatamente uma política de privacidade associada a ele, e esta política deve ter no máximo uma definição de políticas de privacidade de armazenamento. Além disso, tanto $\mathcal{ALUI}(\mathcal{D})$ como $\mathcal{ALUIQ}(\mathcal{D})$ pertence a classe de complexidade computacional *PSPACE-Complete*[1], indicando que a complexidade

[1] Fonte: ferramenta da Universidade de Manchester: http://www.cs.man.ac.uk/~ezolin/dl/

computacional de ambas as ontologias se manteve inalterada, mesmo com o incremento de (Q) na complexidade lógica. Portanto, ao considerar o tamanho, a lógica de descrição, bem como a complexidade computacional associada a esta lógica, percebemos que mesmo com os novos conceitos de privacidade, a característica de leveza da IoT-Lite foi preservada, o que era um de nossos objetivos iniciais.

Com relação às métricas de qualidade e usabilidade de Gruber [30], percebemos que a ontologia IoT-Priv é **clara**, pois todas as definições de conceitos, propriedades e restrições estão devidamente documentadas na Seção 3. Além disso, como foi demonstrado na verificação com privacidade em assistência médica domiciliar (Seção 3.3.2), a ontologia *IoT-Priv* comprovou ser **extensível**, isso graças as nossas decisões de projetos, onde incluímos conceitos genéricos para que possam ser estendidos e especializados sem que seja necessário a revisão das definições já existentes. Por fim, a quantidade de indivíduos se manteve inalterada, e tanto a ontologia IoT-Lite quanto a *IoT-Priv* possuem 0 indivíduos, como foi ilustrado na Tabela 3.2. Isso reflete o **compromisso ontológico** da *IoT-Priv*, evitando fazer afirmações sobre o cenário que está sendo modelado, mas sim fornecendo conceitos e relações para que os usuários possam fazer suas próprias afirmações.

3.4 Considerações Finais

A *IoT-Priv* reusa e estende a ontologia *IoT-Lite*, portanto, todos os conceitos e relacionamentos presentes no IoT-Lite também estão presentes na *IoT-Priv*. Desta forma, a *IoT-Priv* além de fornecer conceitos e relacionamentos suficientes para descrever e representar cenários da IoT, também fornece uma camada extra na qual os conceitos e as relações associados à privacidade são fornecidos. Sendo assim, a ontologia *IoT-Priv* é capaz de representar a privacidade no contexto da Internet das Coisas.

Graças a nossa decisão de projeto de estender a *IoT-Lite* e incluir conceitos de privacidade mais genéricos, bem como pontos de extensão na ontologia, avançamos em direção a uma ontologia de privacidade independente de domínio, possibilitando que os usuários possam estendê-la e adequá-la para seus propósitos, mas sem perdermos a característica de leveza que defendemos.

Consequentemente, também realizamos uma verificação da ontologia proposta, demonstrando como a *IoT-Priv* pode atender aos requisitos de privacidade e ser aplicada em dois domínios de aplicações distintos, sendo eles o compartilhamento de fotos e assistência médica domiciliar. Também demonstramos como é possível estender a ontologia *IoT-Priv*, possibilitando que esta se adapte a necessidades e domínios específicos, tal como foi realizado no cenário de assistência médica domiciliar. Neste sentido, demonstramos que a ontologia *IoT-Priv* abrange as características de expressividade e extensibilidade que defendemos.

No entanto, com relação a verificação da ontologia, devido à escassez de conjuntos de dados disponíveis publicamente, nos esforçamos para popular e construir apenas dois cenários utilizando a ontologia *IoT-Priv*, isso porque sem um conjuntos de dados apropriados é necessário um grande esforço manual para obter resultados mais confiáveis em relação à exatidão e integralidade do modelo produzido.

Além disso, também percebemos que a ontologia *IoT-Priv*, por si só não é capaz de responder a algumas questões de competência, tais como: i) quais as requisições de consumo que foram produzidas dentro de uma determinada localização geográfica, utilizando-se os requisitos #13 e #27; e ii) quais as requisições de consumo cujo consentimento ainda é válido temporalmente, isto é, que ainda não expirou, segundo os requisitos #9 e #12. Neste sentido, para que tais perguntas possam ser respondidas, surge a necessidade de que outros métodos sejam propostos, tais como processamento, inferência ou filtragem de contexto, neste sentido, apresentamos um serviço de gerenciamento de privacidade, nomeado *IoT-PrivServ*, que entre outras funcionalidade, consegue responder estas questões de competência através de filtros espaciais e temporais, as quais não são respondidas unicamente com a ontologia *IoT-Priv*, e é apresentado no Capítulo 4.

Finalmente, uma avaliação foi conduzida, onde medimos o Tamanho e a Expressividade das ontologias *IoT-Priv* e IoT-Lite através de suas métricas. Ao considerar os resultados, foi constatado que a ontologia *IoT-Priv*, apesar do notável aumento de suas métricas em relação à IoT-Lite, houve apenas uma ligeira variação em sua expressividade lógica, e consequentemente, a complexidade computacional se manteve inalterada, concluindo que a *IoT-Priv* é tão leve quanto a IoT-Lite.

CAPÍTULO 4

Serviço de Privacidade *IoT-Priv*Serv

O serviço de gerenciamento de privacidade *IoT-PrivServ* foi projetado com o objetivo de mediar as interações entre os produtores e consumidores de dados, que utilizem a ontologia *IoT-Priv*. Dentre suas funcionalidades, este serviço permite:

1. a validação do modelo de entrada, segundo as definições da ontologia *IoT-Priv*;
2. a derivação de novas informações a partir daquelas já existentes;
3. a aplicação de filtros espaciais e temporais nos resultados das consultas, em complemento aos requisitos de privacidade.

Neste Capítulo, serão apresentados a arquitetura do Serviço *IoT-PrivServ*, bem como os detalhes de implementação em relação a essas principais funcionalidades, além de uma avaliação do serviço proposto.

4.1 Arquitetura

O *IoT-PrivServ* é um serviço projetado para realizar o armazenamento, inferências e consultas aos modelos semânticos construídos com a ontologia *IoT-Priv*, abstraindo das aplicações e/ou usuários a complexidade de realizar tais tarefas.

A arquitetura do serviço *IoT-PrivServ* é organizada em camadas, sendo apoiado pela ontologia *IoT-Priv* e por uma máquina de inferência (Figura 4.1). Internamente ao serviço, tem-se a seguinte organização:

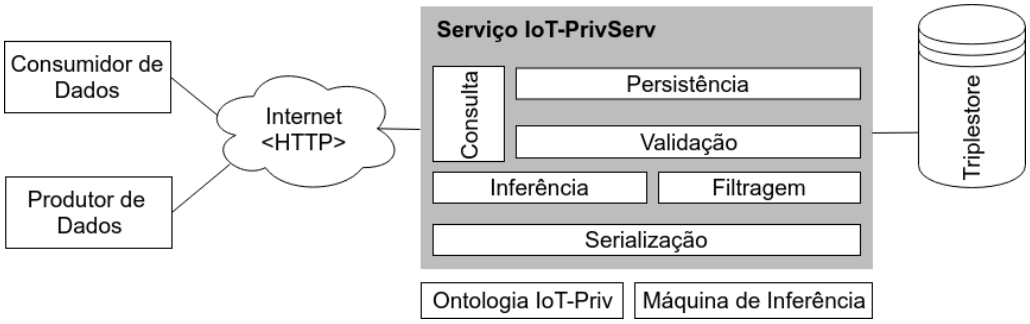

Figura 4.1: *Arquitetura do Serviço de IoT-PrivServ.*

- **Serialização**: serviço da primeira camada, responsável por converter modelos semânticos serializados em formato *Turtle* para um Objeto java, bem como o procedimento inverso.
- **Inferência**: serviço da segunda camada, responsável por inferir novas informações a partis dos fatos instanciados em um determinado modelo semântico, na forma de um Objeto java.
- **Filtragem**: serviço também pertencente a segunda camada, responsável por disponibilizar filtros semânticos, os quais podem ser aplicados em conjuntos com as consultas SPARQL.
- **Validação**: serviço da terceira camada, responsável por validar um modelo semântico, ou seja, verificar se um determinado modelo foi construído corretamente, segundo as definições da *IoT-Priv*, utilizando a máquina de inferência.
- **Persistência**: serviço da quarta camada, responsável por armazenar um modelo semântico na base de dados, desde que tenha sido devidamente validado. Este serviço também é responsável por armazenar as novas informações produzidas pelo serviço de inferência.
- **Consulta**: serviço que tange a quarta e quinta camada, responsável por receber consultas SPARQL e processá-las na base de dados, retornando os resultados obtidos.

A comunicação com o serviço *IoT-PrivServ* acontece da seguinte forma: os produtores (p.ex. um sistema ou aplicativo IoT) e os consumidores (p.ex. um *App*) se comunicam com o *IoT-PrivServ* através da Internet, por meio de requisições HTTP. Consequentemente, o IoT-PrivServ processa a requisição, e se for necessário, se comunica com a base de dados de triplas para armazenar ou consultar informações. Vale ressaltar que a base de dados de triplas foi utilizada, pois ao contrário de um banco de dados relacional ou NoSQL, uma *triplestore* é otimizada para o armazenamento e recuperação de triplas em formato RDF.

Sendo assim, o *IoT-PrivServ* é classificado como um *web service*, e foi assim projetado com o intuito de simplificar a integração do mesmo com os componentes já existentes das infraestruturas de softwares que forem utilizá-lo.

4.2 Implementação do Serviço

Desenvolvido em Java 8.0, utiliza os *frameworks Apache Jena* e *Spring Boot*, que possibilitam criar aplicativos de Web Semântica e *web services* RESTFull, respectivamente. A *triplestore* adotada é a TDB com suporte ao ARQ, ambos integrados ao *Apache Jena* para armazenamento e consulta de dados RDF. A seguir será detalhada a API RESTful, a qual expõe as funcionalidades do serviço *IoT-PrivServ*.

4.2.1 HTTP GET

Representado no Algoritmo 4.1, este método é utilizado apenas para obter informações e recursos do serviço. Ele possui dois parâmetros: *request = status* e *request = iot-priv*, com a seguinte especificação:

Algoritmo 4.1: MÉTODO HTTP GET

```
   Entrada: Parâmetro da requisição HTTP GET
   Saída: Resultado da requisição
   Variaveis: STRING: requisicao
1  início
2      se requisicao = "iot-priv" então
3          retorna getOntologiaIotPriv().serializar() //Em formato Turtle
4      senão se requisicao = "status" então
5          retorna getServerStatus() //Em formato JSON
6      retorna "Erro"
```

- **request = iot-priv**: retorna a ontologia *IoT-Priv*, serializada em *Turtle*.
- **request = status**: retorna o status atual do serviço, bem como suas estatísticas em formato *JSON*, como exemplificado no Código 4.1. O formato JSON foi escolhido pois é leve e fácil de processar, uma vez que praticamente toda linguagem de programação moderna tem suporte a JSON.
  Disponibilizamos estas informações, pois acreditamos que elas podem ser úteis para administradores de sistemas, uma vez que permitem identificar uma série de parâmetros, os quais podem ser utilizados para auditoria, bem como identificar gargalos no desempenho do sistema, tais como:
  - O status atual do serviço, apresentando o *timestamp* em que o serviço foi inicializado (linha 2).
  - A quantidade de requisições HTTP GET, HTTP POST e HTTP PUT respondidas (linhas 4, 8 e 12, respectivamente);
  - A quantidade de vezes que a ontologia *IoT-Priv* foi requisitada (linha 5);
  - A quantidade de modelos inválidos (linha 9);
  - A quantidade de consultas que, para serem respondidas, exigiram uma inferência prévia (linha 13);
  - A quantidade de consultas não permitidas, por exemplo, aquelas do tipo *DESCRIBE* ou *ASK*(linha 14)

Código 4.1 Resposta ao método HTTP GET com o parâmetro request=status

```
{
  "server status":"running since  11/08/2018 13:27:01",
  "get requests": [{
    "qtd":15,
    "qtdGetIotPriv":3
  }],
  "post requests": [{
    "qtd":9
    "qtdPostInvalid":2
  }],
  "put requests": [{
    "qtd":38,
    "qtdPutReasons":16
    "qtdPutNotPermited":2,
  }]
}
```

4.2.2 HTTP POST

Representado no Algoritmo 4.2, este método é utilizado para adicionar um modelo completo à base de dados, contendo informações do sistema e de suas respectivas políticas de privacidade.

Algoritmo 4.2: MÉTODO HTTP POST

Entrada: Modelo serializado em formato TURTLE.
Saída: Resultado da requisição
Variaveis: STRING: modeloSerializado; MODELO: modelo BOOLEANO: eValido

```
1 início
2     modelo ← constroiModelo(modeloSerializado) // parser
3     eValido ← validaModelo(modelo)
4     se eValido = VERDADE então
5         atualizaTripleStore(modelo)
6         retorna "Sucesso"
7     senão
8         retorna "Falha"
```

O método não necessita de parâmetros, no entanto, deve ser enviado no corpo da requisição HTTP os dados a serem armazenados, os quais devem estar modelados segundo a ontologia *IoT-Priv* e serializados em formato *Turtle*. Estes dados são validados

pela máquina de inferência *Pellet*, através da *Inference* API do *framework Apache Jena*, e consequentemente, adicionados à base de dados. Vale ressaltar que, caso o modelo não esteja construído corretamente, ou seja, de acordo com as definições e restrições especificadas na *IoT-Priv*, o mesmo não será armazenado na base de dados. Por fim, o método HTTP POST retorna códigos de sucesso (ou de erro) para quem o invocou, como mostra os Códigos 4.2 e 4.3, respectivamente.

Código 4.2 Resposta ao método HTTP POST com modelo válido

```
1 {
2   "model is valid" : true ,
3   "model was stored " : true ,
4   "request status " : "ok"
5 }
```

Código 4.3 Resposta ao método HTTP POST com modelo inválido

```
1 {
2   "model is valid" : false ,
3   "model was stored " : false ,
4   "request status " : "denied"
5 }
```

Dentre os tipos de erros detectados pela validação de modelos, incluem:

a) Nomes de propriedades ou de classes, escritos de forma incorreta.
P.ex.: *(Pessoa1, iot-priv:o**nw**s, Sistema1)* ou *(Pessoa1, rdf:type, DataO**nw**er)*

b) O não cumprimento de restrições de domínio ou imagem.
P.ex.: ***(Pessoa1**, iot-priv:owns, **Pessoa2**)*, onde Pessoa1 e Pessoa2 são *iot-priv:DataOwners*, sendo que, por definição da ontologia *IoT-Priv*, a propriedade *iot-priv:owns* tem domínio em *iot-priv:DataOwner* e imagem em (*iot-priv:System* OU *iot-priv:Device*), logo, Pessoa2 deveria ser do tipo *ssn:System* ou *ssn:Device*.

c) O não cumprimento de restrições de cardinalidade.
P.ex.: ***(Sistema1**, **iot-priv:isOwnedBy**, Pessoa1)* e ***(Sistema1**, **iot-priv:isOwnedBy**, Pessoa2)*, onde *Sistema1* é um *ssn:System*, e *Pessoa1* e *Pessoa2* são *iot-priv:DataOwners*, e por definição da ontologia *IoT-Priv*, um *ssn:System* é propriedade de um único *iot-priv:DataOwner*. Neste caso, o mesmo Sistema (Sistema1) está sendo propriedade de dois *iot-priv:DataOwners* (Pessoa1 e Pessoa2), sendo assim, há duas possíveis interpretações: i) o modelo está errado; ou ii) Pessoa1 e Pessoa2 são a mesma pessoa, neste caso, para que o modelo possa ser validado, também deve conter a seguinte tripla: *(Pessoa1, owl:sameAs, Pessoa2)*.

4.2.3 HTTP PUT

Representado no Algoritmo 4.3, este método é utilizado para executar uma consulta SPARQL na base de dados. Uma chamada a este método deve enviar no corpo da requisição HTTP uma consulta SPARQL, a qual será executada na base de dados, se for possível.

Algoritmo 4.3: MÉTODO HTTP PUT

```
Entrada: String de busca e Formato de exibição dos resultados
Saída: Resultado da requisição
Variaveis: STRING: stringBusca, STRING: formato, DATASET: dataset, MODEL: modelo
1  início
2      se !stringBusca.eTipoSelect() então
3          retorna "Erro! Consulta não permitida!"
4      senão
5          List<String> palavrasGatilho ← palavrasGatilho()
6          para cada palavra em palavrasGatilho faça
7              se stringBusca.contem(palavrasGatilho) então
8                  List<String> consultasConstruct = consultasConstruct();
9                  para cada construct em consultasConstruct faça
10                     modelo ← constructModel(construct)
11                     dataset.atualizaTripleStore(modelo)
12                 pare
13     retorna dataset.consultaTripleStore(stringBusca, formato)
```

Uma vez que a consulta SPARQL foi recebida pelo serviço, é realizada uma verificação, e caso ela não seja uma consulta do permitida, isto é, do tipo *SELECT*, o método deve retornar um erro. Essa decisão de projeto foi tomada porque as consultas *DESCRIBE* e *CONSTRUCT* retornam não apenas informações, mas partes do grafo armazenado, o que é potencialmente perigoso a nível de privacidade, e portanto foram bloqueadas a nível de usuário, o que está de acordo com a primeira diretriz do *framework* PbD. Consequentemente, as consultas *ASK* também foram bloqueadas, pois apesar de retornarem apenas Sim ou Não, é possível revelar brechas na privacidade dos usuários, dependendo de como e com que frequência estas consultas forem realizadas.

Uma vez que o método determina que a consulta é do tipo *SELECT*, o próximo passo é verificar se a consulta em questão faz referência a algum termo que, para ser utilizado, precise de algum tipo de inferência prévia. Estes termos foram denominados *palavras-gatilhos*, sendo elas: *providesConsentTo*, *consumes*, *isConsumedBy*, *isProtectedBy*, *isOnwedBy*, *isSubSystemOf* e *exposes*.

As palavras-gatilho *providesConsentTo* e *consumes* fazem entender que a consulta está interessada em responder se um determinado usuário tem o consentimento e/ou

pode consumir um determinado serviço, para isso, a prévia inferência do consentimento é necessária. As demais palavras-gatilho são propriedades inversas, as quais também precisam ser preenchidas, se não existirem.

Caso seja necessário, a inferência é realizada aplicando uma série de consultas do tipo *CONSTRUCT*, as quais são listadas nos Códigos 4.4 a 4.10. Estas consultas retornam subgrafos com as novas informações construídas, os quais são armazenados na base de dados, complementando as informações já existentes.

O Código 4.4 constrói a propriedade *iot-priv:providesConsentTo*, verificando se uma requisição de consumo está de acordo com a política de privacidade do serviço que deseja consumir e se o consumidor que produziu esta requisição não está inserido na lista de negação de serviço (*iot-priv:DenyList*). De forma similar, o Código 4.5 também constrói a propriedade *iot-priv:providesConsentTo*, no entanto, a verificação é feita com base apenas na lista de concessão de acesso (*iot-priv:GrantList*), verificando se o consumidor que gerou determinada requisição está incluído nesta lista.

Código 4.4 Construção da propriedade *iot-priv:providesConsentTo* a partir das definições expressas na política de privacidade.

```
1  PREFIX iot-priv: <http://example.org/ontology/iot-priv#>
2  PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
3  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4  CONSTRUCT {?dataOnwer iot-priv:providesConsentTo ?consumeRequest}
5  WHERE{
6    ?dataOnwer iot-priv:owns ?system .
7    ?system iot-lite:exposedBy ?service .
8    ?recipient iot-priv:hasConsumptionRequest ?consumeRequest .
9    ?consumeRequest iot-priv:consumes ?service .
10   ?policy iot-priv:protects ?service .
11   ?policy iot-priv:hasGeneralPrivacyPolicy ?generalPolicy .
12   ?consumeRequest iot-priv:requestPurpose ?consumePurpose .
13   ?consumeRequest iot-priv:hasOptIn ?policy .
14   FILTER EXISTS {
15     ?generalPolicy iot-priv:accessPurposeAllowed ?consumePurpose .
16   }
17   OPTIONAL{
18     ?service iot-priv:hasAccessList ?denyAccessList .
19     ?denyAccessLIst rdf:type iot-priv:DenyList .
20   }
21    FILTER NOT EXISTS {
22     ?denyAccessList iot-priv:hasRecipient ?recipient .
23   }
24 }
```

Código 4.5 Construção da propriedade *iot-priv:providesConsentTo* a partir das definições expressas na lista de acesso do serviço.

```
1  PREFIX iot-priv: <http://example.org/ontology/iot-priv#>
2  PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
3  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4  CONSTRUCT { ?dataOnwer iot-priv:providesConsentTo ?consumeRequest. }
5  WHERE{
6    ?dataOnwer iot-priv:owns ?system .
7    ?system iot-lite:exposedBy ?service .
8    ?recipient iot-priv:hasConsumptionRequest ?consumeRequest .
9    ?consumeRequest iot-priv:consumes ?service .
10   ?service iot-priv:hasAccessList ?grantAccessList .
11   ?grantAccessList rdf:type iot-priv:GrantList .
12   FILTER EXISTS {
13     ?grantAccessList iot-priv:hasRecipient ?recipient .
14   }
15 }
```

A seguir, o Código 4.6 constrói a propriedade inversa *iot-lite:exposes*, isto é, se existe uma relação (*x ,iot-lite:exposedBy, y*), então gera-se (*y, iot-lite:exposes, x*). De forma análoga, os Códigos 4.7, 4.8, 4.9 e 4.10 também constroem as propriedades inversas *ssn:isSubSystemOf*, *iot-priv:isOwnedBy*, *iot-priv:isConsumedBy* e *iot-priv:isProtectedBy*, respectivamente.

Código 4.6 Construção da propriedade *iot-lite:exposes*.

```
1  PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
2  CONSTRUCT { ?service iot-lite:exposes ?system }
3  WHERE{  ?system iot-lite:exposedBy ?service . }
```

Código 4.7 Construção da propriedade *ssn:isSubSystemOf*.

```
1  PREFIX ssn: <http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#>
2  CONSTRUCT { ?system2 ssn:isSubSystemOf ?system }
3  WHERE{  ?system ssn:hasSubSystem ?system2 . }
```

Código 4.8 Construção da propriedade *iot-priv:isOwnedBy*.

```
1  PREFIX iot-priv: <http://example.org/ontology/iot-priv#>
2  CONSTRUCT { ?system iot-priv:isOwnedBy ?dataOnwer }
3  WHERE{  ?dataOnwer iot-priv:owns ?system . }
```

Código 4.9 Construção da propriedade *iot-priv:isConsumedBy*.

```
1 PREFIX iot-priv: <http://example.org/ontology/iot-priv#>
2 CONSTRUCT { ?service iot-priv:isConsumedBy ?consumeRequest }
3 WHERE{  ?consumeRequest iot-priv:consumes ?service . }
```

Código 4.10 Construção da propriedade *iot-priv:isProtectedBy*.

```
1 PREFIX iot-priv: <http://example.org/ontology/iot-priv#>
2 CONSTRUCT { ?service iot-priv:isProtectedBy ?policy }
3 WHERE{  ?policy iot-priv:protects ?service . }
```

Esta abordagem foi utilizada por motivos de desempenho, visto que realizar uma chamada a uma máquina de inferência antes de realizar cada consulta seria muito caro computacionalmente. Já as consultas *CONSTRUCT* conseguem produzir as novas informações que o serviço necessita para responder as consultas, e com um tempo de execução muito inferior em relação à uma máquina de inferência.

Por fim, o método *HTTP PUT* ainda aceita dois possíveis parâmetros, os quais são responsáveis por determinar como será formatada a saída de dados: *output = JSON* e *output = TEXT*. Se nenhum parâmetro for enviado junto à chamada do método, por padrão, a chamada de *output = JSON* será executada.

- **output=TEXT**: retorna o resultado da consulta, serializado em formato de tabelas. Essa versão é preferível quando o resultado da requisição for interpretado por humanos, pois a leitura é mais natural. A Figura 4.2 ilustra o resultado de uma consulta onde é retornado uma lista de sistemas e seus respectivos proprietários.

```
-------------------------------------------------------------------------------------------------------------------------------------------
| sistema                                                                                | proprietario                                                   |
===========================================================================================================================================
| <http://example.org/instances/iot-priv#SmartHospitalManagement> | <http://example.org/instances/iot-priv#SmartHospital> |
| <http://example.org/instances/iot-priv#AndrewSmartHouse>        | <http://example.org/instances/iot-priv#UserAndrew>    |
-------------------------------------------------------------------------------------------------------------------------------------------
```

Figura 4.2: *Resultado de uma requisição HTTP PUT com parâmetro output = TEXT.*

- **output=JSON**: essa versão é preferível quando o resultado da requisição for interpretado por máquinas, pois retorna o resultado da consulta serializado JSON, que é um formato amplamente utilizado para intercâmbio de dados, e além disso, é mais simples de ser processado por *software*. O Código 4.11 ilustra o resultado de uma consulta idêntico ao da Figura 4.2, porém, serializado em formato JSON.
  No resultado ilustrado no Código 4.11, tem-se as seguintes informações: as linhas 3 define as variáveis que serão retornadas. A seguir, os resultados da consulta são

apresentados, o primeiro resultado é apresentado nas linhas 8 e 9, enquanto o segundo é apresentado nas linhas 12 e 13.

Código 4.11 Resultado da requisição HTTP PUT com parâmetro *output = JSON*.

```
1  {
2    "head": {
3      "vars": [ "sistema" , "proprietario" ]
4    } ,
5    "results": {
6      "bindings": [
7        {
8          "sistema": { "type": "uri" , "value": "http://example.org/
               instances/iot-priv#SmartHospitalManagement" } ,
9          "proprietario": { "type": "uri" , "value": "http://example.
               org/instances/iot-priv#SmartHospital" }
10       } ,
11       {
12         "sistema": { "type": "uri" , "value": "http://example.org/
               instances/iot-priv#AndrewSmartHouse" } ,
13         "proprietario": { "type": "uri" , "value": "http://example.
               org/instances/iot-priv#UserAndrew" }
14       }
15     ]
16   }
17 }
```

A seguir, a Tabela 4.1 sumariza a API do serviço *IoT-PrivServ*, supracitada.

Tabela 4.1: *Sumarização da API do Serviço de Gerenciamento de Privacidade.*

Método HTTP	Corpo	Parâmetro	Resposta	Descrição
GET	–	request=status	application/json	Retorna informações sobre o status e estatísticas do serviço.
		request=iot-priv	text/plain	Retorna a ontologia *IoT-Priv*, serializada em formato Turtle.
POST	Modelo *IoT-Priv* no formato *Turtle*	–	application/json	Envia para publicação na base de dados um modelo de privacidade, anotado com a ontologia *IoT-Priv*.
PUT	Consulta SPARQL	output=text	text/plain	Retorna resultado de consulta SPARQL em forma tabular legível por humanos.
		output=json	application/json	Retorna resultado da consulta SPARQL na sintaxe JSON legível por máquinas.

4.2.4 Filtros Temporais e Espaciais

Em complemento ao método HTTP PUT, foram implementados dois tipos de filtros utilizando a API *ARQ* do *framework Apache Jena*, os quais podem ser utilizados em conjunto com consultas SPARQL. Estes fornecem o recurso de filtragem de consultas envolvendo privacidade com base em características espaciais e temporais, filtragem esta realizada através do processamento dos conceitos e propriedades definidos nas políticas de privacidade, via ontologia *IoT-Priv*.

A principal motivação para o desenvolvimento destes filtros foi possibilitar que consultas mais expressivas fossem escritas e respondidas, p.ex.:

1. retornar a lista de consumidores que registraram requisições de consumo a algum serviço dentro de uma determinada área geográfica;
2. retornar a lista de consumidores que ainda tem acesso a um determinado serviço, isso é, que ganharam o consentimento, mas que ainda não expirou.

Estes filtros são disponibilizados através do serviço *IoT-PrivServ*, sob o espaço de nomes *iot-priv*, de modo que qualquer consumidor possa utilizá-los em suas consultas SPARQL. Vale ressaltar que, como os filtros são uma espécie de extensão às consultas SPARQL, também são processados como uma consulta, ou seja, através das requisições HTTP PUT. A seguir, estão as descrições dos filtros supracitados:

1. ***iot-priv:PointWithinPolygon***: corresponde ao filtro espacial para políticas de privacidade. Recebe dois parâmetros, a saber: i) um ponto geográfico, representando a localização na qual um requisição de consumo foi gerada; e ii) um polígono representando a área na qual o acesso à informação é permitido, conforme definido na política de privacidade. Ambos parâmetros são serializados no formato WKT. O Algoritmo 4.4 demonstra a implementação deste filtro, o qual é detalhado a seguir.

Algoritmo 4.4: FILTRO ESPACIAL *iot-priv:PointWithinPolygon*

```
   Entrada: Polígono e Ponto, serializados em WKT
   Saída: Valor booleano, informando se o ponto está contido no polígono.
   Variaveis: STRING: poligono, STRING: ponto
1  início
2      se poligono.éUmPoligonoWKTValido() E ponto.éUmPontoWKTValido() então
3          retorna pontoEstaDentroDoPoligono(ponto, poligono)
```

Inicialmente, tem-se como entrada duas *Strings*, representando um polígono e um ponto. A seguir, o algorítimo verifica se estas *Strings* estão construídas de acordo com a serialização WKT (linha 2), para isto, utiliza-se expressões regulares. Uma vez que as *Strings* foram validadas, o método *pontoEstaDentroDoPoligono* é

invocado (linha 3), retornando *verdade*, caso o ponto esteja contido no polígono, e *falso*, caso contrário. Vale ressaltar que o método *pontoEstaDentroDoPoligono* foi implementado utilizando o algoritmo *Point-In-Polygon*[1]

Um exemplo de uso é o seguinte: suponha que existam 3 pontos geográficos armazenados na base de dados, sendo eles (3, 2), (1, 2), (6, 2). Deseja-se descobrir qual ponto geográfico esta contido em um polígono, determinado pelas coordenadas ((1, 0), (3, 5), (5, 0)). Neste sentido, executa-se a consulta do Código 4.12. Este filtro deve retornar apenas o ponto (3, 2), pois é o único que está contido no polígono descrito, me ilustra a Figura 4.3.

Código 4.12 Exemplo de filtro espacial.

```
1 SELECT ?ponto
2 WHERE{
3     ?x  rdf:type     geo:point;
4         iot-priv:hasWKTPoint     ?p;
5 }
6 FILTER (iot-priv:pointWithinPolygon(?p, "POLYGON((1 0, 3 5, 5 0))"))
```

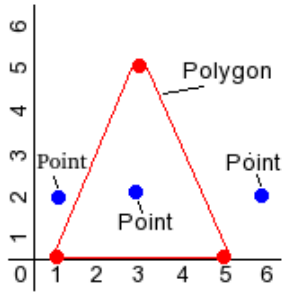

Figura 4.3: *Exemplificação do filtro espacial.*

2. ***iot-priv:ConsentExpired***: corresponde ao filtro temporal para políticas de privacidade. Recebe três parâmetros, sendo eles: i) um valor literal, representando o instante de tempo em que a solicitação de consumo foi gerada, no formato *xsd:dateTimeStamp*; ii) um número inteiro positivo, representando o tempo máximo de retenção de dados composto por um valor unitário ; e iii) um valor literal para a duração do tempo, o qual pode assumir os seguintes valores [*milliseconds, seconds, minutes, hours, days, weeks, months, years*]. O Algoritmo 4.5 demonstra a implementação deste filtro, o qual é detalhado a seguir.

[1] disponível para consulta em http://alienryderflex.com/polygon/

Algoritmo 4.5: FILTRO TEMPORAL *iot-priv:ConsentExpired*

Entrada: Instante inicial, duração máxima e a descrição associada a esta duração.
Saída: Valor *booleano*, informando se o consentimento está expirada ou não.
Variaveis: STRING: instante, INTEIRO POSITIVO: duração, STRING: descriçãoDuração

```
1 início
2     se instante.éDateTime() E descriçãoDuracação.éDescriçãoVálida() então
3         duraçãoMáxima ← calculaMáximaDuração(instante, duração, descriçãoDuração);
4         agora ← obtemTimestampAtual();
5         retorna maior(duraçãoMáxima, agora)
```

Inicialmente, tem-se como entrada um número inteiro positivo, representando a duração máxima e duas Strings, representando o instante em que a solicitação de consumo foi gerada, bem como a descrição associada a esta duração. A seguir, o algorítimo verifica se o conteúdo das *Strings* está de acordo com o permitido (linha 2), para isto, utiliza-se expressões regulares. A seguir, na linha 3 é realizada o procedimento para obter a duração máxima permitida, determinada pelo instante em que a requisição foi gerada somado ao tempo máximo de retenção de dados. Por sua vez, a linha 4 obtêm o *timestamp* atual do sistema, isto é, o instante em que o filtro foi processado no serviço. Por fim, o retorno do algoritmo é fornecido na linha 5, retornando *verdade* caso o consentimento já tenha expirado, isto é, caso o instante atual seja maior que o duração máxima permitida.

Um exemplo de uso é o seguinte: suponha que existam diversas requisições de consumo a serviços armazenados na base de dados, e deseja-se obter a lista de requisições de consumo que foram geradas há mais de 5 dias, isto é, que já expiraram temporalmente e podem ser excluídas da base de dados. Para isso, aplica-se a consulta demonstrada no Código 4.13, o qual é ilustrado na Figura 4.4.

Código 4.13 Exemplo de filtro temporal.

```
1 SELECT ?requisicao
2 WHERE{
3     ?requisicao  rdf:type     iot-priv:ConsumptionRequest;
4                  iot-priv:requestInstant     ?instante.
5 }
6 FILTER (iot-priv:consentExpired(?instante, 5, "days"))
```

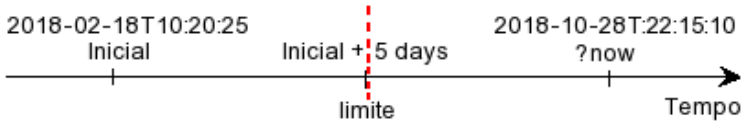

Figura 4.4: *Exemplificação do filtro temporal.*

4.2.5 Esquema de Comunicação Interna

Por fim, esta Subseção ilustra o esquema de comunicação entre os componentes internos ao serviço *IoT-PrivServ*. Para isso, exemplificamos utilizando dois cenários: i) o produtor registrando suas informações no *IoT-PrivServ*; e ii) o consumidor enviando uma consulta qualquer para ser processada no *IoT-PrivServ*.

Visão do Produtor de Informações

Este cenário ilustra a comunicação entre um produtor de informações com o serviço *IoT-PrivServ*, como é ilustrado na Figura 4.5 e detalhado a seguir:

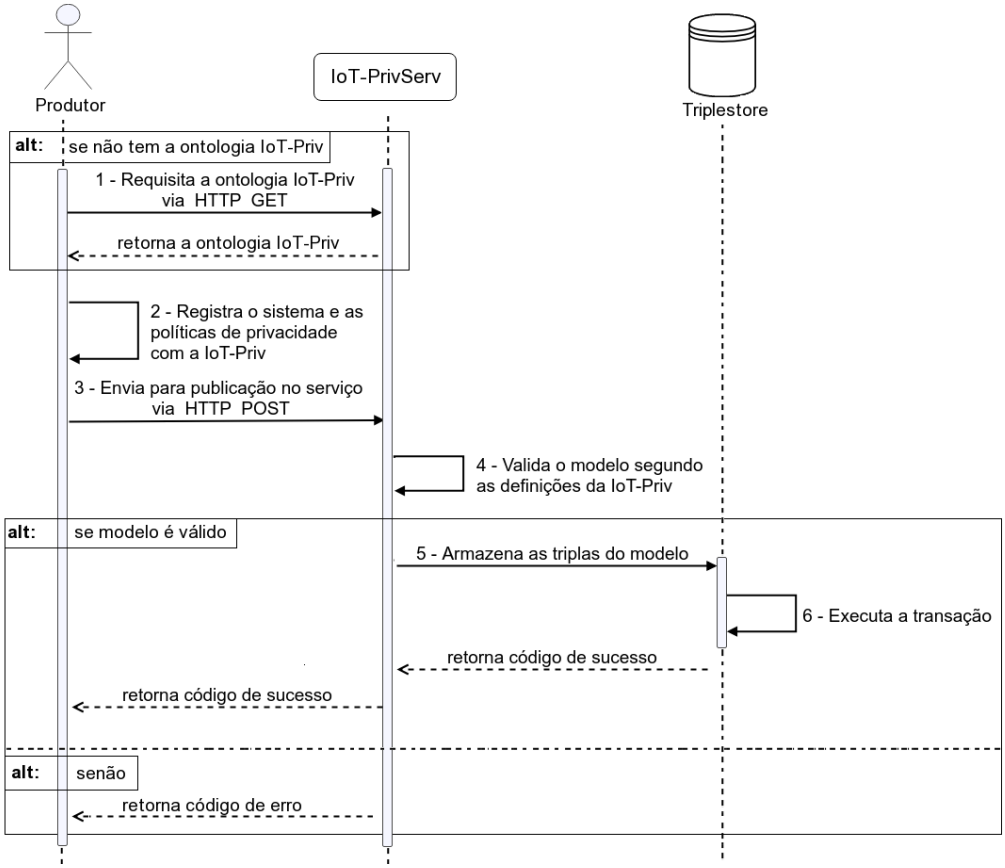

Figura 4.5: *Esquema de comunicação do IoT-PrivServ – visão do produtor de informações.*

1. Inicialmente, o produtor pode requisitar a ontologia *IoT-Priv*, caso ainda não a tenha. Para isso, envia uma requisição do tipo *HTTP GET* com parâmetro *request = iot-priv* para o serviço, o qual retorna a ontologia, serializada em formato *Turtle*.
2. A seguir, o produtor registra o sistema, bem como suas respectivas políticas de privacidade, utilizando a ontologia *IoT-Priv*.
3. O modelo produzido na etapa anterior é enviado para ao serviço *IoT-PrivServ*, através do método *HTTP POST*.

4. O modelo recebido na etapa anterior é validado. Para isso, utiliza-se as definições da ontologia *IoT-Priv* e a máquina de inferência *Pellet*. O resultado desta validação é um valor *booleano*, indicando se o modelo está construído corretamente, de acordo com as definições e restrições da ontologia *IoT-Priv*.
    - Se o modelo é válido: é armazenado na *triplestore*, retornando um código de sucesso para o *IoT-PrivServ*, e consequentemente, para o produtor.
    - Se não: retorna um código de erro diretamente para o produtor.

Visão do Consumidor de Informações

Este cenário ilustra a comunicação entre um consumidor de informações com o serviço *IoT-PrivServ*, como é ilustrado na Figura 4.6 e detalhado a seguir:

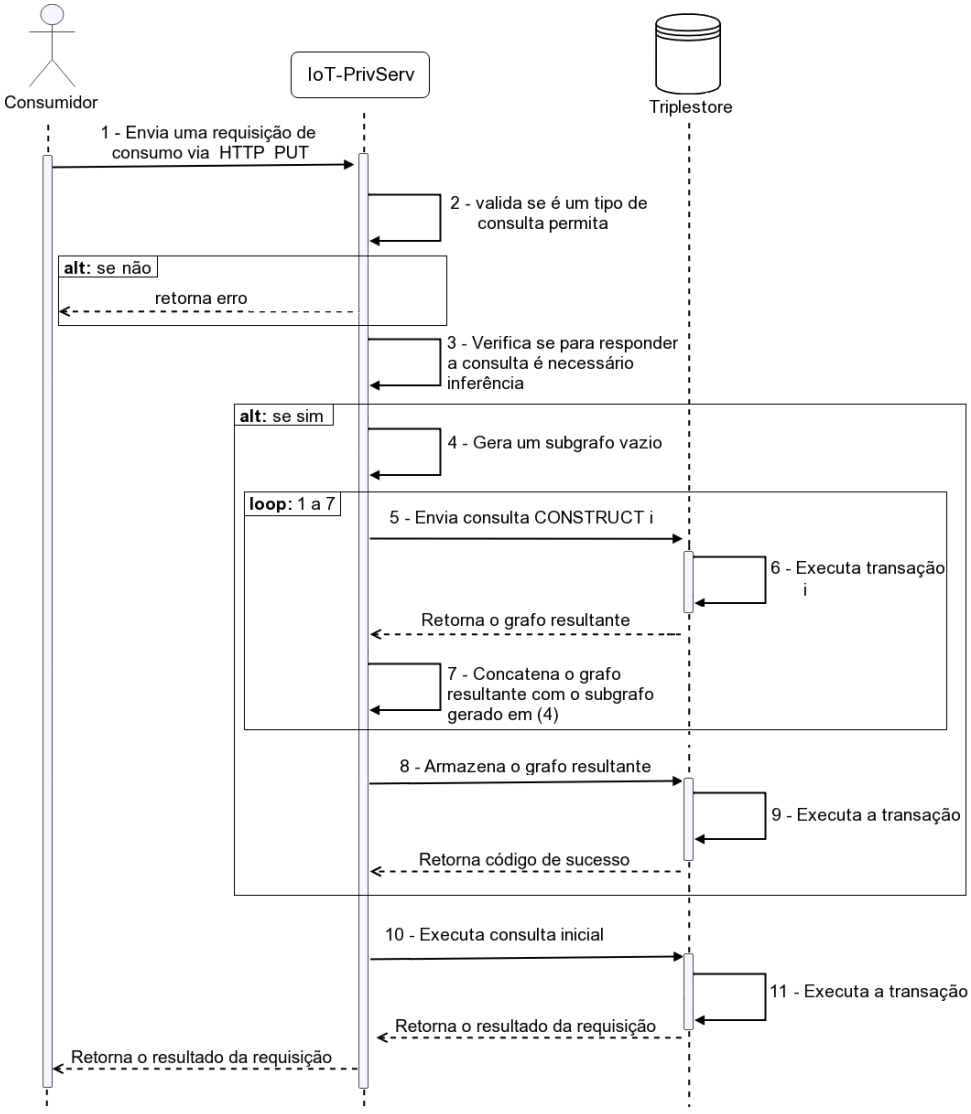

Figura 4.6: *Esquema de comunicação do IoT-PrivServ – visão do consumidor de informações.*

1. O consumidor envia uma consulta SPARQL para ser processada no *IoT-PrivServ*, através do método *HTTP PUT*. Vale ressaltar que esta consulta pode ou não conter os filtros temporais e espaciais.
2. O *IoT-PrivServ*, internamente, realiza uma validação da consulta enviada, determinando se é um tipo permitida.
3. A seguir, o *IoT-PrivServ* verifica se, para responder esta consulta, será necessário *inferência prévia.* Caso seja:

   4 a 7. O *IoT-PrivServ* constrói um subgrafo vazio, e executa as 7 consultas CONSTRUCT na triplestore, concatenando os resultados com o subgrafo inicialmente gerado. O resultado desta etapa é um subgrafo com as novas informações inferidas.

   8 e 9. A seguir, o *IoT-PrivServ* armazena o subgrafo com as novas informações inferidas na *triplestore*, que por sua vez retorna um código de sucesso para o serviço, confirmando que foi devidamente armazenada.

10. Por fim, a consulta recebida no passo (1), é executada na *triplestore*. A seguir, os resultados da consulta são repassados para o *IoT-PrivServ*, e consequentemente, para o consumidor.

4.3 Avaliação do Serviço *IoT-PrivServ*

Nesta Seção é realizada uma avaliação do serviço, no sentido de medir o tempo de resposta de cada funcionalidade do *IoT-PrivServ*. Dentre as funcionalidades avaliadas, incluem:

- adicionar modelos com quantidades de triplas distintas;
- adicionar um modelo válido e também um inválido;
- responder consultas que necessitam de inferência e consultas que não necessitam; e,
- determinar que uma consulta não é válida.

Vale ressaltar que as configurações supracitadas permitem avaliar todas as características e funcionalidades do *IoT-PrivServ*, as quais serão avaliadas através dos modelos de compartilhamento de fotos e assistência médica domiciliar, apresentados na Seção 3.3.

4.3.1 Objetivo e Configuração do Experimento

O objetivo deste experimento é avaliar o custo associado a cada funcionalidade do serviço *IoT-PrivServ*, uma vez que estas são altamente dependentes da ontologia *IoT-Priv*.

Neste sentido, este experimento fornece uma base de comparação de desempenho para os próximos sistemas baseados na ontologia *IoT-Priv*.

- **Métricas**: em todas as funcionalidades será medido o *tempo de resposta*. Para fins de padronização, o tempo será sempre expresso em *Milissegundos (ms)*.
- **Procedimentos**: para efeitos de validade estatística, o tempo de resposta de cada configuração será medido 30 vezes, e então, o tempo final desta configuração é dado pela *média* destes 30 tempos parciais.
  Além disso, vale ressaltar que as configurações que envolvem consultas e/ou inferência, serão realizadas na base de dados já populada, isto é, com os modelos de compartilhamento de fotos e assistência médica domiciliar já armazenados.
- **Configuração de Hardware**: os experimentos foram realizados em um *notebook* ASUS, com processador Intel Core i5-5200U de 2.7GHz, 6 Gb de memória RAM e placa de vídeo *GeForce* 930M.
- **Configuração de Software**: os experimentos foram realizados sob o sistema operacional Ubuntu 16.04 com *kernel Linux 4.4.0-138-generic*. A versão Java 8 foi utilizada (*build 1.8.0_191-b12*), com o *Spring framework* em sua versão 5.1.2 e o *framework Apache Jena* em sua versão 3.8.0. Por fim, a aplicação Java foi exporta da para o formato *WAR* e implantada no servidor *Apache Tomcat*, versão 9.0.10.

4.3.2 Resultados e Discussão

A seguir são apresentados os resultados e as respectivas discussões relacionadas aos testes dos métodos HTTP GET, POST e PUT, respectivamente.

Método HTTP GET

Em virtude de suas funcionalidades, as quais não necessitam de processamento, as requisições do tipo HTTP GET apresentam tempo de resposta baixo e com pouca variação.

Em nossos experimentos, a requisição HTTP GET, quando executada com o parâmetro ***request = status***, gastou em média apenas **12ms**, e apesar de simples, fornece uma série de informações úteis para possíveis administradores de sistema, a um baixo custo.

Por sua vez, a requisição do tipo HTTP GET, quando executada com o parâmetro **request = iot-priv** consome, em média **42ms**, para ser respondida, sendo que o resultado desta requisição é a própria ontologia *IoT-Priv*, serializada em formato *turtle*. Vale ressaltar que este tempo reflete unicamente o custo para enviar a *IoT-Priv* através da rede.

Estes tempos nos permitem concluir que o método HTTP GET do *IoT-PrivServ* não representa um custo significante, visto que o status do servidor, em geral, é uma

informação de interesse apenas de administradores. Já a ontologia *IoT-Priv*, em geral, deverá ser obtida apenas uma vez por cada produtor de dados, sendo assim, o custo de aquisição acontecerá apenas uma vez. Sendo assim, o custo associado ao método HTTP GET é considerado baixo.

Método HTTP POST

Estas requisições foram avaliadas com dois modelos distintos, sendo eles: i) o que representa o cenário do aplicativo *MyPhotos* (Apêndice A.1); e ii) o que representa o cenário de assistência médica domiciliar (Apêndice A.2).

Para o cenário do aplicativo *MyPhotos*, o procedimento de validação e armazenamento instanciado a partir da *IoT-Priv* consumiu em média **2827ms**, enquanto para o cenário de assistência médica domiciliar, em média **3284ms**. Estes altos tempos podem ser justificados pelo *alto custo associado à validação dos modelos de entrada*, que compreende em inicializar a máquina de inferência e carregá-la com a ontologia *IoT-Priv*, além é claro, da realizar a validação, propriamente dita. Além disso, vale ressaltar que, mesmo com a grande diferença de triplas por modelo (**56** e **179**, respectivamente), o tempo gasto para realizar o precedimento não variou muito, **457ms** apenas, confirmando a hipótese de que a validação representa um alto custo no processo de armazenamento de modelos.

Vale ressaltar que, em geral, as modelagens completas, tais como as supracitadas, não serão validadas e armazenadas com grande frequência, apenas quando um sistema se registrar no *IoT-PrivServ*. As triplas que serão frequentemente armazenadas e validadas pelo método HTTP POST são aquelas que representam as requisições de consumo (Código 4.14), uma vez que, idealmente, deveriam ser registradas sempre que um serviço for consumido. No entanto, esse tipo de processamento consome menos tempo, uma vez que são menores e mais simples (7 triplas), consumindo em média, **498ms**.

Código 4.14 Triplas referentes a uma nova requisição de consumo.

```
1  @prefix :          <http://example.org/instances/iot-priv#> .
2  @prefix owl:       <http://www.w3.org/2002/07/owl#> .
3  @prefix rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4  @prefix xsd:       <http://www.w3.org/2001/XMLSchema#> .
5  @prefix iot-priv:  <http://example.org/ontology/iot-priv#> .
6  :NewConsumptionRequest
7        a         owl:NamedIndividual , iot-priv:ConsumptionRequest ;
8        iot-priv:consumes :AndrewMedicalRecordService ;
9        iot-priv:hasOptIn :AndrewPrivacyPolicy ;
10       iot-priv:requestInstant "2018-10-28T22:38:14"^^xsd:dateTime ;
11       iot-priv:requestLocation :StudentDDD_ConsumeLocation ;
12       iot-priv:requestPurpose :AndrewHouse .
```

Além disso, para mensurar o tempo gasto de detecção de um modelo inválido, inserimos erros propositais nos modelos descritos no Apêndice A, antes de enviá-los para o serviço. Entre os erros inseridos, estão os de escrita ou com informações que violariam as restrições de cardinalidade, domínio ou imagem, semelhante ao que foi exemplificado anteriormente (Subseção 4.2.2). Ao realizar os experimentos, constatamos que, independente de onde o erro esteja localizado, seja no início, meio, ou final do documento, o tempo para determinar que o modelo é inválido é próximo ao tempo para determiná-lo válido.

Isto se deve à API *ValidityReport* do *framework ApacheJena*, a qual utilizamos para percorrer o modelo à procura de inconsistências. Essa API percorre todo o modelo à procura de erros para listá-los. Sendo assim, com o modelo estando válido ou não, em média, o tempo para realizar a validação será o mesmo.

Método HTTP PUT

Para avaliar estas requisições, construímos consultas SPARQL, as quais foram enviadas ao serviço e tiveram os seus respectivos tempos de respostas medidos.

Inicialmente, *nosso experimento verificou o tempo gasto para determinar que uma consulta é inválida*. Os resultados mostraram que, independente da complexidade da consulta envolvida, o tempo gasto é fixo, isso deve-se ao fato de que, para determinar que uma consulta é inválida, basta apenas verificar se esta possui as palavras-chave *DESCRIBE, CONSTRUCT* ou *ASK*, o que é um procedimento simples.

Por exemplo, ao executar as consultas dos Códigos 4.15, 4.16 e 4.17, o tempo médio para realizar a verificação de cada, é em média, **13 ms**, retornando o seguinte resultado: *{ "erro" : "tipo de consulta não implementada"}*, mesmo sendo consultas completamente distintas, como é descrito a seguir.

A consulta do Código 4.15 deveria informar se existem serviços que manipulam informações pessoalmente identificáveis, ou seja, serviços que poderiam ser atacados, futuramente, e consequentemente, a consulta do Código 4.16 deveria retornar um grafo, contendo todas as informações a respeito deste serviço, da forma como estão armazenadas na base de dados. Por sua vez, a consulta do Código 4.17 deveria gerar um conjunto de triplas, fazendo com o que os proprietários forneçam consentimento, indiscriminadamente, a todas as requisições de consumo, e posteriormente, estas triplas poderiam ser armazenadas à base de dados via HTTP POST, gerando sérios problemas de privacidade.

Código 4.15 Consulta ASK.

```
1 PREFIX iot-priv: <http://example.org/ontology/iot-priv#>
2 ASK {?service iot-priv:isPersonallyIdentifiable "true"}
```

Código 4.16 Consulta DESCRIBE.

```
1 PREFIX iot-priv:  <http://example.org/ontology/iot-priv#>
2 DESCRIBE ?service
3 WHERE    { ?x iot-priv:isPersonallyIdentiable true }
```

Código 4.17 Consulta CONSTRUCT.

```
1 PREFIX iot-priv: <http://example.org/ontology/iot-priv#>
2 PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
3 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4 CONSTRUCT { ?dataOnwer iot-priv:providesConsentTo ?consumeRequest. }
5 WHERE{
6   ?dataOnwer iot-priv:owns ?system .
7   ?system iot-lite:exposedBy ?service .
8   ?recipient iot-priv:hasConsumeRequest ?consumeRequest .
9 }
```

A seguir, *medimos o quanto o nosso método de inferência por consultas do tipo CONSTRUCT impacta no tempo de resposta*. Para isso, executamos duas consultas: o Código 4.18 apresenta uma consulta simples que retorna um conjunto de proprietários de dados e seus respectivos sistemas. Em contrapartida, o Código 4.19 apresenta uma consulta que retornar o mesmo que a consulta anterior, no entanto, para chegar no resultado desejado é utilizado uma propriedade inversa (*iot-priv:isOwnedBy*), sendo assim, a inferência de novos fatos é necessária e precisa ser realizada *antes* de respondê-la.

Código 4.18 Consulta que não necessita de inferência.

```
1 PREFIX iot-priv: <http://example.org/ontology/iot-priv#>
2 SELECT ?owner  ?system
3 WHERE{
4   ?owner iot-priv:owns ?system .
5 }
```

Código 4.19 Consulta que necessita de inferência.

```
1 PREFIX iot-priv: <http://example.org/ontology/iot-priv#>
2 SELECT ?owner  ?system
3 WHERE{
4   ?system iot-priv:isOwnedBy ?owner .
5 }
```

Ao medir o tempo de resposta, percebemos que a consulta referente ao Código 4.18 gastou, em média, **28ms**, enquanto a consulta referente ao Código 4.19 gastou, em média, **362 ms**. Este notável aumento no tempo de resposta está associado diretamente à complexidade para raciocinar e inferir novos fatos a partir dos já existentes, o que no serviço *IoT-PrivServ* é feito através da execução de diversas consultas CONSTRUCT, as quais retornam subgrafos com novas informações, subgrafos estes que são concatenados e posteriormente adicionados à base de dados, gerando assim, um novo conhecimento, antes implícito, como a propriedade inversa *iot-priv:isOwnedBy*, a qual em momento algum foi especificada manualmente, mas é possível utilizá-la, graças ao mecanismo de inferência por consultas CONSTRUCT.

Além disso, o mecanismo de inferência não se limita em apenas inferir as propriedades inversas, mas também, consegue inferir o consentimento de um proprietário de dados (*iot-priv:DataOwner*) a uma requisição de consumo (*iot-priv:ConsumptionRequest*). Desta forma, o *IoT-PrivServ* dispensa a necessidade de um proprietário de dados fornecer ou negar, explicitamente, seu consentimento a uma requisição, fazendo com que esta tarefa se torne um processo automatizado pelo serviço. Um exemplo disso é se uma nova requisição de consumo for registrada, como a do Código 4.14. Para que essa requisição possa efetivamente acessar e consumir o serviço em questão, é necessário que o proprietário forneça seu consentimento para essa requisição, ou seja, registre a tripla (*:UserAndrew, iot-priv:providesConsentTo, :NewConsumptionRequest*), no entanto, essa tripla pode ser inferida pelo mecanismo de inferência, e preenchida automaticamente, sem a necessidade de que o proprietário *:UserAndrew* forneça, explicitamente, seu consentimento.

Neste sentido, a diferença de tempo entre as consultas com e sem inferência é algo explicável, dadas as circunstâncias em que novas informações são produzidas e adicionadas à base de dados, ao mesmo tempo em que a consulta é respondida.

Po fim, o último experimento foi realizado para avaliarmos *o impacto da filtragem espacial e temporal em comparação a uma consulta semântica típica sobre instâncias da ontologia IoT-Priv*. Sendo assim, desenvolvemos uma consulta SPARQL (Q1) que deve retornar uma lista de Destinatários, suas Requisições de Consumo e os respectivos Serviços solicitados. Em seguida, executamos dois filtros (F1 e F2) sobre essa consulta: o primeiro deve retornar somente as requisições de consumo que foram geradas dentro de uma área espacial, e o segundo deve retornar apenas requisições de consumo que ainda não expiraram, ou seja, aquelas que ainda são válidas temporalmente. A consulta Q1, bem como os filtros F1 e F2 são descritos no Código 4.20.

Código 4.20 Consulta SPARQL e os respectivos filtros Espaciais e Temporais

```
1  % ---------------------- Q1 - Consulta SPARQL ----------------------
2  PREFIX iot-priv: <http://example.org/ontology/iot-priv#>
3  SELECT ?recipient ?consumeRequest ?service
4  WHERE {
5     ?recipient iot-priv:hasConsumptionRequest ?consumeRequest.
6     ?consumeRequest iot-priv:consumes ?service;
7                     iot-priv:requestInstant ?instant;
8                     iot-priv:hasOptIn ?policy;
9                     iot-priv:requestLocation ?requestLocation.
10    ?requestLocation iot-priv:hasWKTPoint ?pointWKT.
11      ?policy iot-priv:protects ?service;
12            iot-priv:hasGeneralPrivacyPolicy ?generalPolicy.
13    ?generalPolicy iot-priv:maxRetentionPeriod ?timeDuration;
14                   iot-priv:accessFrom ?accessRestriction.
15    ?accessRestriction iot-priv:hasWKTPolygon ?polygonWKT.
16    ?timeDuration iot-priv:hasDuration ?duration;
17                  iot-priv:hasDurationDescription ?durationDescription.
18 }
19 % ---------------------- F1 - Filtro Espacial ----------------------
20 FILTER (iot-priv:pointWithinPolygon(?pointWKT, ?polygonWKT))
21 % ---------------------- F2 - Filtro Temporal ----------------------
22 FILTER (iot-priv:consentExpired(?instant, ?duration,?durationDescription))
```

A seguir, executamos a consulta Q1 na base de dados, a qual contém os dados referentes aos modelos descritos no Apêndice A. Posteriormente, executamos a consulta Q1 com cada filtro F1 e F2 de forma independente e, finalmente, a consulta original Q1 foi executada simultaneamente com os dois filtros. Os tempos médios de cada configuração são descritos na Figura 4.7.

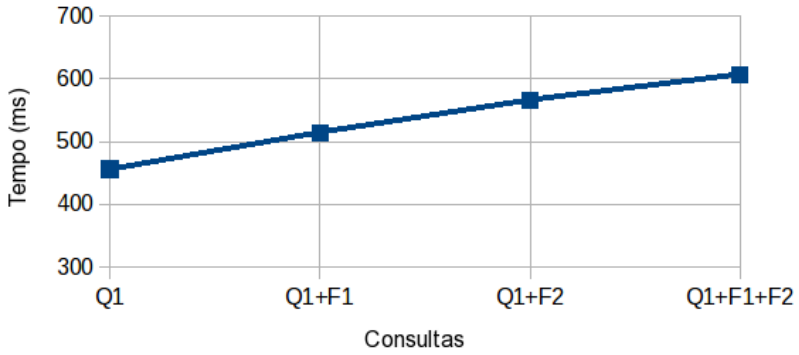

Figura 4.7: *Média dos tempos de resposta de consultas SPARQL com/sem filtros espaciais e temporais.*

Como resultados, percebemos que o tempo médio de resposta para executar Q1 foi ligeiramente inferior à metade de um segundo (**456 ms**). Adicionando o filtro F1 à consulta Q1, o tempo médio de resposta aumentou para **514 ms**, enquanto o filtro F2 sob a consulta Q1 gastou apenas **566 ms** em média. Por fim, medimos um tempo médio de resposta de **607 ms** quando os dois filtros F1 e F2 foram combinados à consulta Q1, simultaneamente.

Em resumo, o aumento no tempo de resposta da consulta Q1, quando executada em conjunto com filtragem espacial e temporal simultânea foi relativamente baixo, aproximadamente **30%**. Esta evolução relativamente pequena no tempo médio de resposta nos dá indicações de que os tempos para processar consultas sobre dados semanticamente anotados com *IoT-Priv* são aceitáveis, mesmo com o processamento da filtragem espacial e temporal, reforçando a hipótese inicial de que a ontologia *IoT-Priv* é leve.

4.4 Considerações Finais

O serviço *IoT-PrivServ* foi projetado com o objetivo de fornecer as principais funcionalidades para aqueles que utilizem a ontologia *IoT-Priv*, tais como validação de modelos, armazenamento e inferência de contexto, além de resposta à consultas, abstraindo das aplicações e/ou usuários a complexidade de realizar tais tarefas.

No entanto, vale ressaltar que nosso objetivo não era construir um serviço completo, repleto de funcionalidades, mas apenas um mínimo que pudesse demonstrar como a ontologia IoT-Priv pudesse ser utilizada por aplicativos na IoT.

Além disso, o *IoT-PrivServ* foi projetado como um *web service RESTful*, disponibilizando suas funcionalidades através de protocolos HTTP, os quais são amplamente difundidos, e consequentemente, o *IoT-PrivServ* torna-se fácil de ser utilizado e integrado a aplicações ou infraestruturas de software, em decorrência de seu projeto arquitetural.

Consequentemente, também realizamos um experimento, cujo objetivo era medir o tempo de resposta de cada funcionalidade fornecidas pelo *IoT-PrivServ*. Neste experimento, obtivemos resultados favoráveis, o que reflete diretamente a hipótese de que a ontologia *IoT-Priv* é leve, já que todas as funcionalidades do serviço são altamente dependentes da ontologia *IoT-Priv*, e consequentemente, de seu desempenho.

No entanto, vale ressaltar que apenas este tipo de avaliação não é suficiente para afirmarmos que, tanto o serviço *IoT-PrivServ*, quanto a ontologia *IoT-Priv* são, realmente, eficientes e leves ao ponto de serem amplamente utilizados em ambientes complexos, como a Internet das Coisas. Isto se deve ao fato de não termos métricas de comparação, como um *benchmark* ou sistemas semelhantes. No entanto, apesar de não ser conclusivo, nosso experimento pode servir como uma referência para sistemas semelhantes que venham a utilizar a ontologia *IoT-Priv*.

CAPÍTULO 5

Trabalhos Relacionados

Este capítulo apresenta os trabalhos relacionados de maneira geral e a sua relação específica com a ontologia de privacidade (*IoT-Priv*) e com o serviço de gerenciamento de privacidade (*IoT-PrivServ*) propostos neste trabalho. Ao fim deste capítulo, também é realizada uma comparação entre os trabalhos relacionados e a abordagem aqui proposta, e a partir dessa análise comparativa, são elencados os pontos positivos, bem como as limitações deste trabalho.

5.1 *LIoPY Ontology*

Com o objetivo de proteger a privacidade durante todo o processo de coleta, transmissão, armazenamento e processamento dos dados coletados por dispositivos inteligentes, Loukil et al. [44] propuseram a ontologia *LIoPY*, a qual visa tornar os dispositivos inteligentes mais autônomos e capazes de inferir direitos de acesso a dados e reforçar a conformidade da política de privacidade no nível de execução.

LIoPY foi construída utilizando a linguagem OWL para definir um vocabulário de privacidade. Além disso, com o objetivo de ser o mais interoperável possível, *LIoPY* importa a ontologia SSN, reaproveitando conceitos e relações associadas a dispositivos e redes de sensores, e a estende, agregando propriedades de privacidade e segurança no contexto da IoT.

Além disso, *LIoPY* foi construída respeitando a característica de modularização, o que é um método comumente usado na engenharia de ontologia para segmentar uma ontologias em partes menores, simplificando o aprendizado e a utilização da mesma, o que impacta significadamente na capacidade de extensão da ontologia, uma vez que estes módulos são fracamente acoplados. *LIoPY* é composta por três módulos principais como é ilustrado na Figura 5.1:

- **IoT Description Module**: descreve o ambiente de IoT de forma geral, incluindo conceitos como Colaboradores, Dispositivos, além dos Resultados, Localização e Recursos associados a estes dispositivos. Além disso, a ontologia SSN foi estendida

com uma nova classe, denominada *Sensed Data*, com o objetivo de facilitar o acesso, uso e verificação das informações geradas pelos recursos da IoT.

- **IoT Resource Management Module**: fornece conceitos e relações baseados em padrões e legislações de privacidade, com o objetivo de possibilitar que o proprietário possa controlar a privacidade associada a seus recursos na IoT. Dentre os padrões e legislações de privacidade incluídos em *LIoPY*, incluem:
  - **ISO 29100** [37]: é uma norma de privacidade que especifica uma terminologia de privacidade, define os atores e seus papéis no processamento de informações pessoalmente identificáveis (PII), descreve considerações de salvaguarda da privacidade e fornece referências a princípios de privacidade voltados para tecnologia da informação.
  - **OECD *Guidelines*** [24]: são um conjunto de diretrizes de privacidade, especificadas pelos países participantes da OECD (tais como Bélgica, Islândia, Espanha, Suíça, entre outros), cujo objetivo é evitar as violações de direitos de privacidade fundamentais, como o armazenamento ilegal de dados pessoais, o armazenamento de dados pessoais imprecisos, o abuso ou a divulgação não autorizada de tais dados, entre outras.
- **IoT Resource Result Sharing Management Module**: fornece conceitos para expressar como os dados devem ser manipulados depois de compartilhados, através de obrigações, criptografia, anonimização de dados e adição de ruído, os quais devem ser aplicados aos dados antes de compartilhá-los com terceiros.

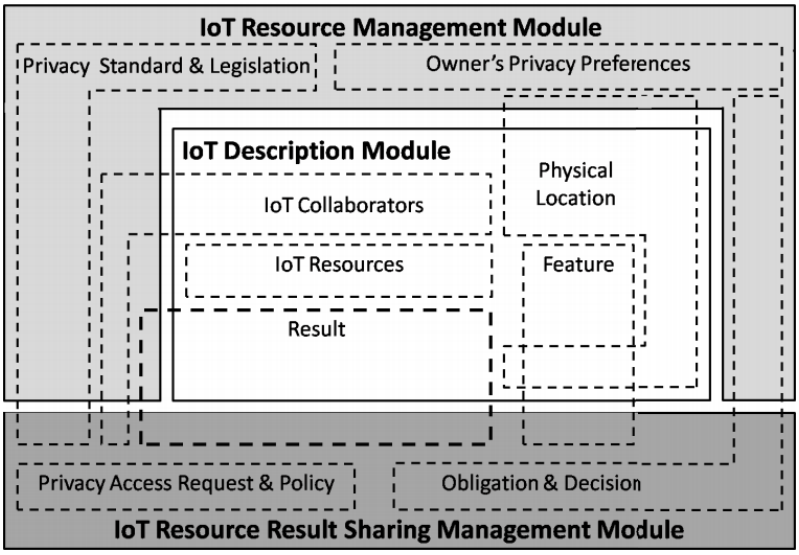

Figura 5.1: *Módulos da ontologia LIoPY [44].*

Por fim, os autores argumentam que algumas relações não podem ser expressas apenas em OWL. Para isso, utilizam a linguagem de regras da Web Semântica (SWRL), as quais expressam um conjunto de regras de inferência, baseadas nos diferentes conceitos e propriedades da ontologia LIoPY. Estas regras estão incorporadas no sistema gerenciador de privacidade, proposto pelos autores, e, em geral, tendem a inferir se um determinado usuário pode obter o consentimento para consumir algum recurso.

A Figura 5.2 ilustra um cenário, onde: um consumidor gera uma requisição no passo (1); a seguir, esta requisição é processada pela máquina de inferência, no passo (2), munido das regras SWRL, como ilustrado no passo (3). A máquina de inferência constrói uma política de privacidade, baseada na ontologia LIoPY, onde é detalhado como, onde, e segundo quais circunstâncias seus dados poderão ser acessados, e a seguir envia está política ao consumidor, no passo (4). Consequentemente, o consumidor só poderá coletar dados armazenados em nuvem, caso concorde com a política de privacidade enviada pelo proprietário.

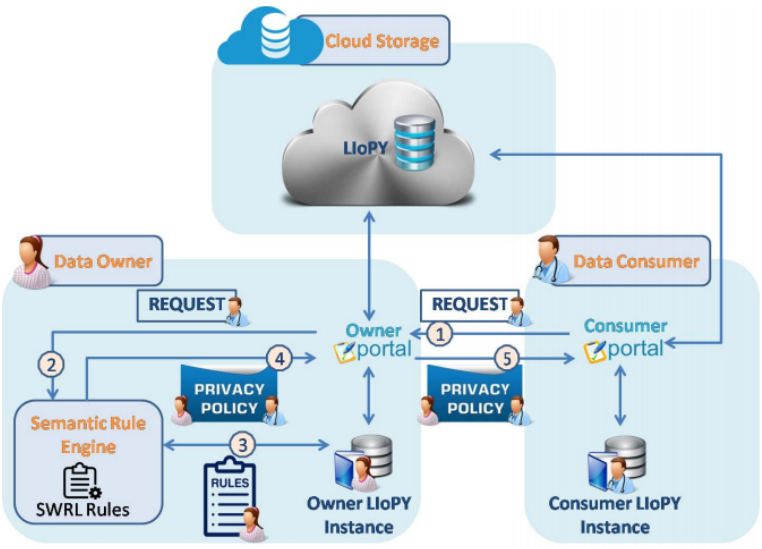

Figura 5.2: *Sistema gerenciador de privacidade, em LIoPY [44].*

Sendo assim, as principais diferenças entre o trabalho de Loukil et al. [44] e a abordagem apresentada neste trabalho, são:

- A LIoPY fornece uma camada onde disponibiliza conceitos e relações associados à padrões e legislações de privacidade, tais como os padrões ISO, OECD, bem como do *Regulamento Europeu de Privacidade*, conceitos estes que não são contemplados em *IoT-Priv*. Resumidamente, LIoPY foca em construir uma ontologia, cujo objetivo maior é disponibilizar conceitos e relações construídos de acordo com as normas e legislações de privacidade, as quais foram definidas previamente por organizações. Já em *IoT-Priv*, o objetivo maior é construir uma ontologia que disponibilize conceitos

de privacidade e suas relações de forma genérica, permitindo que qualquer sistema ou agente de software possa especificar suas políticas da forma como desejarem, sem se preocuparem em estarem de acordo com normas ou legislações específicas.

- LIoPY importa e estende a ontologia SSN, a qual é mais pesada que a ontologia *IoT-Lite*, importada e estendida pela ontologia *IoT-Priv*, o que reflete diretamente na complexidade lógica das ontologias resultantes (LIoPY e *IoT-Priv*).
- O método de inferir o consentimento também difere em ambas abordagens. No trabalho de Loukil et al. [44] é realizado através de regras SWRL, visando uma maior expressividade na especificação das regras, enquanto no serviço *IoT-PrivServ* é realizado por meio de consultas SPARQL do tipo CONSTRUCT, visando um melhor desempenho.
- Em LIoPY, os autores avaliaram a consistência da ontologia, bem como a capacidade de inferir novos fatos através de regras SWRL. Já em *IoT-Priv*, a avaliação se preocupou com a leveza da ontologia, onde medimos suas métricas de tamanho e, principalmente, de complexidade lógica.

5.2 *Data Privacy Ontology*

Evitar que os usuários compartilhem dados não é uma estratégia viável para proteção de privacidade. Assim, Bawany e Shaikh [10] apresentaram uma estratégia, a qual visa tratar os dados como propriedade de uma entidade, os quais serão *protegidos* quando compartilhados. Resumidamente, a estratégia de Bawany e Shaikh compreende um modelo ontológico desenvolvido para representar a privacidade de tal maneira que seja aplicável aos produtores e consumidores de dados (ou serviços), denominado *Data Privacy Ontology*, apresentado na Figura 5.3 e detalhada a seguir:

Em *Data Privacy Ontology*, uma política é um conjunto de regras, as quais são especificadas por um produtor de informações, o qual também é o proprietário dos dados. Esta política é associada diretamente às informações, e dentre outras funcionalidades, é capaz de restringir o acesso aos dados (leitura ou escrita, por exemplo), restringir opções de compartilhamento, restringir localizações de onde os dados podem ser acessados, e quanto tempo uma informação pode ficar sob posse de um consumidor após adquirida. Além disso, o produtor detém os direitos sobre os seus dados, mesmo depois de serem compartilhados com outros usuários, os quais são chamados de *Consumidores* na ontologia, conceitos similares aos que também incluímos em *IoT-Priv*.

A ideia desses autores é que os dados de um produtor só possam ser consumidos por entidades explicitamente autorizadas pelo produtor se, e somente se, satisfizerem os requisitos de privacidade estabelecidos pelo produtor.

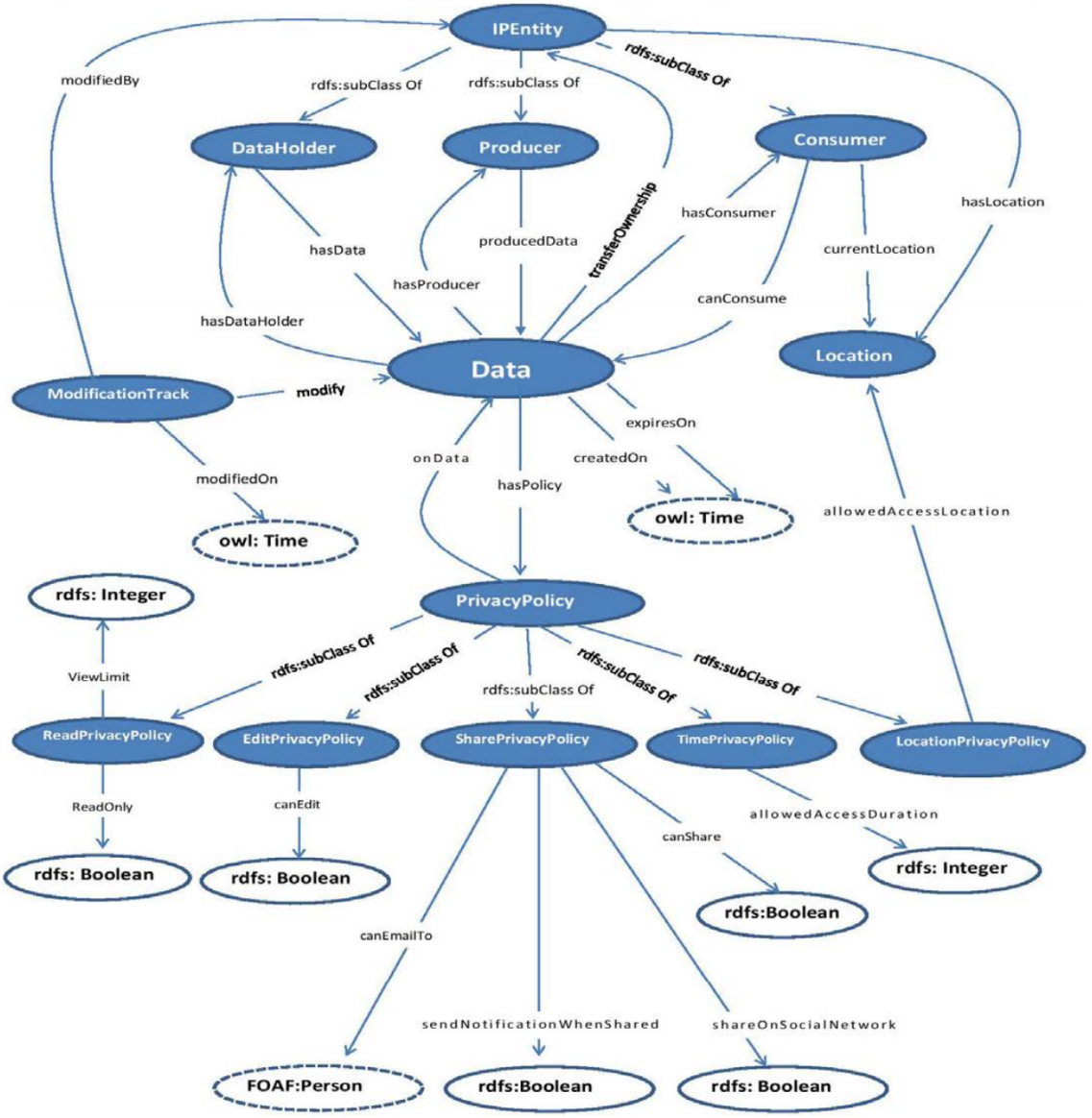

Figura 5.3: *Data Privacy Ontology [10] – visão geral.*

Além da ontologia supracitada, Bawany e Shaikh [10] também apresentaram um sistema para gerenciar a privacidade, como é ilustrado na Figura 5.4. Neste sistema, o produtor de informações (*Producer*) define a política de privacidade associada a seus dados, através do gerenciador de configuração de privacidade (*Privacy Configuration Manager*), o qual gera uma instância da política de privacidade (*DataPrivacyPolicy*), anotada com a ontologia supracitada.

Quando um consumidor qualquer realizar uma requisição de consumo a estes dados, esta requisição só será respondida caso o aplicador de políticas (*Policy Enforcer*) libere o acesso àquelas informações, isto é, se a requisição de consumo estiver de acordo com as restrições definidas na Política de Privacidade. Resumidamente, este sistema aplica as restrições de *Data Privacy Ontology* utilizando uma máquina de inferência.

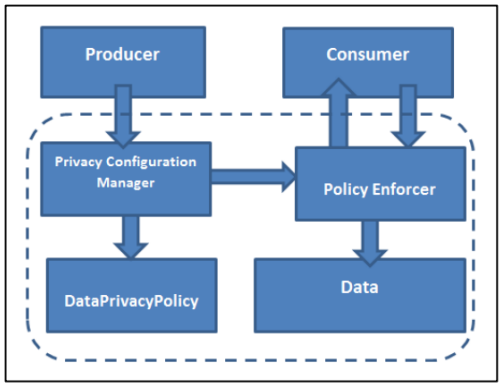

Figura 5.4: *Abordagem proposta por Bawany e Shaikh [10] para registrar e impor a privacidade*

Sendo assim, as principais diferenças entre o trabalho de Bawany e Shaikh [10] e a abordagem apresentada neste trabalho são:

- A ontologia apresentada pelos autores não modela os conceitos relacionados à IoT, tais como dispositivos, sistemas, serviços, entre outros, são modeladas apenas as visões dos produtores e consumidores, os quais podem fazer parte da IoT. Já a ontologia *IoT-Priv* importa a ontologia *IoT-Lite*, e consequentemente, consegue modelar com eficácia os conceitos e relações relevantes para o domínio de IoT, além, dos conceitos e propriedades relevantes para a privacidade.
- A ontologia apresentada pelos autores não cumpre diversos requisitos de privacidade, os quais foram identificados na literatura, e estão incorporados na ontologia *IoT-Priv*, como criptografia, anonimização, associação de valores a serviços, métodos de disputa e resolução de problemas, propósitos de acesso, obrigações, divulgação, entre outros.
- Bawany e Shaikh [10] aplicaram a privacidade através de inferência ontológica, diferentemente de *IoT-PrivServ,* onde a privacidade foi aplicada (ou gerenciada) por meio de consultas CONSTRUCT, e consequentemente, aplicadas pelo processador SPARQL.

5.3 *Semantic Web-based Context Management Framework (SeCoMan)*

Com o objetivo de fornecer uma solução para o desenvolvimento de aplicativos inteligentes sensíveis ao contexto, e que sejam capazes de preservar a privacidade dos usuários na IoT, Celdrán et al. [17] apresentaram o *framework SeCoMan*.

SeCoMan é composto por três partes principais: um conjunto de ontologias, responsáveis por modelar os conceitos de localização *indoor*, bem como o ambiente em que *SeCoMan* será aplicado; as políticas de privacidade, representadas através de regras SWRL; e o módulo de inferência de contexto e de respostas a consultas, denominado *SeCoMan Reasoner*. A Figura 5.5 apresenta o *framework* proposto.

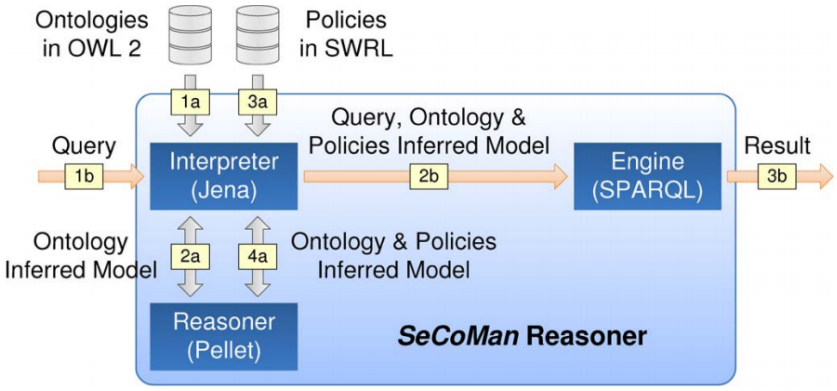

Figura 5.5: *Visão geral do framework SeCoMan [17].*

SeCoMan define uma coleção de ontologias para modelar as informações de espaço e contexto (em 1a). Esta coleção é composta por uma ontologia chamada *Location Ontology*, responsável por modelar os conceitos e relações referentes à localização *indoor*, comum a todos os contextos, e um conjunto de ontologias para as aplicações inteligentes que fornecem serviços específicos em diferentes contextos. Os autores demonstraram a prova de conceito utilizando uma ontologia para o ambiente de supermercados, denominada *Supermarket Ontology*.

SeCoMan ainda especifica um conjunto de regras (em 3a) representando três tipos principais de políticas: *Políticas Operacionais*, as quais são definidas pelos administradores dos aplicativos para produzir novos conhecimentos relacionados ao contexto dos usuários; *Políticas de Autorização*, as quais são definidas pelo administrador do *framework* para conceder ou negar autorização para os usuários permanecerem em um determinado espaço; e as *Políticas de Localização*, as quais são definidas por usuários independentes de aplicativos para especificar as preferências de privacidade sobre sua localização.

Além disso, *SeCoMan* também especifica um conjunto de consultas pré-definidas, as quais são associadas ao domínio de aplicação, no caso, voltadas para o domínio de supermercados (em 1b). Por fim, *SeCoMan* também disponibiliza o módulo *SeCoMan Reasoner*, responsável por responder consultas. Para isso, o módulo recebe como entrada a consulta (1b), as definições especificadas nas ontologias (1a), bem como as políticas especificadas em SWRL (3a), a seguir, realiza inferência ontológica (2a) e inferência baseada em regras SWRL (4a), por fim, o processador SPARQL executa a consulta (2b) e retorna os dados para o usuário solicitante, em (3b).

Sendo assim, as principais diferenças entre SeCoMan e a abordagem apresentada neste trabalho, são:

- Propomos a ontologia *IoT-Priv* como uma ferramenta para que a privacidade em IoT seja expressa, livre de ambiguidade, através de um formalismo ontológico leve. Por sua vez, em *SeCoMan* as políticas de privacidade são representas e processadas através de regras semânticas.
- A ontologia *IoT-Priv* fornece conceitos e relações que permitem que a privacidade seja representada no cenário da IoT de forma genérica. Já em *SeCoMan*, as regras de privacidade são complementares às ontologias de Localização e de Supermercados, proposta pelos autores, de modo que a ontologias definem os cenários com conceitos genéricos, enquanto as regras instanciam e aplicam a privacidade de maneira específica.
- Por fim, em *SeCoMan* há apenas a avaliação do serviço, medindo o tempo médio para responder consultas, similar ao que fizemos em *IoT-PrivServ*. No entanto, em *SeCoMan* esta avaliação é realizada com uma carga crescente de indivíduos instanciados (de 15 mil a 150 mil). Já em *IoT-PrivServ*, realizamos uma avaliação similar, no entanto, a quantidade de indivíduos era reduzida (cerca de 230 triplas).

5.4 *HealthCare Security And Privacy Ontology* (*HCSP*)

Com o objetivo de abordar as necessidades de privacidade das organizações de saúde, bem como as preocupações sobre IoT e políticas de segurança corporativa nos serviços de saúde, Kanaan et al. [39] propuseram a ontologia *HealthCare Security And Privacy* (*HCSP*), uma ontologia de segurança e privacidade para *healthcare*, construída através da importação e extensão da ontologia SSN. A Figura 5.6 ilustra um trecho da ontologia HCSP, como foi apresentada pelos autores.

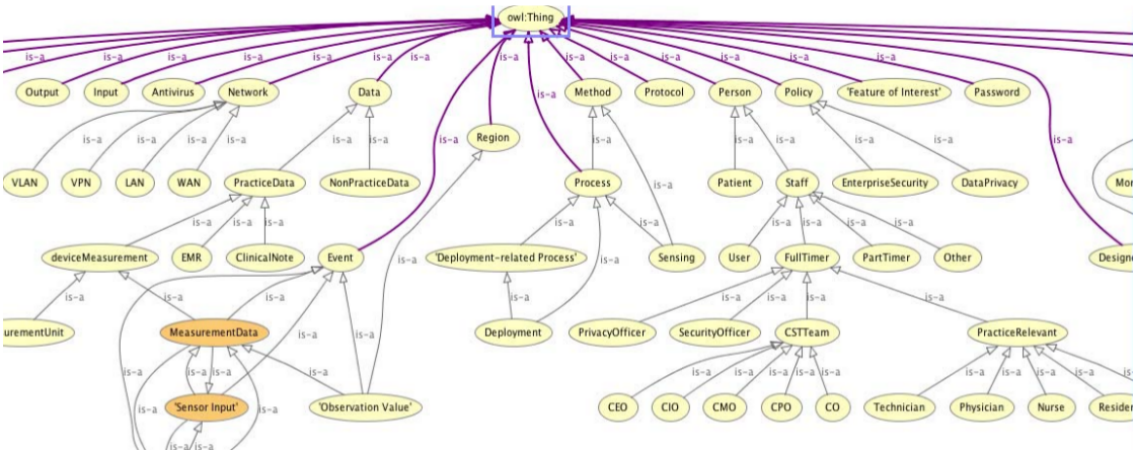

Figura 5.6: *Trecho da ontologia HCSP [39].*

A ontologia *HCSP* conta com três classes principais: Pessoa (*Person*), Dados (*Data*) e Política de Privacidade (*Policy*). A classe Pessoa identifica a pessoa como um empregado de uma organização de saúde ou um paciente. A classe Dados permite a classificação dos dados das pessoas. Por fim, a classe de Política de Privacidade define as políticas de um modo geral.

Além disso, os autores definiram regras baseadas nas políticas de órgãos reguladores, tal como a política do HIPAA dos EUA, regida e mantida pelo Departamento de Saúde e Serviços Humanos dos EUA (HHS). Neste sentido, os autores ainda desenvolveram um conjunto de 18 regras semânticas, escritas em SWRL, para identificar violações de privacidade baseadas na ontologia HCSP. Resumidamente, as regras SWRL verificam a conformidade das instâncias de Pessoa e Dados com os padrões do HIPAA.

Sendo assim, as principais diferenças entre o trabalho de Kanaan et al [39] e a abordagem apresentada neste trabalho, são:

- A ontologia *HCSP*, apesar de modelar conceitos e relações sobre a privacidade, no fim, é uma ontologia específica para o domínio de saúde, enquanto a *IoT-Priv* lida com conceitos de privacidade, independentemente do domínio da aplicação.
- Os autores avaliaram a eficácia da ontologia HCSP em relação à detecção de conformidade com as normas de privacidade de saúde, quando combinada com as regras SWRL por eles desenvolvidas. Já em *IoT-Priv*, nos preocupamos em avaliar, principalmente, a leveza e a extensibilidade do modelo, as manipulações de privacidade (por exemplo, inferência de consentimento) foram realizadas em um serviço externo a ontologia, ou seja, por uma aplicação, munida das definições e propriedades fornecidas na ontologia *IoT-Priv*.
- Além disso, *HCSP* importa e estende a ontologia SSN, maior e mais complexa que a ontologia *IoT-Lite*, importada pela ontologia *IoT-Priv*.
- Por fim, vale ressaltar que Kanaan et al [39] não apresentaram um serviço semelhante ao *IoT-PrivServ*, e consequentemente, não foi possível determinar se HCSP é leve ou não.

5.5 *Privacy Policy Ontology*

Percebendo que as políticas de privacidade de organizações, em geral, são publicadas por meio de extensos documentos de textos, os quais exigem um grande esforço manual e tempo para rastreá-los, interpretá-los e gerenciá-los, Joshi et al. [38] propuseram a *Privacy Policy Ontology*, uma ontologia para converter estes documentos de texto em modelos semânticos de privacidade. As classes principais da ontologia são ilustradas na Figura 5.7:

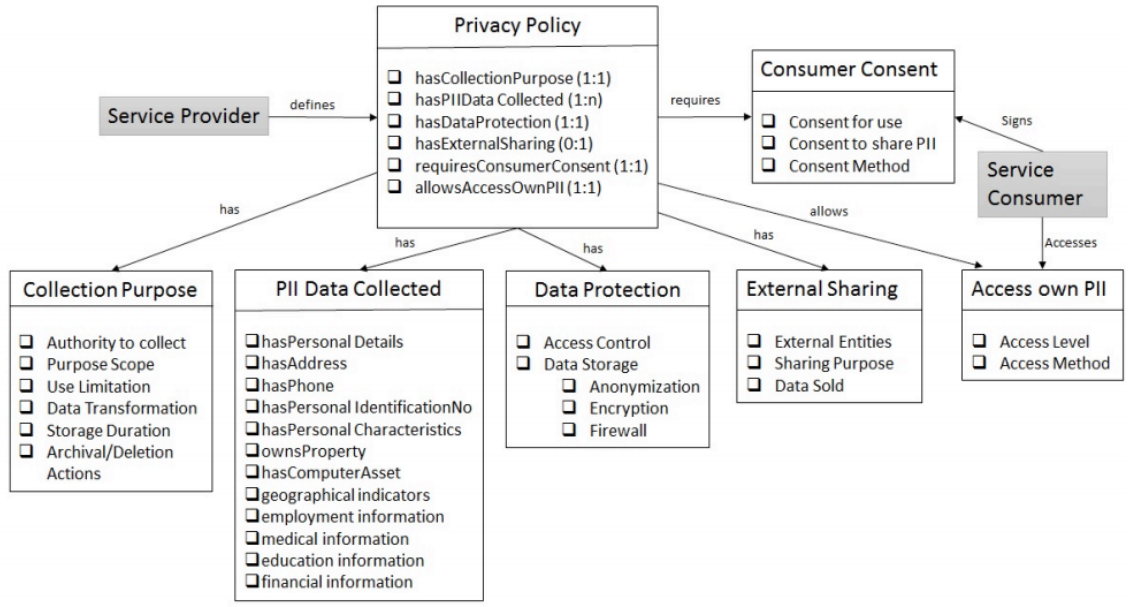

Figura 5.7: *Principais classes da Privacy Policy Ontology [38].*

- ***Privacy Policy***: representa a política de privacidade, de forma geral. Os números entre parênteses indicam a cardinalidade dos relacionamentos especificados na ontologia, portanto, cada política de privacidade deve ter uma instância descrevendo o propósito da coleta (*Collection Purpose*) e os controles de proteção de dados (*Data Protection*); a política de privacidade também deve ter pelo menos uma instância de consentimento do consumidor (*Consumer Consent*), especificação das informações privadas coletadas (*PII Data Collected*), e assim por diante.
- ***Collection Purpose***: captura a finalidade e o escopo da coleta de dados e o uso limitado ao qual os dados serão submetidos. O documento de política também deve especificar a duração em que os dados serão gerenciados pelo coletor de dados e as ações de exclusão e arquivamento que serão tomadas após o término dessa duração.
- ***PII Data Collected***: identifica os principais atributos que contêm informações pessoalmente identificáveis. Estes incluem detalhes pessoais como nomes, informações de contato, como endereço, números de telefone, números de identidade e características de identidade. Outros dados de PII incluem detalhes de emprego, médicos, financeiros e educacionais de uma pessoa.
- ***Data Protection***: inclui propriedades relativas aos controles de acesso a dados e de armazenamento de dados que devem estar em vigor.
- ***External Sharing***: inclui os detalhes das entidades externas com quem os dados serão compartilhados, ou até mesmo vendidos para essas entidades externas.
- ***Consumer Consent***: o consentimento do consumidor deve ser obtido sempre que os dados de PII forem capturados pelo provedor. O consentimento para compartilhar os dados deve ser explicitamente mencionado.

- ***Access own PII***: o consumidor deve poder acessar suas PII que são mantidas pelo provedor. O método de acesso deve ser claramente especificado no documento de privacidade.

Sendo assim, as principais diferenças entre o trabalho de Joshi et al. [38] e a abordagem apresentada neste trabalho são:

- A *Data Privacy Ontology* não importa nenhuma ontologia, consequentemente, apenas os conceitos relacionados à privacidade fornecidos podem não ser suficientes para representar quaisquer cenários da IoT, envolvendo dispositivos, sistemas de dispositivos, consumidores, entre outros.
- A abordagem proposta por Joshi et al. [38] tinha como objetivo automatizar o processo de conversão de documentos textuais, os quais descreviam a privacidade de organizações, para um modelo semântico, baseado na ontologia por eles proposta. Consequentemente, os autores não avaliaram a ontologia, mas sim o processo de conversão, com base em sua assertividade, isto é, avaliando quantas afirmações de privacidade haviam no documento, originalmente, e quantas afirmações foram possíveis modelar com a ontologia. Já nós avaliamos a leveza da ontologia, com o objetivo de possibilitar que esta possa ser amplamente utilizada por sistemas de softwares na IoT.
- O trabalho de Joshi et al. [38] não apresenta um serviço de gerenciamento de privacidade, semelhante ao *IoT-PrivServ*, apresentado neste trabalho.

5.6 *PROACT* Ontology

Visando possibilitar que recursos heterogêneos troquem informações relacionadas à privacidade através de uma linguagem consensual, Panagiotopoulos et al. [52] apresentaram a ontologia *PROACT* (Figura 5.8), a qual contém conceitos e relações para especificar políticas de privacidade em um formato padrão e compreensível por máquinas. Os principais conceitos em *PROACT* são:

- ***Mechanism***: representa o conjunto de ações que podem ocorrer no contexto que descrevemos e temos que fazer principalmente com os dados do sistema.
- ***PolicyMechanisms***: são todos os mecanismos necessários para a implementação de uma política, como controle de acesso.
- ***Resource***: o conjunto dos recursos de um ambiente de computação onipresente, por exemplos, agentes, dispositivos, entre outros.
- ***User***: o usuário incluído no contexto.
- ***ID***: as identidades das entidades do sistema.

Figura 5.8: *Visão geral da ontologia PROACT [52].*

- ***Policy***: todas as políticas que são definidas pelos usuários do sistema de acordo com suas necessidades e preferências.
- ***Time***: é um conceito básico para descrever algumas ações, por exemplo, a duração do armazenamento de dados, ou a validade temporal de uma política.

Além disso, *PROACT* foi enriquecida com um conjunto de regras SWRL, as quais disponibilizam funcionalidades de autenticação e concessão de acesso para aqueles sistemas que utilizem a ontologia desenvolvida pelos autores.

Por fim, a avaliação de *PROACT* foi conduzida apenas com base em suas questões de competência, isto é, dado um conjunto de dados instanciados através de *PROACT*, a máquina de inferência utilizada pelos autores processa as regras SWRL. Neste sentido, os autores avaliaram apenas a assertividade das regras SWRL aplicadas sobre *PROACT*.

Sendo assim, as principais diferenças entre o trabalho de Panagiotopoulos et al. e a abordagem apresentada neste trabalho, são:

- *PROACT* não importa nenhuma ontologia, neste sentido, as relações e conceitos referentes à IoT não foram incluídos.
- *PROACT* agrega regras semânticas à ontologia, o que aumenta a expressividade da mesma. Consequentemente, a determinação de consentimento, definição de proprietários, autenticação, dentre outras funcionalidades é realizada por uma máquina de inferência, uma vez que estas relações foram modeladas como regras SWRL. Já em *IoT-Priv*, as tarefas que exigiram uma maior carga de processamento, tais como a inferência de consentimento (*iot-priv:providesConsentTo*), a definição dos proprietários (*iot-priv:isOwnedBy*), bem como a validação dos modelos foi realizada pelo serviço *IoT-PrivServ*.

5.7 Resumo Comparativo

Esta seção apresenta uma análise comparativa entre os trabalhos supracitados e a abordagem proposta neste trabalho, representada pela díade *IoT-Priv* e *IoT-PrivServ*. A Tabela 5.1 ilustra a comparação entre a ontologia *IoT-Priv* e o serviço *IoT-PrivServ* com os trabalhos relacionados. Para efeitos de simplificação, é utilizada uma notação que relaciona cada critério de comparação a um identificador *Cn*, onde:

C1: A ontologia é exclusivamente para privacidade em IoT?
C2: Quais foram as ontologias importadas e/ou estendidas?
C3: Quais os domínios de aplicação da ontologia?
C4: A ontologia é extensível?
C5: Qual a expressividade da ontologia?
C6: Quais requisitos de privacidade em IoT **não** são tratados pela ontologia?
C7: O serviço baseado na ontologia possui algum módulo que realiza inferência de contexto? Se sim, como?

***Obs*$_1$**: a notação "**–**" significa que não foi possível identificar ou encontrar respostas para aquele critério de avaliação no trabalho avaliado.
***Obs*$_2$**: para o sexto critério de comparação (C6), têm-se a seguinte notação:
- ***I***: denota que não existem conceitos e relações para descrever o cenário de IoT, tais como entidades, dispositivos e serviços.
- ***Au***: denota que não existem conceitos e relações que possibilitem a auditoria, isto é, identificar quais entidades obtiveram dados de outros, quando, onde, e com que propósitos.

Ar: denota que não existem conceitos e relações que possibiltem detalhar políticas de privacidade voltadas para o armazenamento, como quais técnicas de criptografia serão utilizadas, por quanto tempo os dados podem ser amazenados, e quando devem ser excluídos.

L: denota que não existem conceitos e relações para expressar as listas de serviços, isto é, adicionar consumidores a listas que podem conceder ou negar acesso à um determinado dado e/ou serviço.

V: denota que não existem conceitos e relações que possibiltem especificar valores às informações e/ou serviços.

R: denota que não existem conceitos e relações que possibilitem especificar métodos de reclamação e reparação em caso de violação à privacidade.

Tabela 5.1: *Resumo comparativo.*

Critérios / Trabalhos	C1	C2	C3	C4	C5	C6	C7
LIoPY Ontology [44]	✓	SSN	Genérica	✓	Pelo menos $\mathcal{ALCRIQ}$	Au, AR, L, V, R	SWLR
Data Privacy Ontology [10]	✗	✗	Genérica	–	–	I, Ar, L, V, R	SWLR
SeCoMan [17]	✗	✗	Supermercados	✓	–	I, Au, Ar L, V, R	SWLR
HCSP Ontology [39]	✓	SSN	Saúde	✓	Pelo menos $\mathcal{ALCRIQ}$	Au, Ar L, V, R	SWLR
Privacy Policy Ontology [38]	✗	✗	Genérica	–	–	L, V, R	SWLR
Proact [52]	✗	✗	Genérica	–	–	L, V, R	SWLR
IoT-Priv	✓	IoT-Lite	Genérica	✓	$\mathcal{ALUIQ}(\mathcal{D})$	✓	SPARQL CONSTRUCT

Ao analisar os trabalhos relacionados, bem como o critério **C3**, é possível concluir que mesmo existindo ontologias de privacidade para a IoT, a *IoT-Priv* é diferente das demais, pois seu objetivo é fornecer conceitos e relações genéricas, possibilitando que diferentes cenários IoT possam ser descritos semanticamente, através de uma abordagem orientada à privacidade e com um baixo custo computacional associado. *LIoPy* se preocupa com requisitos legais de privacidade. *Data Privacy Ontology* se preocupa em como aplicar a privacidade do ponto de vista de produtores e consumidores, mas exclui demais conceitos referentes a IoT. *SeCoMan* se preocupa em como modelar e aplicar a privacidade baseada em dados temporais e de localização, isto é, determinar quando e onde as entidades podem acessar ou obter um determinado recurso. *HCSP* se preocupa em como atender os requisitos legais de privacidade associados ao domínio de saúde, como a legislação HIPAA. *Privacy Policy Ontology* se preocupa em modelar a privacidade de tal maneira que os documentos textuais, onde estão descritas as políticas de privacidade de organizações possam ser convertidos para um modelo semântico. *PROACT* foi construída para possibilitar que

recursos troquem informações, incluindo as relacionadas à privacidade, através de um formato consensual.

Os critério **C1** e **C4**, quando combinados, nos permitem perceber que os autores de ontologias que são puramente dedicadas à IoT, como LIoPY e HCSP, se preocupam com a extensibilidade das ontologias desenvolvidas, o que é justificado pela dinamicidade dos cenários IoT. Em *IoT-Priv* também nos preocupamos com a extensibilidade, uma vez que projetamos a ontologia de tal maneira que diversas classes foram adicionadas como "pontos de extensão", possibilitando que usuários importem e estendam a *IoT-Priv* de acordo com seus propósitos, como nas classes *iot-priv:AccessPurpose* ou *iot-priv:Currency*.

Por sua vez, os critérios **C1** e **C2**, quando combinados, possibilita perceber que, em geral, trabalhos dedicados à IoT tendem a reutilizar conhecimento da ontologia SSN, como nos trabalhos LIoPY e HCSP. No entanto, como já foi argumentando, a ontologia SSN é pesada, pois possui complexidade lógica do tipo $\mathcal{ALCRIQ}$, o que traduzindo em complexidade computacional, pertence à classe *NEXPTIME-Complete*. Por outro lado, a *IoT-Priv* importa a ontologia IoT-Lite, que diferentemente da ontologia SSN, foi projetada para ser aplicada em cenários de IoT, além de ser mais leve, possuindo complexidade lógica do tipo $\mathcal{ALUI}(\mathcal{D})$, o que traduzindo em complexidade computacional, pertence à classe *PSPACE-Complete*, o que garante um menor custo associado ao processamento e inferência de dados. As complexidades lógicas supracitadas são descritas no critério **C5**.

Além disso, os critério **C1** e **C3**, quando combinados, refletem a necessidade por ontologias de privacidade genéricas, voltadas para a IoT. *HCSP* é voltada para o domínio de saúde, e consequentemente, não é viável para ser ser aplicada em qualquer cenário IoT. Por outro lado, SeCoMan apresenta ontologias voltadas para manusear a privacidade das entidades no cenário de um supermercado. Já trabalhos como *Data Privacy Ontology, Privacy Policy Ontology* e *PROACT*, apesar de modelarem a privacidade de forma genérica, não são dedicados à IoT, por este motivo, diversos conceitos e relações necessárias para este cenário foram excluídos, o que está diretamente associado à não importação de ontologias externas. Dentre os trabalhos analisados, apenas *LIoPY* representa a privacidade em IoT de forma genérica, tal como é realizado em *IoT-Priv*.

Consequentemente, o critério **C6** descreve os requisitos (classes e/ou propriedades) que estão incluídas em *IoT-Priv* e que não foram tratdas pelas ontologias analisadas. Foi possível identificar que **todas** as ontologias analisadas não tratam requisitos referentes a definição de lista de acessos e inclusão de consumidores a estas listas (**L**), a associação de valores a serviços (**V**), bem como a métodos de reclamação e reparação em caso de violação das políticas de privacidade previamente definidas (**R**). Também constatamos que os trabalhos LIoPY, Data Privacy Ontology, SeCoMan e HCSP não incluem conceitos para descrever políticas de privacidade voltadas para o armazenamento de informações (**Ar**). Além disso, os trabalhos LIoPY e SeCoMan também não incluem conceitos voltados para

a auditoria (**Au**). Por fim, os trabalhos Data Privacy Ontology e SeCoMan não incluem conceitos para descrever ambientes IoT (**I**). Estes dados nos permite compreender as principais contribuições de *IoT-Priv* em relação as suas antecessoras, além disso, também foi possível identificar que a ontologia *IoT-Priv* inclue conceitos que não são disponibilizados por nenhuma das ontologias analizadas.

Por fim, constatamos que, além de apresentarem as ontologias de privacidade, diversos trabalhos também propuseram serviços que utilizam estas ontologias para oferecer algum tipo de funcionalidade, como ilustra o critério **C7**. No entanto, estes serviços, em geral, se preocupam apenas em inferir informações baseada nas instâncias desta ontologia. Neste sentido, analisamos a forma como os autores realizam a inferência de contexto, e constatamos que todos os trabalhos utilizam a linguagem SWRL, a qual pode ser processada por máquinas de inferência, como *Pellet* ou *Jess*. No entanto, em nossos experimentos constatamos que esta abordagem necessita de uma grande carga de processamento, mesmo em ontologias leves, como a *IoT-Priv*. Neste sentido, elaboramos uma metodologia para "simular" a inferência de contexto, através de consultas SPARQL do tipo CONSTRUCT, as quais fazem parte do *IoT-PrivServ*.

5.8 Considerações Finais

Este capítulo apresentou uma comparação entre trabalhos relacionados à ontologias e privacidade em IoT com a abordagem proposta nesta dissertação, na forma da ontologia *IoT-Priv* e do serviço de gerenciamento de privacidade *IoT-PrivServ*. Incluindo a proposta desta tese, todos os trabalhos relacionados compartilham do objetivo de apoiar a construção de sistemas IoT.

Com relação à ontologia *IoT-Priv*, foi destacada sua leveza, comprovada pela avaliação de sua expressividade lógica, a qual é inferior a de suas concorrentes. Além disso, também vale ressaltar a sua capacidade de ser extensível e genérica, permitindo que a *IoT-Priv* possa ser aplicada a domínios distintos no contexto de privacidade em IoT. Além disso, a IoT-Priv fornece conceitos e relações, as quais não estão presentes nos modelos analisados, tais como à auditoria de consumo de dados, temporalidade e localização de solicitações de consumo, listas de acesso à serviços e coleta de dados apartir de uma determinada área espacial, além de conceitos relacionados à valores de serviços e métodos de reclamação e reparação em caso de violação de privacidade.

Em relação ao serviço *IoT-PrivServ*, vale destacar suas funcionalidades, entre elas, o método de inferência de contexto baseado em consultas CONSTRUCT's, capaz de inferir o consentimento dos proprietários de dados e popular as propriedades inversas da ontologia. Vale ressaltar que este método foi resultado do refinamento do *IoT-PrivServ*, o qual se mostrou mais eficiente que as tradicionais regras semânticas escritas em SWRL.

CAPÍTULO 6

Conclusões e Trabalhos Futuros

A literatura tem reportado a necessidade de soluções que sejam capazes de garantir que a privacidade do usuário não seja violada na IoT. Neste sentido, o desenvolvimento de modelos de privacidade tem sido cada vez mais considerado como um ponto de partida para lidar com riscos de privacidade em IoT.

Dentre as diversas técnicas de modelagem existentes, as ontologias tem se destacado devido ao formalismo e expressividade fornecidos por esta técnica. No entanto, os modelos ontológicos de privacidade para a IoT devem considerar a baixa demanda de processamento, para isso, as ontologias produzidas para este fim devem ser de baixa complexidade lógica, ou seja, ontologias "leves", uma vez que esta característica impacta diretamente no desempenho de sistemas e/ou serviços que são baseados nestas ontologias.

Sendo assim, o objetivo geral deste trabalho consistiu em elaborar um *modelo ontológico* de *privacidade* para a Internet das Coisas, que seja *expressivo*, *extensível* e ao mesmo tempo *leve* para apoiar o desenvolvimento de aplicações para a IoT.

6.1 Contribuições

Neste trabalho foi apresentada uma *ontologia de privacidade leve para a IoT, denominada IoT-Priv*, a qual foi construída como uma camada de privacidade sobre conceitos fundamentais da IoT, importados de uma ontologia emergente e leve, denominada *IoT-Lite* [11].

Também foi demonstrado como utilizar a ontologia *IoT-Priv* para modelar dois cenário distintos: i) um aplicativo de recomendação de *hashtags*; e ii) um sistema de assistência médica domiciliar. Consequentemente, argumentamos que *a IoT-Priv pode não ser suficiente para atender as demandas de quaisquer cenários IoT*, como ocorreu na modelagem do sistema de assistência médica domiciliar supracitado, onde foi necessário estender a ontologia *IoT-Priv*, adicionando conceitos específicos para este cenário.

Além disso, foi desenvolvido um *serviço de gerenciamento de privacidade*, o qual é responsável por gerenciar requisições de produtores e consumidores que utilizam a ontologia *IoT-Priv* na modelagem de seus dados, denominado *IoT-PrivServ*, o qual fornece

funcionalidades para os consumidores e/ou produtores que utilizam a ontologia *IoT-Priv* na modelagem de seus dados, abstraindo dos usuários a complexidade de realizar tais tarefas, denominado *IoT-PrivServ*;

Por fim, avaliamos a *IoT-Priv* através de suas métricas de tamanho e complexidade lógica. Também avaliamos o serviço *IoT-PrivServ*, medindo o tempo de resposta para executar funções baseadas em IoT-Priv, tais como: validação de modelos semânticos, inferência de consentimento, responder consultas sobre políticas de privacidade, com ou sem filtros espaciais e temporais, entre outros. *Os resultados da avaliação de IoT-Priv e IoT-PrivServ são complementares, e apontam que mantivemos a característica de leveza presente em IoT-Lite*, a qual era um de nossos objetivos iniciais. Além disso, demonstramos que a *IoT-Priv é expressiva e extensível*, uma vez que seus conceitos possibilitam que cenários complexos sejam modelados, e caso seja necessário, *os pontos de extensões incluídos na ontologia permitem que ela seja importada e estendida para atender necessidades mais específicas*.

Resumidamente, são contribuições desta pesquisa, por ordem de importância:

1. A ontologia *IoT-Priv*, a qual demonstramos ser *extensível* e *leve*;
2. A demonstração de como como estender a ontologia proposta e modelar a privacidade em cenários de recomendação de *hashtags* e assistência médica domiciliar;
3. Um serviço de gerenciamento de privacidade, denominado *IoT-PrivServ*;
4. A avaliação do *IoT-PrivServ* com relação ao comportamento temporal ao executar suas funcionalidades;
5. A compilação de 27 requisitos de privacidade para a IoT, dos quais 3 são contribuições deste trabalho.

6.2 Limitações

A *IoT-Priv* não é completa, uma vez que: não modelamos legislações e padrões de privacidade, como foi realizado em *LIoPY*, não incluímos políticas para cenários específicos, como foi realizado em *HCSP*, e também não especificamos em grandes detalhes o que são as informações pessoalmente identificáveis, como foi realizado em *Privacy Policy Ontology*.

No entanto, vale ressaltar que nosso objetivo não era construir uma ontologia de privacidade que abrangesse a todos os conceitos possíveis, mas sim uma ontologia que fornecesse uma base conceitual, através da qual os usuários pudessem gerar suas próprias políticas de privacidade, utilizando a semântica fornecida por *IoT-Priv*.

Com relação à verificação de *IoT-Priv*, vale ressaltar que devido à escassez de conjuntos de dados estruturados e disponíveis publicamente, a única opção foi gerar os modelos manualmente, através da descrição de cenários identificados na literatura. Embora esta abordagem seja suficiente para demonstrarmos como construir modelos semânticos com a ontologia proposta, não é possível garantir que quaisquer usuários conseguirão fazer o mesmo, isto é, utilizar a ontologia *IoT-Priv* e gerar modelos semelhantes (ou exatamente iguais) para um mesmo cenário. Neste sentido, a verificação deveria ter sido realizada a partir de um processo automatizado, onde para cada elemento de entrada (p.ex.: um documento de privacidade escrito em texto plano), deveria existir uma saída associada (p.ex.: um modelo semântico de privacidade anotado com a ontologia *IoT-Priv*).

Por outro lado, vale ressaltar que a avaliação de *IoT-Priv* e *IoT-PrivServ* não foi muito conclusiva. Em *IoT-Priv*, avaliamos suas métricas, em especial, a complexidade lógica. No entanto, para fins de comparação, tínhamos acesso apenas à complexidade lógica das ontologias *SSN* e *IoT-Lite*, o que limitou a avaliação de *IoT-Priv*, pois sem maiores informações, fomos capazes apenas de confirmar que a ontologia *IoT-Priv* e tão leve quanto a *IoT-Lite*. No entanto, não fomos capazes de mensurar o quão leve a ontologia *IoT-Priv* é, se comparada aos demais modelos ontológicos de privacidade, como *HCSP*, *Data Privacy Ontology*, *PROACT*, uma vez que os autores destas ontologias não se preocuparam em calcular esta métrica.

Consequentemente, ao avaliar o serviço *IoT-PrivServ* o problema foi semelhante. Medimos o tempo de resposta das diversas funcionalidades fornecidas pelo serviço, no entanto, como não existe um *benchmark* para sistemas desta natureza, não fomos capazes de concluir se os tempos mensurados na avaliação do serviço são baixos como esperávamos. No entanto, vale ressaltar que estes dados podem ser úteis para a avaliação de próximas versões do serviço (p.ex.: em trabalhos futuros, comparando se houve alguma melhoria).

Resumidamente, são limitação desta pesquisa:

1. A ausência de conceitos relacionados a legislações de privacidade em *IoT-Priv*;
2. A ausência de conceitos relacionados à segurança em *IoT-Priv*, tais como autenticação, controle de acesso, entre outros, uma vez que a segurança e privacidade são conceitos complementares;
3. Apesar do método de inferência por consultas CONSTRUCT ser mais rápido se comparado a métodos tradicionais de inferência por regras SWRL, para alguns casos pode não fornecer a expressividade necessária (p.ex.: se for necessário utilizar funções pré-definidas da linguagem SWRL).
4. A verificação da ontologia *IoT-Priv* é limitada, em virtude da escassez de dados estruturados e disponíveis para acesso. Além disso, a verificação de *IoT-Priv* foi realizada manualmente pelo próprio autor da ontologia, o que não garante que terceiros também o farão da forma correta.

5. A avaliação da ontologia *IoT-Priv* também é limitada, em virtude da escassez de ontologias relacionadas que relatam sua complexidade lógica, para fins de comparação entre os modelos.
6. Por fim, a avaliação do serviço *IoT-PrivServ* também é limitada, uma vez que não existem *benchmarks* de sistemas semelhantes para que possamos comparar o real desempenho da ontologia *IoT-Priv* quando utilizada em conjunto com sistemas de softwares que realizam uma grande carga de processamento.

6.3 Trabalhos Futuros

No intuito de tratar as limitações da ontologia *IoT-Priv* e do serviço *IoT-PrivServ* propostos, dando continuidade a esta pesquisa, são sugeridos os seguintes trabalhos futuros:

- Estender a ontologia *IoT-Priv*, incluindo conceitos e relações que possibilitem construir políticas de privacidade adequadas a legislações e padrões de privacidade.
- Estender a ontologia *IoT-Priv*, incrementando a classe *iot-priv:PrivacyPolicy* com políticas mais específicas que as gerais (*iot-priv:GeneralPrivacyPolicy*) e as de armazenamento (*iot-priv:StoragePrivacyPolicy*).
  O *framework* PbD enfatizam que a privacidade deve ser considera em todas as etapas do ciclo de vida de contexto, tais como a inferência, agregação, coleta e disseminação. Consequentemente, existem trabalhos como [54] [53] que descrevem requisitos de privacidade específicos para estas etapas, os quais poderiam ser incluídos na ontologia *IoT-Priv*, uma vez que não foram considerados ao longo deste trabalho (p.ex.: classes e relações que descrevam as políticas de privacidade de inferência ou de agregação)
- A criação e a publicação de conjuntos de dados estruturados de privacidade em IoT, bem como o desenvolvimento de técnicas que automatizem a população de modelos ontológicos.
- Estender o serviço *IoT-PrivServ*, adicionando uma camada de regras SWRL, a qual complementará a inferência por consultas CONSTRUCT, quando necessário.
- Implantar o serviço *IoT-PrivServ* na infraestrutura de software Hermes.
  Resumidamente, Hermes é uma infraestrutura de software para gerenciamento de informação contextual [62] [46], composta por 4 (quatro) principais componentes: *Hermes Widget*, responsável por produzir modelos semânticos através de ontologias; *Hermes Aggregator*, responsável por agregar contexto semelhante; *Hermes Interpreter*, responsável por gerenciar filtros e regras semânticas, através do processamento de contexto; e *Hermes History*, responsável pelo armazenamento e consulta a dados semânticos. No entanto, Hermes não se preocupa em gerenciar a privacidade dos

usuários, neste sentido o serviço *IoT-PrivServ* poderia ser integrado à infraestrutura, fornecendo suporte à privacidade para seus componentes. A Figura 6.1 apresenta uma sugestão de como o *IoT-PrivServ* poderia ser incluído nesta infraestrutura, fornecendo suporte à privacidade para todos os componentes.

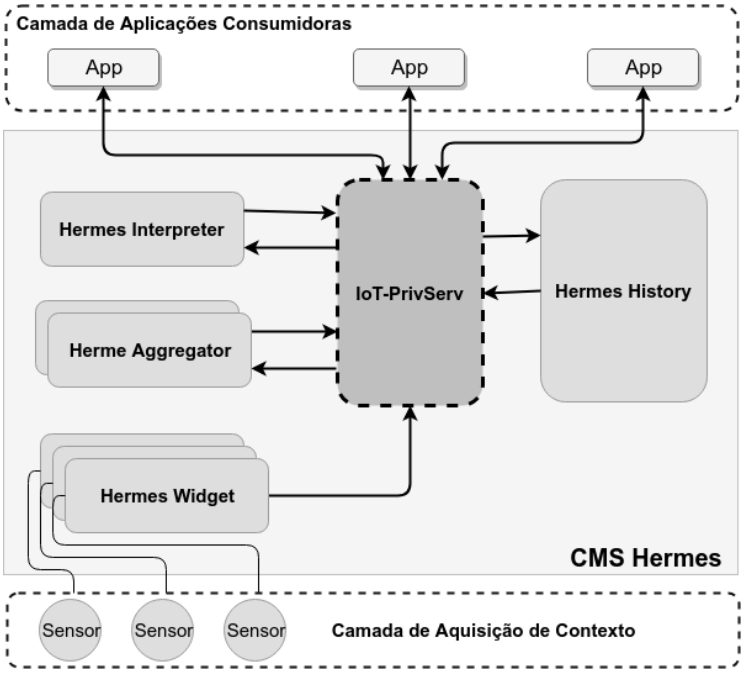

Figura 6.1: *Sugestão de inclusão do IoT-PrivServ à infraestrutura Hermes.*

6.4 Publicações relacionadas a este trabalho

Por fim, vale ressaltar que até o momento foram geradas três publicações derivadas deste trabalho, das quais duas foram apresentadas em conferência B2 (WebMedia '17 e '18) e uma delas foi aceita e será apresentada em conferência A1 (SAC '19).

1. ARRUDA, M.F.; BULCÃO-NETO, R.F. ***Toward a Lightweight Ontology for Privacy Protection in IoT***. *In Proceedings of the Symposium on Applied Computing (SAC '19)*, 2019.
2. VEIGA, E.F.; de FERREIRA, L.B.M; ARRUDA, M.F.; MACEDO, A.A.; BULCÃO-NETO, R.F. ***A Lightweight Mobile Service for Context Representation through an IoT-oriented Ontology***. *In Proceedings of the 24th Brazilian Symposium on Multimedia and the Web (WebMedia '18)*, p. 299-306, 2018.
3. VEIGA, E.F.; ARRUDA, M.F.; BATISTA-NETO, J.A.; BULCÃO-NETO, R.F. ***An Ontology-based Representation Service of Context Information for the Internet of Things.*** *In Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web (WebMedia '17)*, p. 301-308, 2017.

Referências Bibliográficas

[1] ABOWD, G.; DEY, A.; BROWN, P.; DAVIES, N.; SMITH, M.; STEGGLES, P. **Towards a better understanding of context and context-awareness**. In: *Handheld and ubiquitous computing*, p. 304–307. Springer, 1999.

[2] ABOWD, G. D.; MYNATT, E. D. **Charting past, present, and future research in ubiquitous computing**. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):29–58, 2000.

[3] ABOWD, G. D.; MYNATT, E. D.; RODDEN, T. **The human experience [of ubiquitous computing]**. *IEEE pervasive computing*, 1(1):48–57, 2002.

[4] AHAMED, S. I.; TALUKDER, N.; KAMEAS, A. D. **Towards privacy protection in pervasive healthcare**. 2007.

[5] AMERICAN INSTITUTE OF CERTIFIED PUBLIC ACCOUNTANTS.; CANADIAN INSTITUTE OF CHARTERED ACCOUNTANTS. **Generally accepted privacy principles: Cpa and ca practitioner version**, 2009.

[6] ARRUDA, M. F.; BULCÃO NETO, R. F. **Toward a lightweight ontology for privacy protection in iot**. In: *Proceedings of ACM SAC Conference*, SAC '19, New York, NY, USA, 2019. ACM.

[7] ATZORI, L.; IERA, A.; MORABITO, G. **The internet of things: A survey**. *Computer networks*, 54(15):2787–2805, 2010.

[8] BAADER, F.; CALVANESE, D.; MCGUINNESS, D.; PATEL-SCHNEIDER, P.; NARDI, D. **The description logic handbook: Theory, implementation and applications**. Cambridge university press, 2003.

[9] BATTLE, R.; KOLAS, D. **Geosparql: enabling a geospatial semantic web**. *Semantic Web Journal*, 3(4):355–370, 2011.

[10] BAWANY, N. Z.; SHAIKH, Z. A. **Data privacy ontology for ubiquitous computing**. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 8(1):145–149, 2017.

[11] BERMUDEZ-EDO, M.; ELSALEH, T.; BARNAGHI, P.; TAYLOR, K. **Iot-lite: a lightweight semantic model for the internet of things and its use with dynamic semantics**. *Personal and Ubiquitous Computing*, 21(3):475–487, 2017.

[12] BORGIA, E. **The internet of things vision: Key features, applications and open issues**. *Computer Communications*, 54:1–31, 2014.

[13] BORST, W. N. **Construction of engineering ontologies for knowledge sharing and reuse**. Centre for Telematics and Information Technology, 1997.

[14] BULCÃO-NETO.; PIMENTEL, M. G. C. **Toward a domain-independent semantic model for context-aware computing**. In: *Third Latin American Web Congress (LA-WEB'2005)*, p. 10 pp.–, Oct 2005.

[15] CANADIAN INSTITUTE OF CHARTERED ACCOUNTANTS. **Generally accepted privacy principles**. Technical report, 2009.

[16] CAVOUKIAN, A. **Privacy by design, the 7 foundational principles**. https://www.iab.org/wp-content/IAB-uploads/2011/03/fred_carter.pdf, 2009. acessado em em setembro, 2018.

[17] CELDRÁN, A. H.; CLEMENTE, F. J. G.; PÉREZ, M. G.; PÉREZ, G. M. **Seco-man: A semantic-aware policy framework for developing privacy-preserving and context-aware smart applications.** *IEEE Systems Journal*, 10(3):1111–1124, 2016.

[18] COMPTON, M.; BARNAGHI, P.; BERMUDEZ, L.; GARCÍA-CASTRO, R.; CORCHO, O.; COX, S.; GRAYBEAL, J.; HAUSWIRTH, M.; HENSON, C.; HERZOG, A. **The ssn ontology of the w3c semantic sensor network incubator group**. *Web semantics: science, services and agents on the World Wide Web*, 17:25–32, 2012.

[19] DAVIES, N.; MITCHELL, K.; CHEVERST, K.; BLAIR, G. **Developing a context sensitive tourist guide**. In: *1st Workshop on Human Computer Interaction with Mobile Devices, GIST Technical Report G98-1*, volume 1, 1998.

[20] DEY, A.; ABOWD, G.; WOOD, A. C. **Cyberdesk: a framework for providing self-integrating context-aware services**. In: *International Conference on Intelligent User Interfaces (IUI'98)*, p. 47–54, 1998.

[21] DEY, A. K.; ABOWD, G. D.; SALBER, D. **A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications**. *Human-computer interaction*, 16(2):97–166, 2001.

[22] EVANS, D. **The internet of things: How the next evolution of the internet is changing everything**. *CISCO white paper*, 1(2011):1–11, 2011.

[23] FIESTA-IOT. **The m3-lite taxonomy**. http://ontology.fiesta-iot.eu/ontologyDocs/fiesta-iot/doc, 2016. acessado em novembro, 2018.

[24] FOR ECONOMIC CO-OPERATION, O.; DEVELOPMENT. **OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data**. OECD Publishing, 2002.

[25] FORCE, I. E. T. **Rfc 3987 - internationalized resource identifiers (iris)**. http://tools.ietf.org/html/rfc3987, 2005. acessado em em novembro, 2018.

[26] FOUKIA, N.; BILLARD, D.; SOLANA, E. **Pisces: A framework for privacy by design in iot**. In: *Privacy, Security and Trust (PST), 2016 14th Annual Conference on*, p. 706–713. IEEE, 2016.

[27] GARCIA, D.; TOLEDO, M. B. F.; CAPRETZ, M. A.; ALLISON, D. S.; BLAIR, G. S.; GRACE, P.; FLORES, C. **Towards a base ontology for privacy protection in service-oriented architecture**. In: *Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference on*, p. 1–8. IEEE, 2009.

[28] GOMES, F. A.; VIANA, W.; ROCHA, L. S.; TRINTA, F. **A contextual data offloading service with privacy support**. In: *Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web*, Webmedia '16, p. 23–30, New York, NY, USA, 2016. ACM.

[29] GÓMEZ-PÉREZ, A.; FERNÁNDEZ-LÓPEZ, M.; CORCHO, O. **Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web**. Springer Science & Business Media, 2006.

[30] GRUBER, T. R. **Toward principles for the design of ontologies used for knowledge sharing?** *International journal of human-computer studies*, 43(5-6):907–928, 1995.

[31] GRUBER, T. **What is an ontology**. http://www-ksl.stanford.edu/kst/whatis-an-ontology.html, 1993. acessado em outubro, 2018.

[32] GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. **Internet of things (iot): A vision, architectural elements, and future directions**. *Future generation computer systems*, 29(7):1645–1660, 2013.

[33] HE, Q.; ANTÓN, A. I. **A framework for modeling privacy requirements in role engineering**. In: *Proceedings of Requirements Engineering Foundation for Software Quality*, volume 3, p. 137–146, 2003.

[34] HE, Y.; XIANG, Z.; ZHENG, J.; LIN, Y.; OVERTON, J. A.; ONG, E. **The extensible ontology development (xod) principles and tool implementation to support ontology interoperability**. *Journal of biomedical semantics*, 9(1):3, 2018.

[35] HEBELER, J.; FISHER, M.; BLACE, R.; PEREZ-LOPEZ, A. **Semantic web programming**. John Wiley & Sons, 2011.

[36] HOEPMAN, J.-H. **Privacy design strategies**. In: *IFIP International Information Security Conference*, p. 446–459. Springer, 2014.

[37] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 29100:2011 - Information technology – Security techniques – Privacy framework**. 2011.

[38] JOSHI, K. P.; GUPTA, A.; MITTAL, S.; PEARCE, C.; JOSHI, A.; FININ, T. **Semantic approach to automating management of big data privacy policies**. In: *Big Data (Big Data), 2016 IEEE International Conference on*, p. 482–491. IEEE, 2016.

[39] KANAAN, H.; MAHMOOD, K.; SATHYAN, V. **An ontological model for privacy in emerging decentralized healthcare systems**. In: *Autonomous Decentralized System (ISADS), 2017 IEEE 13th International Symposium on*, p. 107–113. IEEE, 2017.

[40] KEPLER, F. N.; PAZ-TRILLO, C.; RIANI, J.; RIBEIRO, M. M.; DELGADO, K. V.; DE BARROS, L. N.; WASSERMANN, R. **Classifying ontologies.** In: *WONTO*, 2006.

[41] KHARBAT, F.; EL-GHALAYINI, H. **Building ontology from knowledge base systems**. In: *Data Mining in Medical and Biological Research*. InTech, 2008.

[42] LEFORT, L. **Ontology for quantity kinds and units: units and quantities definitions**. https://www.w3.org/2005/Incubator/ssn/ssnx/qu/qu-rec20.html, 2005. acessado em novembro, 2018.

[43] LEFORT, L.; HENSON, C.; TAYLOR, K.; BARNAGHI, P.; COMPTON, M.; CORCHO, O.; GARCIA-CASTRO, R.; GRAYBEAL, J.; HERZOG, A.; JANOWICZ, K. **Semantic sensor network xg final report**. *W3C Incubator Group Report*, 28, 2011.

[44] LOUKIL, F.; GHEDIRA-GUEGAN, C.; BOUKADI, K.; BENHARKAT, A. N. **Liopy: A legal compliant ontology to preserve privacy for the internet of things**. In: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, p. 701–706. IEEE, 2018.

[45] MAKSIMOVIĆ, M.; VUJOVIĆ, V. **Internet of things based e-health systems: ideas, expectations and concerns**. In: *Handbook of Large-Scale Distributed Computing in Smart Healthcare*, p. 241–280. Springer, 2017.

[46] MARANHÃO, G. M.; BULCÃO-NETO, R. F. **A semantic filtering mechanism geared towards context dissemination in ubiquitous environments**. *Journal of Universal Computer Science*, 22(8):1123–1147, aug 2016.

[47] MIORANDI, D.; SICARI, S.; DE PELLEGRINI, F.; CHLAMTAC, I. **Internet of things: Vision, applications and research challenges**. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[48] MORAIS, E. A. M.; AMBRÓSIO, A. P. L. **Ontologias: conceitos, usos, tipos, metodologias, ferramentas e linguagens**. *Relatório Técnico–RT-INF-001/07, dez*, 2007.

[49] NGU, A. H.; GUTIERREZ, M.; METSIS, V.; NEPAL, S.; SHENG, Q. Z. **Iot middleware: A survey on issues and enabling technologies**. *IEEE Internet of Things Journal*, 4(1):1–20, 2017.

[50] NOY, N. F.; MCGUINNESS, D. L. **Ontology development 101: A guide to creating your first ontology**. Technical report, Stanford Medical Informatics, CA, 2001.

[51] OPEN GEOSPATIAL CONSORTIUM. **Geographic information - well-known text representation of coordinate reference systems**. http://www.opengis.net/doc/IS/wkt-crs/1.0, 2015. acessado em em julho, 2018.

[52] PANAGIOTOPOULOS, I.; SEREMETI, L.; KAMEAS, A.; ZORKADIS, V. **Proact: An ontology-based model of privacy policies in ambient intelligence environments**. In: *Informatics (PCI), 2010 14th Panhellenic Conference on*, p. 124–129. IEEE, 2010.

[53] PERERA, C. **Privacy guidelines for internet of things: A cheat sheet**. *arXiv preprint arXiv:1708.05261*, 2017.

[54] PERERA, C.; BARHAMGI, M.; BANDARA, A. K.; AJMAL, M.; PRICE, B.; NUSEIBEH, B. **Designing privacy-aware internet of things applications**. *arXiv preprint arXiv:1703.03892*, 2017.

[55] PERERA, C.; LIU, C.; RANJAN, R.; WANG, L.; ZOMAYA, A. Y. **Privacy-knowledge modeling for the internet of things: A look back**. *Computer*, 49(12):60–68, 2016.

[56] PERERA, C.; ZASLAVSKY, A.; CHRISTEN, P.; GEORGAKOPOULOS, D. **Context aware computing for the internet of things: A survey**. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014.

[57] SCHILIT, B. N.; THEIMER, M. M. **Disseminating active map information to mobile hosts**. *IEEE network*, 8(5):22–32, 1994.

[58] SLIMANI, T. **A study investigating typical concepts and guidelines for ontology building**. *arXiv preprint arXiv:1509.05434*, 2015.

[59] TARTIR, S.; ARPINAR, I. B. **Ontology evaluation and ranking using ontoqa**. In: *International Conference on Semantic Computing (ICSC 2007)*, p. 185–192, Sept 2007.

[60] VEIGA, E. F.; ARRUDA, M. F.; BATISTA-NETO, J. A.; BULCÃO NETO, R. F. **An ontology-based representation service of context information for the internet of things**. In: *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, WebMedia '17, p. 301–308, New York, NY, USA, 2017. ACM.

[61] VEIGA, E. F.; FERREIRA, L. B. M.; ARRUDA, M. F.; MACEDO, A. A.; BULCÃO NETO, R. F. **A lightweight mobile service for context representation through an iot-oriented ontology**. In: *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, WebMedia '18, p. 299–306, New York, NY, USA, 2018. ACM.

[62] VEIGA, E. F.; MARANHAO, G.; BULCAO-NETO, R. **Apoio ao desenvolvimento de aplicações de tempo real sensíveis a contexto semântico**. In: *IX Workshop de Teses e Dissertações do XX Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia)*, p. 1–4, 2014.

[63] VERMESAN, O.; FRIESS, P.; GUILLEMIN, P.; GUSMEROLI, S.; SUNDMAEKER, H.; BASSI, A.; JUBERT, I. S.; MAZURA, M.; HARRISON, M.; EISENHAUER, M.; OTHERS. **Internet of things strategic research roadmap**. *Internet of Things-Global Technological and Societal Trends*, 1:9–52, 2011.

[64] W3C. **Basic geo (wgs84 lat/long) vocabulary**. https://www.w3.org/2003/01/geo/, 2003. acessado em novembro, 2018.

[65] W3C. **Notation3 (n3): A readable rdf syntax**. https://www.w3.org/TeamSubmission/n3/, 2011. acessado em em novembro, 2018.

[66] W3C. **Owl 2 web ontology language primer (second edition)**. http://www.w3.org/TR/owl-primer/, 2012. acessado em em novembro, 2018.

[67] W3C. **Sparql 1.1 query language**. http://www.w3.org/TR/sparql11-query/, 2013. acessado em em novembro, 2018.

[68] W3C. **Json-ld 1.0, a json-based serialization for linked data**. https://www.w3.org/TR/json-ld/, 2014. acessado em em novembro, 2018.

[69] W3C. **Rdf 1.1 concepts and abstract syntax**. http://www.w3.org/TR/rdf11-concepts/, 2014. acessado em em novembro, 2018.

[70] W3C. **Rdf 1.1 n-triples, a line-based syntax for an rdf graph**. https://www.w3.org/TR/n-triples/, 2014. acessado em em novembro, 2018.

[71] W3C. **Rdf 1.1 turtle: Terse rdf triple language.** http://www.w3.org/TR/turtle/, 2014. acessado em em novembro, 2018.

[72] W3C. **Rdf 1.1 xml syntax**. https://www.w3.org/TR/rdf-syntax-grammar/, 2014. acessado em em novembro, 2018.

[73] W3C. **Rdf schema 1.1**. http://www.w3.org/TR/rdf-schema/, 2014. acessado em em novembro, 2018.

[74] WEISER, M. **The computer for the 21st century.** *Scientific American*, 265(3):66–75, 1991.

[75] YAQOOB, I.; AHMED, E.; HASHEM, I. A. T.; AHMED, A. I. A.; GANI, A.; IMRAN, M.; GUIZANI, M. **Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges**. *IEEE Wireless Communications*, 24(3):10–16, 2017.

[76] YE, J.; DASIOPOULOU, S.; STEVENSON, G.; MEDITSKOS, G.; KONTOPOULOS, E.; KOMPATSIARIS, I.; DOBSON, S. **Semantic web technologies in pervasive computing: A survey and research roadmap**. *Pervasive and Mobile Computing*, 23:1–25, 2015.

[77] ZIEGELDORF, J. H.; MORCHON, O. G.; WEHRLE, K. **Privacy in the internet of things: threats and challenges**. *Security and Communication Networks*, 7(12):2728–2742, 2014.

APÊNDICE A

Modelos Ontológicos

A.1 Cenário MyPhotos

Código A.1 Modelagem do cenário do aplicativo *MyPhotos* serializada em Turtle.

```
@prefix geo:        <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix :           <http://example.org/instances/iot-priv#> .
@prefix wkt:        <http://www.opengis.net/ont/sf#> .
@prefix iot-lite:   <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .
@prefix owl:        <http://www.w3.org/2002/07/owl#> .
@prefix rdf:        <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:        <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:       <http://www.w3.org/2000/01/rdf-schema#> .
@prefix iot-priv:   <http://example.org/ontology/iot-priv#> .
@prefix ssn:        <http://purl.oclc.org/NET/ssnx/ssn#> .

:Request_002
        a       owl:NamedIndividual , iot-priv:ConsumptionRequest ;
        iot-priv:hasOptIn :Alexandre_PrivacyPolicy ;
        iot-priv:requestInstant
                "2018-02-18T20:20:25"^^xsd:dateTime ;
        iot-priv:requestLocation
                :Shopping_Location ;
        iot-priv:requestPurpose
                :HashTags_Recommend .

:MyPhotos_GeneralPrivacy
        a       owl:NamedIndividual , iot-priv:GeneralPrivacyPolicy ;
        iot-priv:accessPurposeAllowed
                :HashTags_Recommend ;
        iot-priv:hasDataCollectionMethod
                :Location_Collection ;
        iot-priv:privateDataCanBeUpdated
                "true"^^xsd:boolean .

:Stadium
        a       iot-lite:Coverage , owl:NamedIndividual ;
        iot-priv:hasWKTPolygon
                "POLYGON((30.34 10.14, 40.23 40.12, 20.75 40.123, 10.123 20.63))"^^
                    wkt:wktLiteral .

:Request_001
        a       owl:NamedIndividual , iot-priv:ConsumptionRequest ;
```

```
38          iot-priv:hasOptIn :Alexandre_PrivacyPolicy ;
39          iot-priv:requestInstant
40                  "2018-02-18T10:18:16"^^xsd:dateTime ;
41          iot-priv:requestLocation
42                  :Location_001 ;
43          iot-priv:requestPurpose
44                  :HashTags_Recommend .
45
46  :MyPhotos_Alexandre_Service
47          a       iot-lite:Service , owl:NamedIndividual ;
48          iot-priv:hasServiceDescription
49                  "Service who recommends hashtags touser according to his context"^^
                        xsd:string ;
50          iot-priv:isPersonalyIdentifiable
51                  "false"^^xsd:boolean ;
52          iot-lite:endpoint "www.myphotos.net/service"^^xsd:anyURI ;
53          iot-lite:interfaceDescription
54                  "www.myphotos.net/interface"^^xsd:anyURI ;
55          iot-lite:interfaceType
56                  "RestFULL"^^xsd:string .
57
58  :Location_001
59          a       owl:NamedIndividual , geo:Point ;
60          iot-priv:hasWKTPoint
61                  "POINT(30.34 10.14)"^^wkt:wktLiteral .
62
63  :MyPhotos_Alexandre
64          a       owl:NamedIndividual , ssn:System ;
65          iot-priv:hasDataCollectionMethod
66                  :Location_Collection ;
67          iot-lite:exposedBy :MyPhotos_Alexandre_Service .
68
69  :Alexandre_PrivacyPolicy
70          a       iot-priv:PrivacyPolicy , owl:NamedIndividual ;
71          iot-priv:hasGeneralPrivacyPolicy
72                  :MyPhotos_GeneralPrivacy ;
73          iot-priv:protects :MyPhotos_Alexandre_Service .
74
75  :Shopping_Location
76          a       owl:NamedIndividual , geo:Point ;
77          iot-priv:hasWKTPoint
78                  "POINT(120.5 -159.16)"^^wkt:wktLiteral .
79
80  :MyPhotos
81          a       owl:NamedIndividual , ssn:System ;
82          ssn:hasSubSystem :MyPhotos_Alexandre .
83
84  :Location_Collection
85          a       owl:NamedIndividual , iot-priv:DataCollection ;
86          iot-priv:collectionMethodDescription
87                  "Collect GPS Data"^^xsd:string ;
88          iot-priv:collectsFrom
89                  :Stadium ;
90          iot-priv:dataUsageDescription
91                  "The Data Collected is used torecommend hashtags to users"^^xsd:
                        string .
92
93  :HashTags_Recommend
```

```
94          a          owl:NamedIndividual , iot-priv:AccessPurpose .
95
96  :Alexandrew
97          a          iot-priv:DataOwner , iot-priv:Recipient , owl:NamedIndividual ;
98          iot-priv:hasPrivacyPolicy
99                     :Alexandre_PrivacyPolicy ;
100         iot-priv:owns :MyPhotos_Alexandre .
```

Total de triplas: 56

A.2 Cenário de Assistência Médica Domiciliar

Código A.2 Modelagem do cenário de Assistência Médica Domiciliar serializada em Turtle.

```
1   @prefix geo:       <http://www.w3.org/2003/01/geo/wgs84_pos#> .
2   @prefix :          <http://example.org/instances/iot-priv#> .
3   @prefix wkt:       <http://www.opengis.net/ont/sf#> .
4   @prefix iot-lite:  <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .
5   @prefix owl:       <http://www.w3.org/2002/07/owl#> .
6   @prefix rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
7   @prefix xsd:       <http://www.w3.org/2001/XMLSchema#> .
8   @prefix rdfs:      <http://www.w3.org/2000/01/rdf-schema#> .
9   @prefix iot-priv:  <http://example.org/ontology/iot-priv#> .
10  @prefix iot-priv-m: <http://example.org/ontology/iot-priv/med#> .
11  @prefix ssn:       <http://purl.oclc.org/NET/ssnx/ssn#> .
12
13  :DeleteRate
14          a          iot-priv:TimeDuration , owl:NamedIndividual ;
15          iot-priv:hasDuration
16                     "10"^^xsd:long ;
17          iot-priv:hasDurationDescription
18                     "Years"^^xsd:string .
19
20  :AmericanDollar
21          a          owl:NamedIndividual , iot-priv-m:AmericanDollar .
22
23  :SmartHospital
24          a          iot-priv:DataOwner , owl:NamedIndividual ;
25          iot-priv:hasPrivacyPolicy
26                     :SmartHospitalPrivacyPolicy ;
27          iot-priv:owns :SmartHospitalManagement .
28
29  :MedicalAccessPurpose
30          a          owl:NamedIndividual , iot-priv:AccessPurpose .
31
32  :DDD_AccessPurpose
33          a          owl:NamedIndividual , iot-priv:AccessPurpose .
34
35  :Request0004_to_AndrewService
36          a          owl:NamedIndividual , iot-priv:ConsumptionRequest ;
37          iot-priv:consumes :AndrewMedicalRecordService ;
38          iot-priv:hasOptIn :AndrewPrivacyPolicy ;
39          iot-priv:requestInstant
```

```
40                  "2018-02-17T20:25:15"^^xsd:dateTime ;
41          iot-priv:requestLocation
42                  :StudentDDD_ConsumeLocation ;
43          iot-priv:requestPurpose
44                  :RoutineTreatment .
45
46  :RoutineTreatment
47          a       owl:NamedIndividual , iot-priv-m:RoutineTreatment .
48
49  :Andrew_DenyList
50          a       owl:NamedIndividual , iot-priv:DenyList ;
51          iot-priv:hasRecipient
52                  :Person_CCC .
53
54  :AndrewHouse
55          a       iot-lite:Rectangle , owl:NamedIndividual ;
56          iot-priv:hasWKTPolygon
57                  "POLYGON((30 10, 40 40, 20 40, 10 20, 30 10))"^^wkt:wktLiteral .
58
59  :StudentDDD_ConsumeLocation2
60          a       <http://www.opengis.net/ont/geosparql#SpatialObject> , owl:
                NamedIndividual , geo:Point ;
61          iot-priv:hasWKTPoint
62                  "POINT(30 10)"^^wkt:wktLiteral ;
63          iot-lite:altRelative
64                  "2st Floor"^^xsd:string ;
65          geo:lat "25"^^xsd:decimal ;
66          geo:long "25"^^xsd:decimal .
67
68  :RSACriptography
69          a       owl:NamedIndividual , iot-priv-m:RSA .
70
71  :AndrewGeneralPrivacyPolicy
72          a       owl:NamedIndividual , iot-priv:GeneralPrivacyPolicy ;
73          iot-priv:accessFrom :AndrewHouse ;
74          iot-priv:accessPurposeAllowed
75                  :EmergencyTreatment , :RoutineTreatment ;
76          iot-priv:discloses :SmartHospitalDisclose ;
77          iot-priv:hasDataCollectionMethod
78                  :SensorCollection ;
79          iot-priv:hasObligations
80                  :NotShare_Obligation ;
81          iot-priv:hasServiceValue
82                  :AndrewService_Value ;
83          iot-priv:maxRetentionPeriod
84                  :RetentionPeriod ;
85          iot-priv:privateDataCanBeUpdated
86                  "true"^^xsd:boolean ;
87          iot-priv:useAnonymizationTechnique
88                  :k-anonymiti_Technique .
89
90  :Request0002_to_AndrewService
91          a       owl:NamedIndividual , iot-priv:ConsumptionRequest ;
92          iot-priv:consumes :AndrewMedicalRecordService ;
93          iot-priv:hasOptIn :AndrewPrivacyPolicy ;
94          iot-priv:requestInstant
95                  "2018-02-17T20:25:15"^^xsd:dateTime ;
96          iot-priv:requestLocation
```

```
97                  :StudentDDD_ConsumeLocation2 ;
98          iot-priv:requestPurpose
99                  :RoutineTreatment .
100
101
102 :Request0004_to_AndrewService
103         a       owl:NamedIndividual , iot-priv:ConsumptionRequest ;
104         iot-priv:consumes :AndrewMedicalRecordService ;
105         iot-priv:hasOptIn :AndrewPrivacyPolicy ;
106         iot-priv:requestInstant
107                 "2018-02-17T20:25:15"^^xsd:dateTime ;
108         iot-priv:requestLocation
109                 :StudentDDD_ConsumeLocation2 ;
110         iot-priv:requestPurpose
111                 :RoutineTreatment .
112
113 :Nurse_BBB
114         a       iot-priv:Recipient , owl:NamedIndividual ;
115         iot-priv:hasConsumptionRequest
116                 :Request0002_to_AndrewService ;
117         iot-priv:hasOptIn :AndrewPrivacyPolicy .
118
119 :AndrewSmartHouse
120         a       owl:NamedIndividual , ssn:System ;
121         iot-priv:hasDataCollectionMethod
122                 :SensorCollection ;
123         iot-lite:exposedBy :AndrewMedicalRecordService ;
124         iot-lite:hasCoverage
125                 :AndrewHouse .
126
127 :SmartHospitalDisclose
128         a       owl:NamedIndividual , iot-priv:Disclosure ;
129         iot-priv:discloseFor
130                 :SmartHospitalManagement ;
131         iot-priv:disclosureDescription
132                 "Disclose personal informations to Management System" ;
133         iot-priv:disclosureRate
134                 :DisclosureRate .
135
136 :AndrewStoragePrivacyPolicy
137         a       iot-priv:StoragePrivacyPolicy , owl:NamedIndividual ;
138         iot-priv:criptographyOnStorage
139                 :RSACriptography .
140
141 :AndrewPrivacyPolicy
142         a       iot-priv:PrivacyPolicy , owl:NamedIndividual ;
143         iot-priv:hasGeneralPrivacyPolicy
144                 :AndrewGeneralPrivacyPolicy ;
145         iot-priv:hasStoragePrivacyPolicy
146                 :AndrewStoragePrivacyPolicy ;
147         iot-priv:protects :AndrewMedicalRecordService .
148
149 :SmartHospitalManagement
150         a       owl:NamedIndividual , ssn:System ;
151         iot-lite:exposedBy :SmartHospitalManagementService ;
152         ssn:hasSubSystem :AndrewSmartHouse .
153
154 :UserAndrew
```

```
155         a         iot-priv:DataOwner , owl:NamedIndividual ;
156         iot-priv:hasPrivacyPolicy
157                   :AndrewPrivacyPolicy ;
158         iot-priv:owns :AndrewSmartHouse .
159
160 :Andrew_GrantList
161         a         owl:NamedIndividual , iot-priv:GrantList ;
162         iot-priv:hasRecipient
163                   :Nurse_BBB , :Doctor_AAA .
164
165 :DisclosureRate
166         a         iot-priv:TimeDuration , owl:NamedIndividual ;
167         iot-priv:hasDuration
168                   "1"^^xsd:long ;
169         iot-priv:hasDurationDescription
170                   "months"^^xsd:string .
171
172 :AndrewService_Value
173         a         owl:NamedIndividual , iot-priv:ServiceValue ;
174         iot-priv:hasCurrency
175                   :AmericanDollar ;
176         iot-priv:hasValue "10"^^xsd:decimal .
177
178 :SmartHospitalPrivacyPolicy
179         a         iot-priv:PrivacyPolicy , owl:NamedIndividual ;
180         iot-priv:hasGeneralPrivacyPolicy
181                   :SmartHospitalGeneralPrivacyPolicy ;
182         iot-priv:hasStoragePrivacyPolicy
183                   :SmartHospitalStoragePrivacyPolicy ;
184         iot-priv:protects :SmartHospitalManagementService .
185
186 :NotShare_Obligation
187         a         iot-priv-m:NotShareObligations , owl:NamedIndividual .
188
189 :StudentDDD_ConsumeLocation
190         a         <http://www.opengis.net/ont/geosparql#SpatialObject> , owl:
              NamedIndividual , geo:Point ;
191         iot-priv:hasWKTPoint
192                   "POINT(-16.67363 -49.24468)"^^wkt:wktLiteral ;
193         iot-lite:altRelative
194                   "2st Floor"^^xsd:string ;
195         geo:lat "25"^^xsd:decimal ;
196         geo:long "25"^^xsd:decimal .
197
198 :RetentionPeriod
199         a         iot-priv:TimeDuration , owl:NamedIndividual ;
200         iot-priv:hasDuration
201                   "7"^^xsd:long ;
202         iot-priv:hasDurationDescription
203                   "days"^^xsd:string .
204
205 :NurseAccessPurpose
206         a         owl:NamedIndividual , iot-priv-m:NurseAccessPurpose .
207
208 :Doctor_AAA
209         a         iot-priv:Recipient , owl:NamedIndividual ;
210         iot-priv:hasConsumptionRequest
211                   :Request0001_to_AndrewService ;
```

```
212         iot-priv:hasOptIn :AndrewPrivacyPolicy .
213
214
215 :Mayke
216         a       iot-priv:Recipient , owl:NamedIndividual ;
217         iot-priv:hasConsumptionRequest
218                 :Request0004_to_AndrewService ;
219         iot-priv:hasOptIn :AndrewPrivacyPolicy .
220
221 :SmartHospitalGeneralPrivacyPolicy
222         a       owl:NamedIndividual , iot-priv:GeneralPrivacyPolicy ;
223         iot-priv:accessPurposeAllowed
224                 :MedicalAccessPurpose , :ManagementPurpose , :NurseAccessPurpose ;
225         iot-priv:privateDataCanBeUpdated
226                 "true"^^xsd:boolean ;
227         iot-priv:requestConsumerLog
228                 "false"^^xsd:boolean ;
229         iot-priv:useAnonymizationTechnique
230                 :k-anonymiti_Technique .
231
232 :k-anonymiti_Technique
233         a       iot-priv-m:k-anonymity_Technique , owl:NamedIndividual .
234
235 :ManagementPurpose
236         a       owl:NamedIndividual , iot-priv-m:ManagementPurpose .
237
238 :EmergencyTreatment
239         a       owl:NamedIndividual , iot-priv-m:EmergencyTreatment .
240
241 :SmartHospitalStoragePrivacyPolicy
242         a       iot-priv:StoragePrivacyPolicy , owl:NamedIndividual ;
243         iot-priv:criptographyOnStorage
244                 :RSACriptography ;
245         iot-priv:deleteDataAfter
246                 :DeleteRate .
247
248 :Person_CCC
249         a       iot-priv:Recipient , owl:NamedIndividual ;
250         iot-priv:hasConsumptionRequest
251                 :Request0003_to_AndrewService .
252
253 :SensorCollection
254         a       owl:NamedIndividual , iot-priv:DataCollection ;
255         iot-priv:collectionMethodDescription
256                 "Collect data from body sensors"^^xsd:string ;
257         iot-priv:dataUsageDescription
258                 "The data collected is analyzed by doctors and nurses authorized"^^
                        xsd:string .
259
260 :Request0001_to_AndrewService
261         a       owl:NamedIndividual , iot-priv:ConsumptionRequest ;
262         iot-priv:consumes :AndrewMedicalRecordService ;
263         iot-priv:hasOptIn :AndrewPrivacyPolicy ;
264         iot-priv:requestInstant
265                 "2018-11-17T20:25:15"^^xsd:dateTime ;
266         iot-priv:requestLocation
267                 :StudentDDD_ConsumeLocation ;
268         iot-priv:requestPurpose
```

```
269                 :RoutineTreatment .
270
271 :AndrewMedicalRecordService
272         a       iot-lite:Service , owl:NamedIndividual ;
273         iot-priv:hasAccessList
274                 :Andrew_GrantList , :Andrew_DenyList ;
275         iot-priv:hasServiceDescription
276                 "Service that returns Andrew's medical records."^^xsd:string ;
277         iot-priv:isPersonalyIdentifiable
278                 "true"^^xsd:boolean ;
279         iot-lite:endpoint "www.smarthospital.net/medicalrecords/andrew/service"^^xsd:
                anyURI ;
280         iot-lite:interfaceDescription
281                 "www.smarthospital.net/medicalrecords/andrew/interface"^^xsd:anyURI ;
282         iot-lite:interfaceType
283                 "RestFULL"^^xsd:string .
284
285 :Student_DDD
286         a       <http://www.w3.org/2006/vcard/ns#Individual> , iot-priv:Recipient ,
                owl:NamedIndividual ;
287         iot-priv:hasConsumptionRequest
288                 :Request0001_to_AndrewService ;
289         <http://www.w3.org/2006/vcard/ns#bday>
290                 "1992"^^xsd:gYear ;
291         <http://www.w3.org/2006/vcard/ns#country-name>
292                 "Brasil"^^xsd:string ;
293         <http://www.w3.org/2006/vcard/ns#given-name>
294                 "Mayke Ferreira"^^xsd:string ;
295         <http://www.w3.org/2006/vcard/ns#nickname>
296                 "MaykeFA"^^xsd:string .
297
298 :SmartHospitalManagementService
299         a       iot-lite:Service , owl:NamedIndividual ;
300         iot-priv:hasServiceDescription
301                 "Service to manage the Smart Hospital. It's acessible by Doctors,
                        Nurses and Hospital Managers."^^xsd:string ;
302         iot-priv:isPersonalyIdentifiable
303                 "true"^^xsd:boolean ;
304         iot-lite:endpoint "www.smarthospital.net/service"^^xsd:anyURI ;
305         iot-lite:interfaceDescription
306                 "www.smarthospital.net/interface"^^xsd:anyURI ;
307         iot-lite:interfaceType
308                 "RestFUL"^^xsd:string .
```

Total de triplas: 179

APÊNDICE B

IoT-PrivM – Uma extensão da Ontologia *IoT-Priv* para o Domínio de Assistência Médica Domiciliar

Este apêndice demonstra como é realizada a extensão da *IoT-Priv*, seguindo uma abordagem tradicional de duas camadas, que consiste em importar a ontologia *IoT-Priv* (linha 11) e estender a semântica dos conceitos genéricos, tais como *AccessPurpose*, *CriptographyTechniques*, *Obligations*, Currency, entre outros. Neste caso, a *IoT-Priv* foi estendida para se adequar aos propósitos e necessidades de um domínio específico, no caso, assistência médica domiciliar.

Código B.1 *IoT-PrivM* – Uma extensão da Ontologia *IoT-Priv* para o Domínio de Assistência Médica Domiciliar

```
1  @prefix : <http://example.org/ontology/iot-priv/med#> .
2  @prefix owl: <http://www.w3.org/2002/07/owl#> .
3  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4  @prefix xml: <http://www.w3.org/XML/1998/namespace> .
5  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
6  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
7  @prefix iot-priv: <http://example.org/ontology/iot-priv#> .
8  @base <http://example.org/ontology/iot-priv/med> .
9
10 <http://example.org/ontology/iot-priv/med> rdf:type owl:Ontology ;
11   owl:imports <http://example.org/ontology/iot-priv> ;
12   rdfs:comment "Especializacao da Ontologia IoT-Priv para o dominio de assistencia
         medica domiciliar" .
13
14 ###################################################################
15 #    Classes
16 ###################################################################
17
18 ###  http://example.org/ontology/iot-priv/med#AmericaCurrency
19 :AmericaCurrency rdf:type owl:Class ;
20                   rdfs:subClassOf iot-priv:Currency .
21
22 ###  http://example.org/ontology/iot-priv/med#AmericanDollar
23 :AmericanDollar rdf:type owl:Class ;
24                  owl:equivalentClass :Dollar ;
```

```
25                  rdfs:subClassOf :UnitedStatesCurrency .
26
27  ###  http://example.org/ontology/iot-priv/med#AsymmetricKey
28  :AsymmetricKey rdf:type owl:Class ;
29                 rdfs:subClassOf iot-priv:CriptographyTechniques .
30
31  ###  http://example.org/ontology/iot-priv/med#BrazilCurrency
32  :BrazilCurrency rdf:type owl:Class ;
33                  rdfs:subClassOf :AmericaCurrency .
34
35  ###  http://example.org/ontology/iot-priv/med#BrazilianReal
36  :BrazilianReal rdf:type owl:Class ;
37                 owl:equivalentClass :Real ;
38                 rdfs:subClassOf :BrazilCurrency .
39
40  ###  http://example.org/ontology/iot-priv/med#Conciliation
41  :Conciliation rdf:type owl:Class ;
42                rdfs:subClassOf :Law .
43
44  ###  http://example.org/ontology/iot-priv/med#DataAccessPurpose
45  :DataAccessPurpose rdf:type owl:Class ;
46                     rdfs:subClassOf :MedicalPurpose .
47
48  ###  http://example.org/ontology/iot-priv/med#DenyAccessList
49  :DenyAccessList rdf:type owl:Class ;
50              rdfs:subClassOf iot-priv:ResolutionType .
51
52  ###  http://example.org/ontology/iot-priv/med#Dollar
53  :Dollar rdf:type owl:Class ;
54          rdfs:subClassOf :UnitedStatesCurrency .
55
56  ###  http://example.org/ontology/iot-priv/med#EmergencyTreatment
57  :EmergencyTreatment rdf:type owl:Class ;
58                      rdfs:subClassOf :TreatmentPurpose .
59
60  ###  http://example.org/ontology/iot-priv/med#Law
61  :Law rdf:type owl:Class ;
62       rdfs:subClassOf iot-priv:ResolutionType .
63
64  ###  http://example.org/ontology/iot-priv/med#Lawsuits
65  :Lawsuits rdf:type owl:Class ;
66            rdfs:subClassOf :Law .
67
68  ###  http://example.org/ontology/iot-priv/med#ManagementPurpose
69  :ManagementPurpose rdf:type owl:Class ;
70                     rdfs:subClassOf :MedicalPurpose .
71
72  ###  http://example.org/ontology/iot-priv/med#MedicalAccessPurpose
73  :MedicalAccessPurpose rdf:type owl:Class ;
74                        rdfs:subClassOf :DataAccessPurpose .
75
76  ###  http://example.org/ontology/iot-priv/med#MedicalPurpose
77  :MedicalPurpose rdf:type owl:Class ;
78                  rdfs:subClassOf iot-priv:AccessPurpose .
79
80  ###  http://example.org/ontology/iot-priv/med#NotShareObligation
81  :NotShareObligation rdf:type owl:Class ;
82                      rdfs:subClassOf iot-priv:Obligations .
```

```
83
84  ###  http://example.org/ontology/iot-priv/med#NurseAccessPurpose
85  :NurseAccessPurpose rdf:type owl:Class ;
86                      rdfs:subClassOf :DataAccessPurpose .
87
88  ###  http://example.org/ontology/iot-priv/med#Obfuscation
89  :Obfuscation rdf:type owl:Class ;
90               rdfs:subClassOf iot-priv:AnonymizationTechniques .
91
92  ###  http://example.org/ontology/iot-priv/med#RSA
93  :RSA rdf:type owl:Class ;
94       rdfs:subClassOf :AsymmetricKey .
95
96  ###  http://example.org/ontology/iot-priv/med#Real
97  :Real rdf:type owl:Class ;
98        rdfs:subClassOf :BrazilCurrency .
99
100 ###  http://example.org/ontology/iot-priv/med#RoutineTreatment
101 :RoutineTreatment rdf:type owl:Class ;
102                   rdfs:subClassOf :TreatmentPurpose .
103
104 ###  http://example.org/ontology/iot-priv/med#SymmetricKey
105 :SymmetricKey rdf:type owl:Class ;
106               rdfs:subClassOf iot-priv:CriptographyTechniques .
107
108 ###  http://example.org/ontology/iot-priv/med#TreatmentPurpose
109 :TreatmentPurpose rdf:type owl:Class ;
110                   rdfs:subClassOf :MedicalPurpose .
111
112 ###  http://example.org/ontology/iot-priv/med#UnitedStatesCurrency
113 :UnitedStatesCurrency rdf:type owl:Class ;
114                       rdfs:subClassOf :AmericaCurrency .
115
116 ###  http://example.org/ontology/iot-priv/med#WithoutObligation
117 :WithoutObligation rdf:type owl:Class ;
118                    rdfs:subClassOf iot-priv:Obligations .
119
120 ###  http://example.org/ontology/iot-priv/med#k-anonimity_Technique
121 :k-anonimity_Technique rdf:type owl:Class ;
122                        rdfs:subClassOf :Obfuscation .
```