UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SMART E-PING: *FRAMEWORK* DE INTEROPERABILIDADE DA ARQUITETURA E-PING COM A PLATAFORMA FIWARE PARA O USO EM CIDADES INTELIGENTES

Dissertação de Mestrado

Thauane Moura Garcia

São Cristóvão – Sergipe

2019

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Thauane Moura Garcia

SMART E-PING: *FRAMEWORK* DE INTEROPERABILIDADE DA ARQUITETURA E-PING COM A PLATAFORMA FIWARE PARA O USO EM CIDADES INTELIGENTES

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito final para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Prof. Dr. Rogério Patrício Chagas do Nascimento
Coorientador(a): Prof. Dr. Michel dos Santos Soares

São Cristóvão – Sergipe

2019

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Garcia, Thauane Moura
M929s Smart e-ping: framework de interoperabilidade da arquitetura e-ping com a plataforma fiware para o uso em cidades inteligentes / Thauane Moura Garcia ; orientador Rogério Patrício Chagas do Nascimento . - São Cristóvão, 2019.
127 f.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2019.

1. Computação. 2. Framework (Programa de computador). 3. Software – Governo. 4. Middleware. 5. Informação governamental eletrônica. I. Nascimento, Rogério Patrício Chagas do orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: THAUANE MOURA GARCIA

Em 29 dias do mês de maio do ano de dois mil e nove, com início às 9h00min, realizou-se na sala de seminário do Dcomp da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **THAUANE MOURA GARCIA**, que desenvolveu o trabalho intitulado: **"SMART E-PING: FRAMEWORK DE INTEROPERABILIDADE DA ARQUITETURA E-PING COM A PLATAFORMA FIWARE PARA O USO EM CIDADES INTELIGENTES"**, sob a orientação do Prof. Dr. ROGERIO PATRICIO CHAGAS DO NASCIMENTO. A Sessão foi presidida pelo Prof. Dr. ROGERIO PATRICIO CHAGAS DO NASCIMENTO (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. MICHEL DOS SANTOS SOARES (PROCC/UFS) e, em seguida, ao Prof. EDUARDO JAMES PEREIRA SOUTO(UFAM). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) APROVADO "*(aprovado/reprovado)*". Atendidas as exigências da Instrução Normativa 01/2017/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 25/2014/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos*", 29 de maio de 2019.

Prof. Dr. ROGERIO PATRICIO CHAGAS DO NASCIMENTO
(PROCC/UFS)
Presidente

Prof. Dr. MICHEL DOS SANTOS SOARES
(PROCC/UFS)
Examinador Interno

Prof. Dr. EDUARDO JAMES PEREIRA SOUTO
(UFAM)
Examinador Externo

THAUANE MOURA GARCIA
Candidato

Agradecimentos

Agradeço primeiramente a Deus, por estar sempre comigo, me guiando, mostrando sempre o melhor para mim. Obrigada Deus por nunca ter me desamparado nos momentos difíceis que passei durante o mestrado. Só o senhor sabe, o quanto lutei, sofri, sorri e dei o meu melhor. Obrigada Deus, por ter traçado o meu caminho e feito a minha escolha pela área de computação. Obrigada também por ter me dado forças e sabedoria para cumprir com mais uma etapa na minha vida. Que venham as próximas.

Aos meus pais, pessoas fundamentais e essenciais da minha vida, por terem me dado educação, por me ensinarem sobre a vida e terem me dado amor e carinho. Obrigada por acreditarem que sou capaz de ir além. Ao meu irmão, a minha cunhada Joyce e a minha sobrinha Felícia obrigada pelo apoio e pela convivência. Amo vocês.

Aos meus avós, que sempre estiveram presentes na minha vida em todos os meus momentos me dando forças nessa grande trajetória.

Agradeço também a toda a minha família, tia(o)s, primo(a)s, pelo apoio que me deram e sempre me dão. Obrigada por me ajudarem demais nessa trajetória. Aos demais familiares, agradeço de coração. Aos meus amigos que conquistei antes e durante o mestrado na UFS: X Mestrado (Marianne, Davy, Cícero, Erick, Denise, Breno), Alef, Faustino, Hugo, Ademir, Bruno, Weslan, Daniela, Verônica, Augusto, Leonardo, Leo Macedo, Rose, George, Elder, Elayne, Elaine, Erickson, Sara, Luan, Aninha, Dina e dentre outros que me ajudaram e apoiaram direta ou indiretamente. Obrigada pelo carinho de todos e muito obrigada pelas ajudas e apoios.

Ao meu namorado, Maicson, por toda paciência, compreensão, carinho e amor. Obrigada por me ajudar muitas vezes a solucionar problemas quando estavam difíceis de se resolver. Além deste trabalho mestrado, dedico todo meu amor a você. À família Silva, pelo acolhimento e pelo carinho. Só tenho a agradecer por todo o carinho que cada um tem por mim. Obrigada pelo apoio, por tudo o que fazem por mim. Obrigada por me apoiarem em momento difíceis, obrigada por me deixarem fazer parte dessa família. Obrigada minha segunda família.

Agradeço a todos os professores por terem me proporcionado o conhecimento e aprendizado. Em especial, ao meu orientador Rogério e meu coorientador Michel, por terem transmitido seus conhecimentos, experiências, dentre outros. Serei eternamente grata por tudo. Não poderia deixar de agradecer ao meu coorientador não oficial Gilton, vulgo Gilton Mal. Obrigada por me aturar, pelos meus enjoos diários, por ter me acolhido e coorientado por fora.

Obrigada também ao Weslan, que me ajudou bastante durante o meu mestrado. Obrigada Weslan, não tenho como agradecer pela grande ajuda. Por fim, a todos que direta ou indiretamente fizeram parte do meu mestrado, o meu muito obrigada.

Resumo

A interoperabilidade possui uma função importante para o Governo Eletrônico Brasileiro (GEB), provendo uma única interface com todas as informações armazenadas em diferentes servidores na rede. Dessa maneira, o GEB utiliza um padrão de interoperabilidade, chamado e-PING, para auxiliar na integração de sistemas e compartilhamento das informações. No entanto, a interoperabilidade ainda é um fator desafiante para inúmeras organizações devido à grande quantidade e diversidade de dispositivos que estão predispostos a trocar informações entre si. Para integrar os inúmeros sistemas, aplicativos e/ou dispositivos com diferentes plataformas disponíveis no mundo, pode-se utilizar um *middleware*. Uma plataforma *middleware* também pode auxiliar as Cidades Inteligentes (CI) fornecendo acessos padronizados aos dados e serviços provenientes das interfaces dos pilares de uma CI, além de promover o reuso de serviços genéricos. A plataforma FIWARE foi identificada como a mais adequada para aprimorar o GEB, pois trata-se de uma das plataformas mais utilizadas em trabalhos científicos. Diante deste cenário, este trabalho teve como principal objetivo desenvolver o *Framework Smart* e-PING que pretende possibilitar a interoperabilidade entre *Softwares* Públicos, que utilizam a arquitetura e-PING, e a plataforma FIWARE. A fim de contemplar o objetivo proposto, foi adotado um estudo exploratório por meio de procedimentos bibliográficos e, por fim, foi realizado um estudo de caso para a validação do *Framework* proposto. O estudo de caso foi realizado em um cenário real que teve como objetivo analisar o *Framework Smart* e-PING, avaliando a interoperabilidade entre um *Software* Público e a plataforma FIWARE. Dessa forma, foi validado o *Framework Smart* e-PING com base em três questões de validações, simulando 35 testes. A primeira questão avaliou a frequência com que o *Framework Smart* e-PING traduzia corretamente o arquivo do Banco de Dados do *Software* Público para a plataforma FIWARE. Para essa questão, o *Framework Smart* e-PING obteve uma taxa de sucesso de 97,14%. A segunda questão avaliou a frequência com que o *Framework Smart* e-PING causava perda de informação na tradução do arquivo do Banco de Dados do *Software* Público para a plataforma FIWARE. Para a segunda questão, o *Framework Smart* e-PING obteve uma taxa de erro de 2,85%. Por fim, a última questão avaliou o tempo estimado com que o *Framework Smart* e-PING permitiria a interoperabilidade entre o *Software* Público e a plataforma FIWARE. Para essa questão de pesquisa, o *Framework Smart* e-PING teve um tempo médio para cada iteração de 248 milissegundos. Com base nesse estudo de caso, inferiu-se que o *Framework Smart* e-PING atende parcialmente o objetivo proposto neste trabalho. Portanto, o *Framework Smart* e-PING permite a integração e facilita o uso da plataforma aberta FIWARE para as organizações que já utilizam a arquitetura e-PING. Desta forma, esse trabalho dá suporte ao GEB quanto a redução dos seus custos, otimização e melhoria dos seus serviços públicos por meio de soluções inteligentes e a reutilização de recursos inteligentes.

Palavras-chave: e-PING. FIWARE. *Framework Smart* e-PING. Governo Eletrônico.

Abstract

Interoperability has an important function for the Brazilian Electronic Government (GEB), providing a single interface with all information stored on different servers in the network. In this way, the GEB uses an interoperability standard, called e-PING, to aid in the integration of systems and information sharing. However, interoperability is still a challenging factor for many organizations because of the large number and diversity of devices that are predisposed to exchange information with each other. To integrate the numerous systems, applications and/or devices with different platforms available in the world, one can use a middleware. A middleware platform can also assist Smarts Cities (CI) by providing standardized access to data and services from the interfaces of the pillars of an CI, as well as promoting the reuse of generic services. The FIWARE platform was identified as the most suitable to improve the GEB, since it is one of the platforms most used in scientific work. In this scenario, the main objective of this work was to develop the Smart e-PING Framework, which intends to enable interoperability between Public Softwares, which use the e-PING architecture, and the FIWARE platform. In order to contemplate the proposed objective, an exploratory study was adopted through bibliographic procedures and, finally, a case study was carried out to validate the proposed Framework. The case study was carried out in a real scenario that aimed to analyze the Smart Framework e-PING, evaluating the interoperability between a Public Software and the FIWARE platform. Thus, the Smart e-PING Framework was validated based on three validation questions, simulating 35 tests. The first question evaluated the frequency with which the Smart e-PING Framework correctly translated the Public Software Database file into the FIWARE platform. For this issue, the Smart Framework e-PING achieved a success rate of 97.14%. The second question evaluated the frequency with which the Smart e-PING Framework caused loss of information in the translation of the Public Software Database file to the FIWARE platform. For the second question, the Smart Framework e-PING obtained an error rate of 2.85%. Finally, the last question evaluated the estimated time frame with which the Smart e-PING Framework would allow interoperability between the Public Software and the FIWARE platform. For this research question, the Smart Framework e-PING had an average time for each iteration of 248 milliseconds. Based on this case study, it was inferred that the Smart e-PING Framework partially meets the objective proposed in this work. Therefore, the Smart e-PING Framework enables integration and facilitates the use of the FIWARE open platform for organizations that already use the e-PING architecture. In this way, this work supports GEB in reducing costs, optimizing and improving its public services through intelligent solutions and the re-use of intelligent resources.

Keywords: e-PING. FIWARE. Smart e-PING Framework. Government Interoperability.

Lista de ilustrações

Lista de tabelas

Lista de códigos

Lista de abreviaturas e siglas

API	*Application Programming Interface*
ASPIRE	*Advanced Sensors and lightweight Programmable middleware for Innovative Rfid Enterprise applications*
BSC	*Balanced Score Card*
CAPES	Coordenação de Aperfeiçoamento Pessoal de Nível Superior
CI	Cidades Inteligentes
CKAN	*Comprehensive Knowledge Archive Network*
CRUD	*Create,Read,Update,Delete*
CSV	*Comma-Separated Values*
DPWS	*Devices Profilefor Web Services*
EBBITS	*Enabling Bussiness-Based Internet of Things and Services*
e-GIF	*Government Interoperability Framework*
e-Gov	Governo Eletrônico
EIF	*European Interoperability Framework*
EPC	*Eletronic Product Code*
e-PING	Padrão de Interoperabilidade do Governo Brasileiro
ePMG	Padrão de Metadados do Governo Brasileiro
FIFO	*First-In, First-Out*
FIOCRUZ/BA	Fundação Oswaldo Cruz da Bahia
FIWARE	*Future Internet Ware*
FUNDACENTRO	Fundação Jorge Duprat Figueiredo de Segurança e Medicina do Trabalho
GE	*Generic Enabler*
GEB	Governo Eletrônico Brasileiro

GQM	*Goal/Question/Metric*
GSN	*Global Sensor Network*
HTTP	*Hypertext Transfer Protocol*
I2ND	*Interface to Networks and Devices Architecture*
IBGE	Instituto Brasileiro de Geografia e Estatística
ID	IDentificador
ICEIS	*International Conference on Enterprise Information Systems*
IDE	*Integrated Development Environment*
IFBA	Instituto Federal da Bahia
IFBAIANO	Instituto Federal Baiano
IoT	*Internet of Things*
IPv4	*Internet Protocol* versão 4
JADE	*Java Agent Develpment Framework*
JSON	*JavaScript Object Notation*
JVM	*Java Virtual Machine*
MNI	*Brazilian e-Justice Interoperability Model*
MPOG	Ministério do Planejamento, Orçamento e Gestão
MQTT	*Message Queue Telemetry Transport*
MTC	*Many-Task Computing*
NFC	*Near Field Communication*
NGSI	*Next Generation Service Interface*
NGSIv1	*Next Generation Service Interface* versão 1
NGSIv2	*Next Generation Service Interface* versão 2
OMA	*Open Mobile Alliance*
OpenHAB	*Open Home Automation Bus*
OSGi	*Open Service Gateway Initiative*

P2P	*Peer-to-peer*
PSPB	Portal do *Software* Público Brasileiro
REST	*Representational State Transfer*
RFID	*Radio-Frequency IDentification*
RPC	*Remote Procedure Call*
S^3OIA	*Smart Spaces and Smart Objects Interoperability Architecture* I
SP	*Software* Público
SDK	*Software Development Kit*
SIRENA	*Service Infrastructure for Real-time Embedded Networked Applications*
SM4All	*Smart Home For All*
SOA	*Service-Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SOCRADES	*Service-Oriented Cross-Layer Infrastructure for Distribued smart Embedded devices*
SOFIA	*Smart Objects for Intelligent Applications*
SQL	*Structured Query Language*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
TIC	Tecnologia da Informação e Comunicação
UDP	*User Datagram Protocol*
UFS	Universidade Federal de Sergipe
UML	*Unified Modeling Language*
UPnP	*Universal Plug and Play*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
Wi-Fi	*Wireless Fidelity*
XML	*Extensible Markup Language*

Sumário

1 Introdução

Com a crescente importância das Tecnologias de Informação e Comunicação (TIC) em uso, o governo brasileiro vem aprimorando seus serviços para a população. O Governo Eletrônico Brasileiro (GEB) utilizado pelo Governo Federal vem se tornando cada vez mais presente (ARAUJO; REINHARD, 2014; SILVEIRA; WAZLAWICK; ROVER, 2015), passando a ter serviços digitais prestados para essa modalidade de Governança. A importância do GEB ocorre na contribuição de implementações do Governo Eletrônico (e-Gov), que abrange a integração dos sistemas distribuídos para estabelecer uma infraestrutura que suporta melhorias nos serviços oferecidos aos cidadãos. Por fim, o GEB busca reduzir ou eliminar a necessidade de acessar órgãos do governo a fim de obter um único serviço, facilitando o acesso do cidadão aos serviços públicos (SANTOS; REINHARD, 2010).

Existe a necessidade dos sistemas utilizados pelo governo interoperarem entre si, de modo a permitir que os dados provenientes das aplicações disponíveis na rede possam ser utilizados em uma interface unificada que apresenta todos os recursos para o usuário final, trazendo como principais benefícios: (a) aperfeiçoamento da prestação de serviços ao cidadão; (b) possibilidade de melhor tomada de decisão dos gestores; (c) maior confiabilidade e eficiência nas informações; (d) melhor coordenação dos programas e serviços do Governo; (e) melhor transparência nas ações do governo; (f) rápida resolução dos problemas; (g) reuso de soluções; (h) compartilhamento das informações; e (i) atenuação de informações redundantes (SANTOS; REINHARD, 2010; PARDO; NAM; BURKE, 2012; MELLO; MESQUITA; VIEIRA, 2015; MONDORF; WIMMER, 2017). Dessa maneira, o GEB criou o padrão de interoperabilidade e-PING, para auxiliar na integração de sistemas, no compartilhamento das informações e nos serviços do seu Governo.

Para lidar com o crescimento populacional, o Governo Federal vem empregando melhorias em seus serviços e infraestrutura das cidades, facilitando o desenvolvimento sustentável destas cidades, trazendo melhor qualidade de vida (BĂTĂGAN, 2011). Por outro lado, o Bra-

sil está no processo de construção de Cidades Inteligentes (CI), obtendo maior eficiência nas atividades que envolvem a gestão das cidades (WEISS; BERNARDES; CONSONI, 2017).

Uma plataforma *middleware* pode auxiliar as CI fornecendo acessos padronizados aos dados e serviços provenientes das interfaces das CI, além de proporcionar o reuso de serviços genéricos, facilitando no desenvolvimento de novas aplicações (WEISS; BERNARDES; CONSONI, 2017). Desta forma, no paradigma *Internet of Things* (IoT) as CI representam grande parcela desse modelo (SCHLEICHER et al., 2016). Ao desenvolver uma plataforma *middleware* específica para CI é possível obter os mesmos requisitos para IoT além dos requisitos específicos para CI (WEISS; BERNARDES; CONSONI, 2017). Os seguintes requisitos foram detectados para plataformas *middlewares* voltadas para IoT, que são interoperabilidade, descoberta e gerenciamento de dispositivos, segurança e privacidade, conectividade, adaptação dinâmica, ciência em contexto, dentre outros (KRCO; POKRIC; CARREZ, 2014; CAVALCANTE et al., 2015; RAZZAQUE et al., 2016; PFLANZNER; KERTÉSZ, 2016).

A FIWARE como uma plataforma *middleware* de IoT auxilia as CI oferecendo um conjunto de *Applications Programming Interface* (APIs) abertas para desenvolver novas aplicações e serviços para os setores tais como agricultura, energia, manufatura, dentre outros. A fim de facilitar o desenvolvimento nas aplicações e/ou serviços nas CI, as organizações e/ou empresas podem realizar testes em suas soluções desenvolvidas no ambiente da FIWARE (BROGAN; THUEMMLER, 2014).

A FIWARE é a mais utilizada e mais citada em trabalhos científicos (MARTINEZ et al., 2017). Outra grande vantagem é que os desenvolvedores não terão que aprender a complexidade e fragmentação das tecnologias IoT dessa plataforma aberta. Logo, só precisarão entender o protocolo *Next Generation Service Interface* (NGSI), que é o mais utilizado para representar todas as informações. A FIWARE possui uma documentação completa de seus componentes e de sua arquitetura. Como ela trabalha com *Generic Enablers* (GE) e *Application Programming Interface* (API) torna-se mais fácil para os desenvolvedores que utilizam essa plataforma *open source* para sua aplicação, pois é possível reutilizar ferramentas criadas previamente (KOVACS et al., 2016; MARTINEZ et al., 2017).

1.1 Problemática e Hipótese

Tendo em vista a importância do compartilhamento das informações no GEB e das dificuldades enfrentadas na integração dos diferentes dispositivos/aplicativos, facilitando o desenvolvimento de aplicações e permitindo a integração de dispositivos/sistemas (SANTOS; REINHARD, 2010; RAZZAQUE et al., 2016; LIU et al., 2016), há a necessidade de utilizar um elemento integrador como um *middleware* (RAZZAQUE et al., 2016). Outrossim, é necessário utilizar plataformas *middlewares* que permitam o uso de recursos de CI para facilitar a interoperabilidade entre dispositivos/sistemas, o que poderia fornecer a reutilização de serviços

inteligentes e tornar as comunicações mais homogêneas (BLAIR et al., 2011; SCHAFFERS et al., 2011).

Diante dessa situação, a solução aqui proposta a ser explorada é a possibilidade da integração de um padrão de interoperabilidade do GEB com uma plataforma FIWARE, permitindo a integração dos recursos e facilitando o uso da FIWARE para as organizações que utilizam a arquitetura e-PING. Desta forma, foi desenvolvido um *Framework* para realizar a interoperabilidade entre um *Software* Público que adere aos padrões da arquitetura e-PING e a plataforma FIWARE.

Assim a pergunta que permite o direcionamento da pesquisa é:

a) Como fornecer a interoperabilidade entre um *Software* Público baseado no padrão e-PING com a plataforma FIWARE?

Foram definidas as seguintes hipóteses:

a) **Hipótese nula H_0**: Utilizando o *Framework*, não será possível a interoperabilidade entre um *Software* Público que adere ao e-PING e a plataforma FIWARE.

b) **Hipótese alternativa H_1**: Utilizando o *Framework*, será possível a interoperabilidade entre um *Software* Público que adere ao e-PING e a plataforma FIWARE.

1.2 Justificativa

Devido ao crescimento populacional e problemas na infraestrutura das cidades, nas últimas décadas, há a necessidade de utilizar recursos para Cidades Inteligentes (CI), que está se tornando uma realidade em diferentes partes do mundo. Em especial no Brasil, o tema de CI está gradativamente presente nos seguintes setores, universidades, governos federais, empresas particulares, população, dentre outros (BOLÍVAR; LÓPEZ-QUILES, 2018).

As CI vêm aprimorando os seus serviços urbanos expandindo a qualidade de vida da população, trazendo maior eficiência. Uma plataforma de *software* é capaz de facilitar o auxílio da criação e integração de aplicações desenvolvidas para Cidades Inteligentes. As CI contêm doze pilares, que são Economia, Governança, Meio Ambiente, Modo de Vida, Educação, Saúde, Segurança, Fontes de Energia, Água, Mobilidade Urbana, Pessoas, Arquitetura e Urbanismo. Esses pilares favorecem propriamente as aplicações inteligentes (WEISS; BERNARDES; CONSONI, 2017).

Segundo Tanenbaum e Steen (2007), uma plataforma *middleware* é constituída em uma camada de *software*. Nesta camada encontra-se a infraestrutura e a aplicação, que oferece um acesso formal aos dados e serviços necessários por meio de interfaces de alto nível. Ademais, proporcionam o reuso desses serviços que colaboram para o desenvolvimento de aplicações

eficientes. Uma plataforma *middleware* pode fornecer requisitos desejados para as CI, como por exemplo a interoperabilidade, descoberta e gerenciamento de dispositivos, escalabilidade, adaptação dinâmica, segurança, gerenciamento de dados, ferramentas de desenvolvimento, e tratamento de volume de dados (KRCO; POKRIC; CARREZ, 2014; CAVALCANTE et al., 2015; RAZZAQUE et al., 2016; PFLANZNER; KERTÉSZ, 2016).

A necessidade em desenvolver um *Framework* para o Governo Eletrônico Brasileiro (GEB) consiste na possibilidade do uso da e-PING e das suas características/vantagens, no contexto de Cidades Inteligentes, por meio de uma plataforma. A criação de um *Framework* pode ainda otimizar a comunicação entre esses requisitos, definindo a construção de um *middleware* independente da tecnologia que está sendo utilizada. Ademais, poderá integrar esses recursos e facilitar o uso da plataforma para as organizações que utilizam a arquitetura e-PING. Por fim, os desenvolvedores poderão desenvolver aplicações/soluções, tendo em vista as facilidades propostas por um *Framework*. Os benefícios de uma plataforma já consolidada como a e-PING são: otimização dos serviços públicos por meio de soluções inteligentes, redução de custos, troca de dados, interoperabilidade, comunicação, transparência e a reutilização dos recursos inteligentes. A arquitetura e-PING tem sido utilizada por organizações brasileiras do Poder Executivo Federal, oferecendo interoperabilidade entre as plataformas e suas organizações, em virtude da utilização de padrões abertos (MACHADO; SANTOS; JUNIOR, 2013).

A importância de se conceber um *Framework* para as organizações brasileiras que estão utilizando a e-PING (MACHADO; SANTOS; JUNIOR, 2013; MELLO; MESQUITA; VIEIRA, 2015; BRASIL, 2016) ocorre possibilidade de interoperabilidade com a plataforma de CI FIWARE. A intenção de se criar um *Framework* é a de servir como intermediário, que irá tornar os dados padronizados, dispensando a necessidade de tratá-los na nova aplicação que está sendo desenvolvida.

Ainda que a e-PING não possua a integração com os recursos de CIs, torna-se onerosa a conectividade desses recursos com a e-PING sem a utilização de um *Framework* definido. Dessa forma, um *Framework* poderá permitir a interoperabilidade entre o *Software* Público que adere ao e-PING com a plataforma FIWARE.

1.3 Objetivos

Para a realização deste trabalho, os seguintes objetivos foram elencados.

O objetivo principal deste trabalho é desenvolver um *Framework*, aqui chamado de *Smart* e-PING, que permita a interoperabilidade dos *Softwares* Públicos que aderem ao uso da e-PING, suas características e vantagens, no contexto de Cidades Inteligentes, permitindo a interoperabilidade com a plataforma aberta FIWARE.

Além do objetivo geral, foram estabelecidos os seguintes objetivos específicos necessários

para atingir o objetivo proposto.

a) Investigar por meio de uma revisão sistemática a utilização de outras plataformas de IoT abertas voltadas para Cidades Inteligentes;

b) Desenvolver um *Framework* interoperável entre um *Software* Público e a plataforma FIWARE;

c) Validar a interoperabilidade do *Framework Smart* e-PING por meio da execução de um processo real de um *Software* Público que adere ao e-PING com a plataforma FIWARE.

1.4 Trabalhos Relacionados

Com o objetivo de identificar trabalhos semelhantes, foi realizada uma busca por artigos científicos nas bases de dados *Scopus*, *Springer*, *ScienceDirect*, *Web of Science*, *IEEE Xplore*, *ACM Digital Library* e BDBComp. Todas as bases, exceto a BDBComp, foram acessadas via portal da CAPES[1]. Também foram utilizados o repositório de teses e/ou dissertações da Universidade Federal de Minas Gerais, Universidade Federal de Pernambuco, Universidade de Brasília e Universidade de São Paulo. Para a seleção desses trabalhos, privilegiaram-se aqueles que continham relação com o tema, objetivos, a linha de pesquisa e com QUALIS[2] (somente para os artigos) ou com trabalhos relevantes.

Durante o processo de pesquisa, foi identificado que a e-PING vem oferecendo contribuições significativas com a prática de implantação e suas melhorias, da mesma maneira que os recursos de CI vêm fornecendo ferramentas para otimizar o compartilhamento, a localização e o uso dos dados. Existem trabalhos que vêm lidando especialmente com soluções que forneçam ao e-Gov a sua inserção no cenário de CI, por meio da utilização de *middleware* voltado para prover recursos advindos de funcionalidades já existentes. Os mais relevantes são apresentados a seguir:

a) Santos (2008): apresenta o desenvolvimento e implementação do padrão de interoperabilidade do Governo Eletrônico Brasileiro (e-PING). Identificou os benefícios, que são expansão do padrão de interoperabilidade; a necessidade da adoção dos padrões de interoperabilidade; melhoria na prestação de conta públicas; e auxílio na resolução dos problemas. Também destacou o incentivo do reuso dos sistemas, com propósito de reduzir os custos, aumentando a flexibilidade nas arquiteturas de TIC.

b) Santos e Reinhard (2012): o estudo de caso analisado por esses autores apresentam os resultados preliminares por meio do quadro da e-PING e baseiam-se em entrevistas e

[1] <https://www.periodicos.capes.gov.br>.

[2] QUALIS é usado pela CAPES para avaliar/classificar a qualidade dos artigos nos veículos dos programas de pós-graduação (CAPES, 2010).

análise de documentações. Outrossim, identificaram que a contribuição da adoção da e-PING também destaca os pontos que podem provocar a criação dos padrões.

c) Machado, Santos e Junior (2013): analisaram o nível de adoção da e-PING em cinco instituições públicas federais: Fundação Oswaldo Cruz da Bahia (FIOCRUZ/BA), Instituto Federal da Bahia (IFBA), Instituto Federal Baiano (IFBAIANO), Instituto Brasileiro de Geografia e Estatística (IBGE) e Fundação Jorge Duprat Figueiredo de Segurança e Medicina do Trabalho (FUNDACENTRO). Outrossim, identificaram um baixo nível das barreiras enfrentadas por cada instituição, que são: organizacionais, gerenciais e tecnológicas. Relataram a falta de apoio dos gestores para a adoção dos padrões de interoperabilidade no ambiente do e-Gov.

d) Silveira, Wazlawick e Rover (2015): propuseram um modelo brasileiro de interoperabilidade e-Justice (MNI, do inglês *Brazilian e-Justice Interoperability Model*), com propósito de melhorar os níveis de interoperabilidade estabelecidos por *European Interoperability Framework* (EIF) e entre a e-PING e Padrão de Metadados do Governo Eletrônico (ePMG). Irá aumentar o nível de interoperabilidade entre os padrões existentes no Governo Eletrônico Brasileiro e expandir o MNI para outros setores.

e) Fazio et al. (2015): apresentaram uma arquitetura de *software* para adoção da FIWARE para projetos em Internet das Coisas. Este trabalho descreve os principais componentes da FIWARE que foram adotados para o projeto da arquitetura.

f) Preventis et al. (2016): propuseram uma arquitetura compatível com IoT, que utiliza os GE da plataforma FIWARE. Ademais, compararam os projetos utilizando a FIWARE com projetos IoT, com objetivo de detectar os principais recursos da FIWARE para permitir implementações entre sistemas compatíveis com projetos IoT. Os autores destacaram que há falta de ferramentas GE e que as mesmas precisam ser implementadas para oferecer funcionalidades necessárias para as camadas de arquitetura especificadas pela IoT.

g) Pinto (2016): propõe uma arquitetura utilizando a FIWARE para o tratamento de dados heterogêneos promovendo um papel ativo aos cidadãos e desenvolvedores de aplicativos móveis. Essa arquitetura coletará as informações sobre os dados móveis das ocorrências de envolvimento dos cidadãos e apoiar configurações infinitas para modelos empresariais ou desenvolvimento de ferramentas de suporte para os funcionários do governo da cidade.

Conforme pode ser observado nos trabalhos apresentados acima, nenhum deles abrangiam os dois temas. Os trabalhos de Santos (2008), Santos e Reinhard (2012), Machado, Santos e Junior (2013) e Silveira, Wazlawick e Rover (2015) somente contribuem com relação a arquitetura e-PING. Por fim, os trabalhos de Fazio et al. (2015), Preventis et al. (2016) e Pinto (2016) somente contribuem com relação a plataforma FIWARE. Portanto, evidencia-se a importância e

o relacionamento de cada um dos trabalhos apresentados com o trabalho proposto. O trabalho proposto, contempla esses dois temas.

1.5 Metodologia

Este capítulo tem como objetivo descrever os critérios adotados na metodologia para o desenvolvimento deste trabalho. A Seção 1.5.1 apresenta o arcabouço para execução do mapeamento sistemático, assim como a classificação da natureza da pesquisa realizada. Enfim, a Seção 1.5.2, descreve as etapas da dissertação.

1.5.1 Classificação de Pesquisa

Conforme apresentado na Seção 1.3, o principal objetivo deste trabalho é desenvolver um *Framework* que permita a interoperabilidade dos *Softwares* Públicos que aderem o uso da e-PING, suas características e vantagens, no contexto de Cidades Inteligentes, apoiando a plataforma aberta FIWARE, por meio da e-PING para as aplicações que utilizam esta plataforma. A fim de contemplar o objetivo proposto é adotada a metodologia descrita a seguir.

A metodologia de pesquisa é um passo essencial para o pesquisador, que contém o objetivo de aumentar o conhecimento que será construído, atingir os resultados e colaborar na condução da pesquisa (GIL, 2017). Essa metodologia é dividida em quatro tipos: quanto à natureza, a abordagem dos dados, aos objetivos e aos procedimentos técnicos (WAZLAWICK, 2015).

Figura 1 – Tipos de Pesquisa e suas características.

Fonte – Autora (2019).

A Figura 1 apresenta os elementos de realização concreta de uma investigação planejada, desenvolvida e redigida, de acordo com as normas da metodologia adequada para a pesquisa. Considerando a metodologia adotada, pode-se classificar a relação quanto à natureza, em duas: básica ou aplicada (GERHARDT; SILVEIRA, 2009). Portanto, diante desse contexto, utilizou-se a pesquisa aplicada, que visa conhecimentos para aplicações práticas, com o objetivo de solucionar problemas específicos (SILVEIRA; CÓRDOVA, 2009). Ou seja, o *Framework* irá direcionar as organizações que utilizam a arquitetura e-PING com as aplicações que utilizam a plataforma FIWARE.

Dadas as características gerais da pesquisa, pode haver diferentes abordagens dos dados, podendo elas serem qualitativa ou quantitativa. Pesquisa qualitativa é uma pesquisa que busca aprofundar os fenômenos complexos específicos e que não se preocupa com análises estatísticas, para análises e classificações (MERRIAM, 1998; FONTELLES et al., 2009). A pesquisa quantitativa é uma pesquisa que contém um número representativo da população para classificá-los e analisá-los estatisticamente, tais como média, desvio padrão, mediana, correlação. A técnica essencial para coleta de dados para essa pesquisa é o questionário estruturado (MALHOTRA, 2012). Este trabalho é classificado como uma pesquisa qualitativa por se tratar de um *Framework* de melhoria para o GEB.

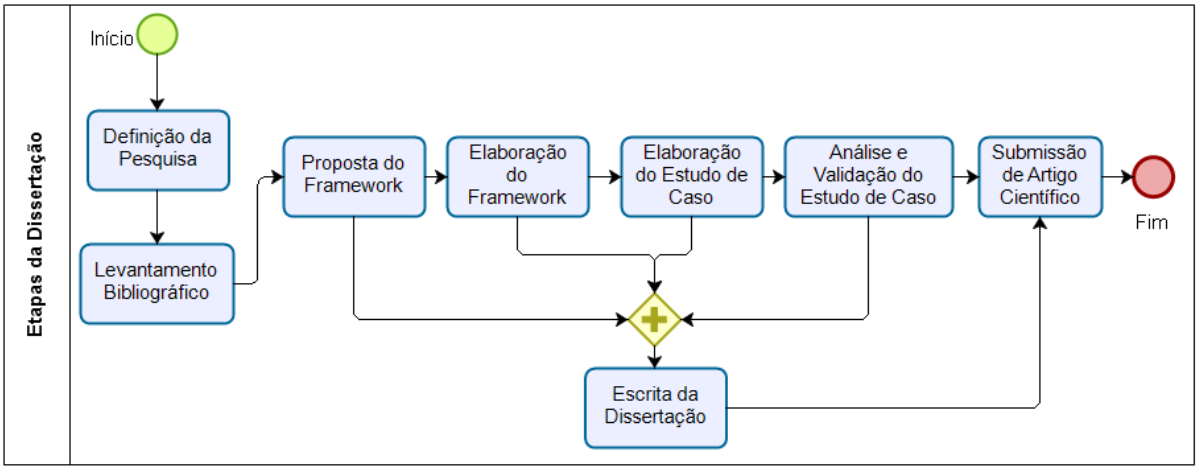
Observando a relação aos objetivos, a pesquisa pode ser classificada como exploratória, descritiva ou explicativa. Esta pesquisa classifica-se como exploratória pois, segundo Gil (2017) uma pesquisa exploratória tem como objetivo tornar o problema mais familiar, isto é, tornar evidente o que de fato há de se resolver (GIL, 2017). Ou seja, seu contexto é devido à familiarização com o experimento investigado, possibilitando melhor precisão no desenvolvimento.

Fazendo menção aos procedimentos técnicos, a pesquisa em si pode conter diferentes classificações, sendo elas: pesquisa de campo, bibliográfica, experimental, estudo de caso, *survey* e dentre outras. Este trabalho é considerado como bibliográfico e estudo de caso. A pesquisa bibliográfica representa por resultar em estudos de artigos, dissertações, monografias, teses, livros, jornais ou quaisquer outras publicações concedidas por ferramentas acessíveis ao público (WAZLAWICK, 2015). Dessa maneira, será descrita a solução do trabalho. Por fim, a pesquisa de estudo de caso busca validar o *Framework*, em forma de provas de conceito, objetivando uma investigação profunda e exaustiva com o intuito de verificar o desempenho de acordo com os critérios estabelecidos (GIL, 2017).

1.5.2 Etapas da Pesquisa

A Figura 2 ilustra o processo de condução para esta pesquisa, que contêm 8 atividades desenvolvidas sequencialmente.

Figura 2 – Atividades da Dissertação.



Fonte – Autora (2019).

A seguir serão descritas as oito atividades:

1. **Definição da Pesquisa:** consistiu na caracterização do problema que foi trabalhado de forma precisa, a fim de obter coesão entre o estudo e o problema levantado. Assim, proporcionou um estudo bem definido e detalhado atingindo os objetivos propostos.

2. **Levantamento Bibliográfico:** consistiu em identificar pesquisas em bases de artigos científicos, teses, dissertações e demais. Essa pesquisa auxiliou para a fundamentação teórica do presente trabalho (apresentada no Capítulo 1). Para esse levantamento, foram realizadas buscas nas bases apresentadas na Seção 1.4.

3. ***Framework Smart* e-PING**: essa atividade teve como objetivo propor um *Framework*, denominado *Smart* e-PING, que pretendeu interoperar entre um *Software* Público, com a necessidade de inseri-lo no contexto de Cidades Inteligentes, utilizando uma plataforma *middleware*.

4. **Elaboração do *Framework Smart* e-PING**: consistiu em desenvolver o *Framework* proposto.

5. **Elaboração da Validação do *Framework Smart* e-PING:** refere-se a validação que abrangeu uma investigação profunda e exaustiva em ambiente real, com o intuito de obter conhecimento detalhado do objeto de estudo. Logo, neste trabalho foi utilizado um *Software* Público, denominado GP*Web*[3].

6. **Análise e Validação do Estudo de Caso:** visou atingir os objetivos desejados de acordo com uma análise descritiva em relação a validação, tendo como base o *Framework* proposto.

[3] Disponível em <https://softwarepublico.gov.br/social/gpweb> ou <https://www.sistemagpweb.com/>.

7. **Escrita da Dissertação:** consistiu na elaboração da escrita da dissertação. Ademais, evidenciou as consequências proporcionadas do trabalho de mestrado, pelo qual tornou os resultados mais evidentes, o que pôde proporcionar a construção de outros trabalhos. Essa etapa de pesquisa foi realizada paralelamente entre as atividades 3 a 6.

8. **Submissão de Artigo Científico:** submissão de artigo científico em revista, conferência, jornal ou outro meio de publicação.

1.6 Metodologia Adotada para a Execução do Mapeamento Sistemático

Nesta seção foi detalhado o objetivo específico **"Investigar a utilização de outras plataformas de IoT abertas voltadas para Cidades Inteligentes"**, respectivamente apresentado na Seção 1.3 deste trabalho.

A realização deste mapeamento sistemático teve como objetivo identificar plataformas, arquiteturas, *middlewares* e/ou *Frameworks* sugeridos para Cidades Inteligentes na Administração Pública. Dessa forma, seguiu-se os protocolos estabelecidos por Petersen et al. (2008).

Nas próximas subseções, serão apresentadas a realização deste mapeamento sistemático. Dessa maneira, outros pesquisadores que possuam interesses neste trabalho poderão repetir/avaliar de forma mais precisa, podendo efetuar os mesmos passos.

1.6.1 Questão de Pesquisa

O escopo para utilização deste mapeamento sistemático relaciona-se em identificar a importância da adoção das Cidades Inteligentes. Portanto, a questão de pesquisa formou-se com o intuito de alcançar o objetivo desta revisão.

Questão de pesquisa da dissertação: Quais (*Framework*/ *Middleware*/ Plataforma/ Arquitetura) são sugeridos para Cidades Inteligentes?

1.6.2 Estratégia e Fontes de Busca

A base de dados utilizada para esta pesquisa na área da computação foi a *Scopus*. A *Scopus* foi escolhida por ser uma das maiores fontes de referência a qual abrange trabalhos das seguintes bases: *ACM Digital Library*, *IEEE Xplore*, *ScienceDirect*, dentre outras (SCOPUS, 2018).

Para a execução da pesquisa na base foram utilizadas *strings* de busca para realçar as definições dos termos ou palavras chaves a serem utilizadas. Nessa pesquisa foram adotados métodos de filtragem, como o título (*title*), o resumo (*abstract*) e as palavras-chaves (*keywords*) dos artigos. Ademais, também foram considerados artigos em inglês e da área de pesquisa

referente a Computação, com o objetivo de remover os artigos que não contribuíram para a temática abordada.

Na Tabela 1 são exibidos os termos genéricos utilizados na realização do mapeamento sistemático, bem como a *String* de busca da *Scopus* com tais termos.

Tabela 1 – Termos e *String* de busca utilizados no mapeamento.

Termos Genéricos	*String* da Scopus
Smart City, *Middleware*, *Framework*, *Platform*, *Architecture*	*TITLE-ABS-KEY*("*smart cit**") *OR* ("*smartcit**") *OR* ("IoT") *OR* ("*Internet of Things*")) *AND* ("*middleware*") *OR* ("*Framework*") *OR* ("*architecture*") *OR* ("*platform*"))).

Fonte – Autora (2019).

Conforme apresentado na Tabela 1, cinco termos (em inglês) foram definidos para a realização da busca. Uma vez que os termos são considerados de suma importância, foi possível dar origem a *string* de busca estabelecida acima.

Após a conclusão das pesquisas na *Scopus*, começou a seleção dos artigos filtrados com base nos critérios e análise dos métodos de seleção definidos na seção a seguir.

1.6.3 Critérios de Seleção e Procedimentos de Estudo

Com o propósito de excluir os estudos que não colaboram com as questões de pesquisa, foram definidos os critérios de Inclusão e os de Exclusão, características essenciais de um Mapeamento Sistemático (PETERSEN, 2011). Ademais, esses critérios são essenciais para evitar os pontos de vista dos avaliadores no processo de Inclusão e Exclusão. De acordo com a Tabela 2, foram definidos esses critérios.

Tabela 2 – Critérios de Inclusão e Exclusão.

Critérios de Inclusão	Critérios de Exclusão
a) Estudos sobre plataformas, arquiteturas, *middlewares* e/ou *Frameworks* sugeridos para Cidades Inteligentes; b) Estudos abrangendo o ano de publicação superior a 2006; c) Estudos contendo o escopo da pesquisa: artigos, anais de conferências e jornais. d) Estudos contendo os idiomas inglês e português.	a) Estudos que não especificam quais plataformas, arquiteturas, *middlewares* e/ou *Frameworks* sugeridos para Cidades Inteligentes; b) Estudos em que o *abstract* não define claramente o objetivo/contribuição do trabalho; c) Estudos abrangendo o ano de publicação inferior a 2007; d) Estudos duplicados; e) Estudos que não contemplam os idiomas inglês ou português.

Fonte – Autora (2019).

1.6.4 Análise dos resultados do Mapeamento Sistemático da Literatura

O Mapeamento Sistemático teve como objetivo identificar as plataformas, arquiteturas, *middlewares* e/ou *Frameworks* sugeridos para Cidades Inteligentes. Uma única questão que foi analisada, foi aquela que os artigos continham a apresentação das plataformas, arquiteturas, *middlewares* e/ou *Frameworks*.

O Mapeamento Sistemático foi realizado entre os meses de março de 2017 e agosto de 2017. Com a aplicação da *string* de busca e palavras chaves em inglês na base da *Scopus* foram encontradas inicialmente 8.584 artigos. Após as buscas, iniciou-se a seleção por meio dos critérios de Inclusão e Exclusão, retornando 1.500 artigos para efetuar uma análise mais aprofundada. Desses 1.500 artigos, 73 foram selecionados para estabelecer a resposta deste mapeamento sistemático. Para cada artigo selecionado, foram extraídas informações dos *middleware* e/ou *Framework* e a sua descrição. No Capítulo 3 são detalhadas as plataformas identificadas nesse Mapeamento Sistemático. Por fim, no Capítulo 2, na Seção 2.5.1, é detalhado o *middleware* mais utilizado para Cidades Inteligentes.

1.7 Organização da Dissertação

A organização desta dissertação está organizada em cinco capítulos. Os tópicos a seguir descrevem o conteúdo de cada um deles:

No Capítulo 2 é apresentada a fundamentação teórica, que embasa os conceitos fundamentais para o desenvolvimento do trabalho, tais como: Interoperabilidade, e-PING, Cidades Inteligentes, Requisitos *Middleware* para Cidades Inteligentes e *Middleware* para Cidades Inteligentes.

No Capítulo 3 são apresentadas as plataformas *middlewares* para Cidades Inteligentes existentes na literatura e uma avaliação quanto ao atendimento dos requisitos e plataformas apresentadas.

No Capítulo 4 é descrito o *Framework Smart* e-PING, descrevendo todas as etapas.

No Capítulo 5 é apresentada a validação do *Framework Smart* e-PING.

Finalmente, no Capítulo 6, são apresentadas as conclusões, as dificuldades, os trabalhos futuros e as contribuições relacionadas à dissertação.

2

Fundamentação Teórica

Neste capítulo são apresentados os conceitos teóricos relacionados a Interoperabilidade, Padrão de Interoperabilidade do Governo Eletrônico, *Softwares* Públicos, Cidades Inteligentes, *Middleware*, Requisitos de *Middlewares* para Cidades Inteligentes e Arquitetura de *Software*.

2.1 Interoperabilidade

Interoperabilidade (do inglês, *interoperability*) é a integração e comunicação entre sistemas, organizações, serviços, aplicações ou plataformas trabalhando em conjunto (SANTOS; ROCHA; CARVALHO, 2014). As organizações, pessoas ou sistemas computacionais (distribuídos, independentes e desenvolvidos em diferentes tecnologias ou plataformas) deveriam compartilhar informações de maneira eficaz e eficiente (SANTOS; ROCHA; CARVALHO, 2014; DESAI; SHETH; ANANTHARAM, 2015). Com a interoperabilidade, são elaborados processos eficientes, com a finalidade de reduzir os custos organizacionais e prevenir a redundância dos dados entre os sistemas utilizados (SERRANO et al., 2015). A interoperabilidade também é considerada uma conexão entre diferentes computadores com a capacidade de compartilhar informações, em que é necessário haver a cooperação das organizações (BARBOSA et al., 2016).

Quando os sistemas são interoperáveis, eles apresentam características importantes como a eficiência, melhor prestação de contas (*accountability*) e serviços, redução de custos, reuso de soluções, escalabilidade das soluções, maior transparência, coordenação nas ações do governo e a promoção da cooperação internacional. Outrossim, possibilita a comunicação de diferentes dispositivos (serviços, sistemas, aplicações) interconectados na rede, proporcionando o compartilhamento de informações e eliminando as informações redundantes, independente do protocolo, *software* ou *hardware* que está sendo utilizado (SANTOS; REINHARD, 2010; DELICATO; PIRES; BATISTA, 2013; MELLO; MESQUITA; VIEIRA, 2015).

Para possibilitar a interoperabilidade, é necessário haver a cooperação das organizações,

e essas devem estar em conformidade sobre a forma como interoperar (SANTOS; REINHARD, 2012). É essencial o conhecimento da utilização de padrões tecnológicos utilizados nos mecanismos de interoperabilidade a fim de reduzir o esforço na construção das interfaces de integração e comunicação de maneira ágil e eficaz (JANSSEN; ESTEVEZ; JANOWSKI, 2014).

A interoperabilidade é um problema complexo e possui desafios a serem superados para conseguir obter o sucesso (CURRY, 2012). Os autores Curry (2012) e Scholl e Klischewski (2007) apontam dez desafios encontrados na implementação da interoperabilidade, que são:

a) **Conceituais:** visa garantir a interpretação/significado exato das informações trocadas;

b) **Tecnológica:** relacionadas à heterogeneidade das plataformas de sistemas gerando a incompatibilidade de protocolos, codificação, infra-estruturas, padrões ou plataformas;

c) **Organizacionais:** corresponde à incompatibilidade das responsabilidades, autoridades e estruturas organizacionais;

d) **Constitucional/Legal:** referente a separação dos poderes, níveis e ramos do governo. Oferece e integração de sanções e interoperação dentro de certas fronteiras;

e) **Jurídica:** operam de forma independentemente uns dos outros. Possuem interação voluntária das suas informações e processos de negócios, não podendo ser imposta;

f) **Colaborativas:** referente às organizações distintas que não estão dispostas e disponíveis para a colaboração e interoperação;

g) **Informacionais:** diz respeito às organizações que promovem o uso de informações compartilhadas, porém não as compartilham;

h) **Custos:** refere-se às restrições orçamentárias inesperadas das organizações;

i) **Desempenho:** relacionada ao desempenho dos sistemas em termos de resposta quando há um grande número de sistemas interoperando entre si;

j) **Gerencial:** referente aos diferentes interesses e necessidades das organizações. Quando os interesses são comuns se torna mais fácil a interoperabilidade e a integração dos sistemas.

A interoperabilidade pode ser projetada no início do sistema tornando-se eficaz, fácil e menos custosa. Por outro lado, a interoperabilidade pode ser adaptada em um sistema existente, porém demandará grande esforço em seu processo de adaptação. Esses esforços podem afetar os aspectos desse sistema existente, como por exemplo os protocolos de comunicação. Estas considerações podem representar obstáculos à adaptação da interoperabilidade aos sistemas existentes (MADNI; SIEVERS, 2014).

2.2 Padrão de Interoperabilidade do Governo Eletrônico (e-PING)

A arquitetura e-PING é um padrão de interoperabilidade do Governo Eletrônico Brasileiro (GEB) que vem sendo desenvolvido desde 2003 pelo Governo Federal Brasileiro para ser utilizado em seus órgãos. Essa arquitetura foi criada pelo Departamento de Governo Eletrônico, que faz parte da Secretaria de Logística e Tecnologia da Informação, localizada no Ministério do Planejamento, Orçamento e Gestão (BRASIL, 2007; BARROS; CEPIK; CANABARRO, 2010).

O foco da especificação inicial dessa arquitetura é o Poder Executivo do Governo Federal Brasileiro, porém a estrutura da arquitetura contempla informações dos Poderes Legislativo e Judiciário do Governo Federal, Governos Estaduais e Municipais, o Ministério Público e organizações nacionais e internacionais. Esse padrão contém um conjunto de políticas, premissas e especificações técnicas que estabelece utilização das Tecnologia de Informação e Comunicação (TIC) na interoperabilidade dos serviços eletrônicos do governo no Brasil, ao qual determina condições para integração com outras instituições governamentais (BRASIL, 2007).

Figura 3 – Políticas gerais utilizadas na construção da e-PING.

Fonte – Adaptado de Brasil (2016).

Conforme apresentado na Figura 3, consiste em 5 políticas gerais utilizadas na sua construção, uma vez que estas são especificações técnicas relacionadas às soluções quanto a interoperabilidade. A seguir são descritas essas políticas (BRASIL, 2016):

a) **Adoção preferencial de padrões abertos:** especifica que são adotados padrões abertos, porém os padrões proprietários são aceitos de forma transitória, se não houver perspectivas substituições de imigração respeitando os requisitos estabelecidos de segurança e integridade;

b) **Uso de *Software* Livre e/ou *Software* Público:** os padrões definidos devem priorizar o uso dos *softwares* livres e/ou *softwares* públicos, em conformidade com as diretrizes do Governo Eletrônico e as normas do Sistema de Administração de Recursos de Tecnologia da Informação;

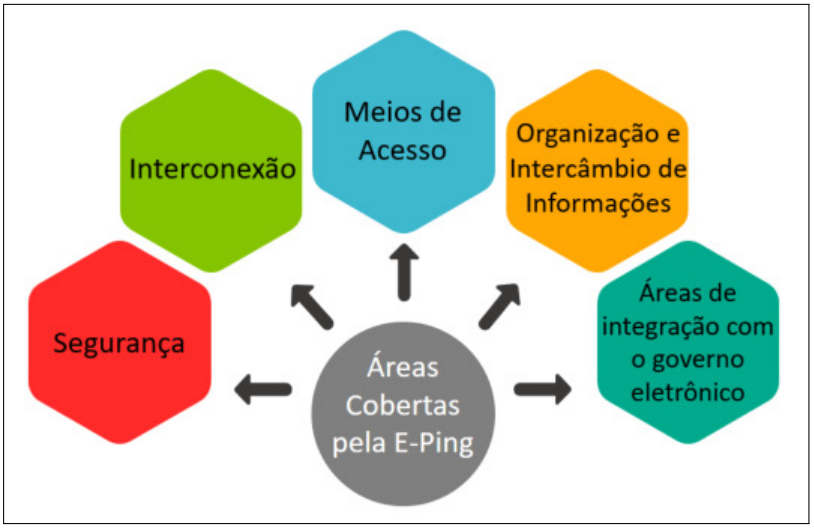
c) **Transparência:** todos os documentos da e-PING devem estar disponíveis via Internet para a disposição da sociedade;

d) **Segurança:** considerar o nível de segurança exigido na prestação de serviços, com a máxima transparência;

e) **Suporte de mercado:** todas as especificações técnicas incluídas na e-PING abrangem soluções altamente utilizadas no mercado, a fim de reduzir os riscos e os custos na produção dos serviços nos sistemas governamentais.

Segundo Brasil (2016), a e-PING é considerada uma estrutura básica utilizada para a estratégia do GEB que foi definida por meio da arquitetura *Government Interoperability Framework* (e-GIF), que foi desenvolvida pelo governo britânico em 2000. Devido ao tempo de implementação e sua evolução contínua, o e-GIF se tornou uma grande referência para a e-PING.

Figura 4 – Áreas cobertas pela e-PING.



Fonte – Adaptado de Brasil (2016).

Conforme apresentado na Figura 4, esse padrão foi segmentado em cinco grupos:

a) **Interconexão:** trata da padronização para que os órgãos do governo e a sociedade se interconectem;

b) **Segurança:** refere-se aos aspectos de segurança de TIC para proporcionar a validade e a privacidade das operações;

c) **Meios de Acesso:** determina as políticas e especificações dos padrões dos dispositivos de acesso aos serviços do Governo Eletrônico;

d) **Organização e intercâmbio de informações:** apresenta a forma de modelar/ organizar as informações para possibilitar a reutilização das informações de uma maneira simples;

e) **Áreas de integração para o Governo Eletrônico:** define as especificações técnicas para o intercâmbio das informações e intercâmbio baseados nas definições da e-PING, em áreas transversais da atuação governamental.

A segmentação Meios de Acesso apresenta uma política geral que fornece recursos para *middlewares* (BRASIL, 2016). Esses *middlewares* são utilizados para Cidades Inteligentes (ATZORI; IERA; MORABITO, 2010; BORGIA, 2014).

2.3 *Softwares* Públicos

Na década de 1960 os pesquisadores começaram a compartilhar seus códigos, devido a indisponibilidade de soluções comerciais nessa época (HERTEL; NIEDNER; HERRMANN, 2003). Com o crescimento significativo dos *Softwares* Livres nas organizações ao passar dos anos, os pesquisadores adotaram essa solução, mais rápida e inovadora, em suas organizações (MARSAN; PARÉ; BEAUDRY, 2012). Para Gwebu e Wang (2011) *Software* Livre é uma solução importante que proporciona a flexibilidade na reutilização de inovações desenvolvidas por outros, investigar e adaptar às suas necessidades, agregando qualidade em suas aplicações.

O termo *Software* Livre ou *open source* é um *software* de código-aberto que disponibiliza ao público o código-fonte de um determinado *software*. Um *Software* Livre permite aos seus desenvolvedores executar, estudar, compartilhar, busca melhor gestão de recursos no desenvolvimento, colaborar no desenvolvimento, aperfeiçoar o código em seus projetos adequando as suas necessidades e a possibilidade de outros desenvolvedores se beneficiarem dessas mudanças (ALVES; PESSÔA; SALVIANO, 2011; SANTANA; ALENCAR; CORREIA, 2017).

A utilização de *Software* Livre oferece vantagens como a flexibilidade na reutilização de soluções/inovações desenvolvidas por outros, a inovação tecnológica, possibilita a investigação e aperfeiçoamento do código em seus projetos, adequando as suas necessidades. Outrossim, evita duplicação de esforços e identifica e corrige *bugs* em menos tempo (HERTEL; NIEDNER; HERRMANN, 2003; ALVES; PESSÔA; SALVIANO, 2011; GWEBU; WANG, 2011; MARSAN; PARÉ; BEAUDRY, 2012; SANTANA; ALENCAR; CORREIA, 2017). O *Software* Livre é visto como uma solução importante para os desenvolvedores, que proporciona aperfeiçoar a experiência desses usuários, assim como o código-fonte que é público. Dessa forma, o *Software* Público[1] é um tipo de *Software* Livre que possui as seguintes diferenças: (i) a proteção do seu

[1] <https://softwarepublico.gov.br/social/>.

código-fonte relacionados aos artefatos da constituição do *software*; e (ii) vinculado às responsabilidades da administração pública no processo de disponibilização de soluções à sociedade em geral (MEIRELLES et al., 2017).

Uma vez considerado como um bem público, é necessário atender a determinados pré-requisitos legais em relação ao controle de patrimônio público, ou seja, registrá-lo no Portal do *Software* Público Brasileiro (PSPB) (FERREIRA; MEIRELLES; NERI, 2017). A fim de disponibilizar um ambiente capaz de permitir o compartilhamento de projetos, o PSPB foi disponibilizado em 2007, possibilitando a criação de comunidades para o desenvolvimento de cada projeto de forma individualizada (SANTANA; ALENCAR; CORREIA, 2017).

O PSPB se trata de um ambiente *Web* que permite à sociedade e órgãos brasileiros compartilhar seus *softwares*, bem como disponibilizar seus manuais de instalação, fóruns de discussão, e documentação necessária para aperfeiçoamento proveniente dessas contribuições. O PSPB oferece: (i) um espaço para discussões; (ii) listas de e-mails; (iii) sistemas de controle de versões; (iv) recursos para incentivar os desenvolvedores, fomentando o desenvolvimento econômico das áreas em que promoveu a criação de empregos no país; e (v) redes sociais (MEIRELLES et al., 2017).

Com base no número de *softwares* disponibilizados no PSPB, observa-se que houve uma demanda significativa pelos serviços no portal, à medida que são criados usuários e estes passem a utilizar de forma efetiva o portal. Todavia, ainda não há uma efetividade comprovada no compartilhamento do conhecimento, que se contrapõe à ideia principal do portal. Isso ocorre devido a insuficiência técnica ou capital dos usuários, que existe o interesse pelos *softwares*, porém não há conhecimento disponível para aperfeiçoá-los. Dessa forma, é apresentada a demanda para o PSPB a fim de desenvolver de maneira técnico-cientifica os usuários do portal (SANTANA; ALENCAR; CORREIA, 2017; MEIRELLES et al., 2017).

2.4 Cidades Inteligentes

O termo Cidades Inteligentes (CI) contém uma variedade de sinônimos, que podem ser frequentemente substituídos por "*smart city*" ou adjetivos alternativos, como por exemplo: "*intelligent city*", "*virtual city*", "*ubiquitous city*", "*sustainable city*", "*learning city*", "*green city*", "*knowledge city*", "*information city*", "*virtual city*", "*digital city*" ou "*wired city*". Porém, o mais utilizado é *smart city* ou *smartcity* (COCCHIA, 2014; ALBINO; BERARDI; DANGELICO, 2015).

Esse termo foi utilizado pela primeira vez na década de 1990, com foco eficiente de Tecnologias de Informação e Comunicação (TIC) para auxiliar no provimento de infraestrutura e serviços da cidade, podendo trazer melhorias para o dia a dia dos habitantes (ALBINO; BERARDI; DANGELICO, 2015; GARCIA; FORTES; NASCIMENTO, 2018). CI surgiu para solucionar os problemas enfrentados com o aumento populacional nas cidades, como por exemplo:

urbanos, transporte, saúde, energia, água, dentre outros (BEHERA; REDDY; ROY, 2016; AZIZ et al., 2016; KIM, 2017; SAHOO; RATH, 2018). Uma CI impulsiona o crescimento econômico sustentável e uma alta qualidade de vida para os cidadãos (KOMNINOS; TSARCHOPOULOS; KAKDERI, 2014; BOLÍVAR; LÓPEZ-QUILES, 2018).

Segundo Kourtit e Nijkamp (2012), CI é a junção de estratégias inovadoras e direcionadas ao conhecimento, a fim de aperfeiçoar os desempenhos competitivos dos resultados sociais, empreendedores, da infraestrutura e do capital humano. Para os autores Bakici, Almirall e Wareham (2013), Cidade Inteligente é um ambiente tecnológico que consiste em novas tecnologias em ambientes urbanos conectando as pessoas e as informações, em favor de um ambiente sustentável (KUIKKANIEMI et al., 2011), por meio de um ponto de vista geográfico e social (SHAPIRO, 2006; WINTERS, 2011). Para Bolívar e López-Quiles (2018), CI refere-se à satisfação do cidadão com a prestação dos serviços públicos, que são resultantes da alavancagem de aplicações de TIC.

Segundo Giffinger et al. (2007), as CI estão divididas em seis pilares essenciais, que são:

a) **Ambiente Inteligente (*Smart Environment*):** assegura a atratividade de condições naturais, proteção ambiental e controle de recursos, dentre outros.;

b) **Economia Inteligente (*Smart Economy*):** define a competitividade da inovação, do empreendedorismo, eficácia, dentre outros.;

c) **Governo Inteligente (*Smart Governance*):** referente à tomada de decisão comunicativa oferecendo serviços aos cidadãos e ao governo público;

d) **Mobilidade Inteligente (*Smart Mobility*):** objetiva a acessibilidade, disponibilidade das redes de TIC, dentre outros.;

e) **Modo de Vida Inteligente (*Smart Living*):** representa a qualidade de vida, incluindo a segurança, situações de saúde, ambiente, cultura, educação, dentre outros.;

f) **Pessoas Inteligentes (*Smart People*):** apresenta a qualificação, flexibilidade, criatividade dos cidadãos, dentre outros.

No entanto, a fim de melhorar a qualidade de vida não basta utilizar apenas esses eixos (MOSTASHARI et al., 2011). Desse modo, para que funcione de forma adequada é necessário dar semântica aos dados provenientes dos dispositivos para melhor atender as soluções que derrubem as barreiras de uma qualidade de vida. A interoperabilidade é um requisito essencial (ERGAZAKIS; METAXIOTIS; PSARRAS, 2004), porém, ainda é um grande desafio devido a heterogeneidade dos dispositivos, tecnologias, protocolos de comunicação, dentre outros (GAMA; TOUSEAU; DONSEZ, 2012; ZANELLA et al., 2014).

Para Ergazakis, Metaxiotis e Psarras (2004), CI objetiva incentivar a contínua criação, compartilhamento e atualização do conhecimento. Isso é alcançado por meio da interoperabilidade entre os próprios cidadãos da mesma cidade ou de cidades diferentes. Essas cidades devem conter uma alta capacidade para a aprendizagem e inovação, que é embutida na criatividade de sua população, suas instituições de criação de conhecimento e sua infra-estrutura digital para comunicação e gestão do conhecimento.

2.5 *Middlewares*

Segundo Sun e Blatecky (2004), o *middleware* ou mediador é um *software* utilizado por aplicações distribuídas para transportar dados ou informações de diferentes aplicativos, suportando uma sincronização escalável de replicação em sistemas distribuídos. Esse *software*, por meio de camadas lógicas, abstrai a complexidade existente no desenvolvimento e na execução desses sistemas. Borgia (2014) define um *middleware* como um *software* para o desenvolvimento de novos serviços e novas tecnologias integradas para aplicações específicas em Internet das Coisas (ou IoT, do inglês *Internet of Things*).

Um *middleware* contém uma camada de *software* entre as aplicações e as camadas de comunicações de rede, fornecendo confiabilidade, segurança, privacidade, consistência e disponibilidade dos serviços às aplicações (RAZZAQUE et al., 2016). O *middleware* facilita e coordena a integração dos dispositivos heterogêneos de computação e comunicação, apoiando a interoperabilidade das aplicações e domínios (LIU et al., 2016). Ele é essencial para IoT, computação distribuída e alto desempenho em computação paralela (SUN; BLATECKY, 2004).

Um *middleware* fornece uma interface abstrata que dá ao desenvolvedor uma visão uniforme de sistemas operacionais e redes de baixo nível. Mais precisamente, permite que cientistas e engenheiros utilizem e compartilhem de forma transparente recursos como computadores, dados, redes e instrumentos (SUN; BLATECKY, 2004). As plataformas de *middleware* surgiram para facilitar o desenvolvimento de aplicações, possibilitando a integração de dispositivos, pessoas, sistemas, dados e uma série de serviços necessários para as aplicações (LIU et al., 2016).

O *middleware* é uma camada entre o sistema operacional e aplicativos distribuídos, que esconde a complexidade e a heterogeneidade dos sistemas. Outrossim, proporciona um grau de transparência de distribuição (TANENBAUM; STEEN, 2007). Para Alonso et al. (2004), um *middleware* é basicamente um conjunto de abstrações de programação desenvolvidas para facilitar o desenvolvimento de sistemas complexos.

O *middleware* pode facilitar o processo de desenvolvimento por meio da integração heterogênea de dispositivos e comunicações, além de suportar a interoperabilidade dentro das diversas aplicações e serviços (RAZZAQUE et al., 2016). As plataformas de *middleware* contém diferentes aspectos gerais. Pode ser difícil identificar as semelhanças e obter uma perspectiva abrangente da funcionalidade que cada uma delas pode oferecer (ALONSO et al., 2004).

O *middleware* contém dois aspectos importantes que devem ser diferenciados. No primeiro, o *middleware* fornece abstrações de programação para projetar diferentes aplicativos distribuídos. No segundo, o *middleware* implementa a funcionalidade fornecida pelas abstrações de programação.

Os principais *middlewares* para a comunicação são: *Middleware Remote Procedure Call* (RPC), *Middleware* Orientado a Mensagem e *Middleware* para IoT (ALONSO et al., 2004; COULOURIS et al., 2011).

a) ***Middleware Remote Procedure Call* (RPC):** um *middleware* propicia abstrações de programação que escondem as complexidades dos aplicativos. Ao invés do programador ter que lidar com todos os aspectos de uma aplicação, o *middleware* será o responsável por isso. Por meio destas abstrações de programação, o desenvolvedor tem acesso a funcionalidades que de outra forma teriam que ser implementadas desde fases iniciais. Neste caso, utiliza-se o *middleware* de procedimento de chamadas remotas baseados no modelo cliente/servidor. O RPC é um tipo de *middleware* utilizado para a abstração de programação. Outrossim, permite a interoperabilidade entre as aplicações, localmente ou remotamente (COULOURIS et al., 2011).

b) ***Middleware* Orientado a Mensagem (*Messag-Oriented Middleware*) :** é um *middleware* utilizado para a troca de mensagens entre aplicações distribuídas de forma assíncrona. Ao utilizar esse *middleware* não há garantia de que sua mensagem será lida por outro aplicativo, nem o tempo que levará para essa mensagem ser entregue. Essas mensagens podem levar minutos para serem entregues, opostamente do RPC que entrega as mensagens em milissegundos ou segundos. As mensagens são enviadas e recebidas por meio do *First-In, First-Out* (FIFO), ou seja, a primeira mensagem enviada é a primeira mensagem a ser enviada da fila. Os clientes enviam essas mensagens e aguardam elas serem encaminhadas ao destino (ALONSO et al., 2004; COULOURIS et al., 2011).

c) ***Middleware* para IoT:** define como uma arquitetura para IoT que fornece uma camada de abstração entre os componentes físicos e as aplicações, sendo assim se torna possível criar uma camada de serviços em que o desenvolvedor de aplicações IoT não precisa conhecer detalhes do *hardware* para utilizá-lo (CHAQFEH; MOHAMED, 2012). A IoT possui dois conceitos fundamentais para seu funcionamento: Internet e Coisas. O primeiro é considerado a união das redes de computadores, que por meio do uso de protocolos é possível prover serviços e conectar com os dispositivos. Enquanto o segundo é considerado como um paradigma fundamental, uma vez que se baseia em qualquer dispositivo (Coisa) que possa ser conectado(a) à Internet (BANDYOPADHYAY et al., 2011; MIORANDI et al., 2012). Nestes dispositivos IoT existem uma certa transparência nas informações, bem como uma heterogeneidade de plataformas (BANDYOPADHYAY et al., 2011). Os desafios vão surgindo à medida que novas aplicações estão sendo desenvolvidas. Dessa forma,

uma vez que existe essa heterogeneidade, é necessário criar uma solução *middleware* para IoT abrangendo diferentes requisitos para solucionar os desafios presentes (BANDYOPADHYAY et al., 2011; MIORANDI et al., 2012; FERSI, 2015). Esses requisitos são descritos na Seção 2.6.

Detrás das abstrações de programação proporcionadas pelo *middleware*, existe uma infraestrutura de *software* que implementa essas abstrações. Atualmente, a tendência é aumentar a complexidade, pois os produtos tentam fornecer abstrações de programação cada vez mais sofisticadas, tornando as plataformas de *middleware* mais complexas (ALONSO et al., 2004).

2.5.1 FIWARE

FIWARE ou simplesmente FI-WARE é uma plataforma aberta de *middleware* para IoT desenvolvida pela Comissão Europeia e o setor privado para a implantação de aplicativos para Internet do Futuro (*Future Internet*). Essa plataforma é utilizada em modelos para CI (PAN, J.; PAUL, S.; JAIN, 2011; FERREIRA et al., 2017), por meio de dados abertos[2], proporcionando *Application Programming Interfaces* APIs desenvolvidas em componentes *Generic Enablers* (GE). Essas APIs são abertas, genéricas e extensíveis (RAZZAQUE et al., 2016). A FIWARE disponibiliza ferramentas para diferentes funcionalidades utilizadas no desenvolvimento de aplicações e serviços para CI, logística, energias renováveis, transportes sustentáveis, dentre outros. Elas asseguram a interoperabilidade e a criação de modelos padronizados de dados (PREVENTIS et al., 2016).

Os GE, chamados de habilitadores genéricos, compõem o núcleo da plataforma e são considerados peças principais para a construção funcional da FIWARE (ELICEGUI et al., 2017). Além de ser *open source*, ele é uma solução para softwares proprietários. Os GE tornam a plataforma viável para produção de *softwares*, criando uma plataforma padrão para desenvolvimento desses softwares para a IoT (DOUZIS et al., 2016). As suas implementações compõem um conjunto de componentes, que permitem funcionalidades proporcionados pelas API. Oferecem interfaces interoperáveis, conforme as especificações abertas estabelecidas para cada GE (MOLTCHANOV; ROCHA, 2014). Para cada GE é obrigatório conter todas as informações essenciais para construir produtos, que estejam dentro dos parâmetros desejados e habilitados para aplicar com outros GE, bem como, a possibilidade de modificar uma implementação (FERREIRA et al., 2017).

Os habilitadores genéricos estão divididos em capítulos técnicos (do inglês: *Technical Chapters*), que são chamados de conjuntos de funcionalidades fornecidas pelas APIs, usados para prover serviços dentro da plataforma, tais como: segurança, infraestrutura, dentre outros

[2] Dados abertos (definido pelo *Open Knowledge Internacional*) são dados que podem ser utilizados, reutilizados, acessados, compartilhados livremente por qualquer pessoa para qualquer finalidade sob uma licença das mesmas regras estabelecidas (BRASIL, 2011).

(ARMANDO et al., 2017). Atualmente, a especificação da FIWARE, possui 7 capítulos técnicos, que são (NAMIOT; SNEPS-SNEPPE, 2014):

a) ***Cloud Hosting*:** trata de uma abordagem mais moderna em relação a infraestrutura utilizada na hospedagem em *Cloud Computer* (computação na nuvem);

b) ***Data/Context Management*:** permite alto desempenho no processamento, exploração de informações de contexto e na gestão;

c) ***Internet of Things* (IoT) *Services Enablement*:** possibilita o aumento da disponibilidade de recursos que possuam contextos utilizáveis em aplicativos da FIWARE;

d) ***Applications, Services and Data Delivery*:** objetiva criar um ambiente harmônico e sustentável de modo que seja possível promover inovação;

e) ***Security*:** aspecto fundamental para garantir a integração como os novos padrões provenientes da Internet das Coisas;

f) ***Interface to Networks and Devices* (I2ND) *Architecture*:** busca atender três domínios. O primeiro se refere a RDS (redes definidas por *software*, ou do inglês, *Software Defined Network*). O segundo se refere a um *middleware* que permite integração avançada entre dispositivos robóticos onde o foco é a comunicação entre todos os GEs. Por fim, os próprios dispositivos robóticos com integrações com outros GEs;

g) ***Advanced Web-based User Interface*:** contém o objetivo de implementar interfaces interativas. Portanto, é necessário oferecer opções de aplicações e serviços que contenham abrangência.

Os principais GEs utilizam a interface *Next Generation Service Interface* (NGSI) como padrão para a comunicação entre as aplicações. O NGSI é um padrão de gerenciamento de contextos baseada nas especificações da *Open Mobile Alliance* (OMA), que oferece uma interface simples ao seus usuários, podendo ser utilizada em diferentes serviços *Web*. Desta forma, os desenvolvedores não necessitam tratar dos problemas de complexidade e fragmentação das tecnologias IoT (MOLTCHANOV; ROCHA, 2014; KOVACS et al., 2016)).

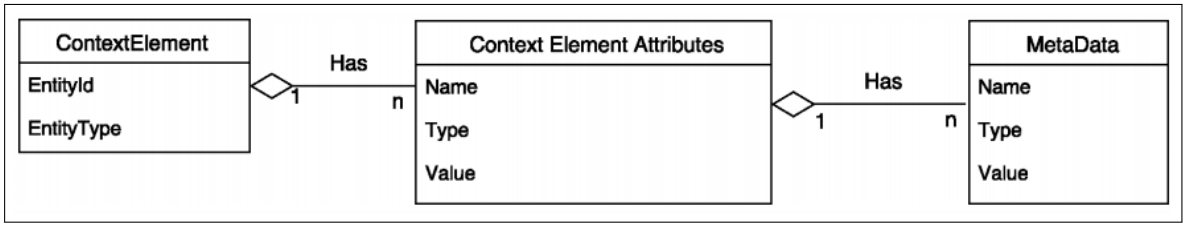
2.5.1.1 NGSI

A FIWARE utiliza o padrão NGSI para se comunicar com os seus componentes, padronizando as entidades de contexto. A fim de transformar os dados provenientes dos objetos em "coisas"é necessário consumir os dados baseados em um cenário, publicar e direcionar para aplicações. Essas entidades são representações de entidades ou objetos do mundo real, que podem ser utilizadas para representar um sensor, dentre outros (FIWARE, 2018).

Atualmente existem duas versões, a *Next Generation Service Interface* versão 1 (NGSIv1) e a *Next Generation Service Interface* versão 2 (NGSIv2). A diferença entre elas é que a NGSIv1 faz apenas requisições do tipo *GET*, enquanto a NGSIv2 consegue realizar todos os tipos de métodos em suas requisições (FIWARE, 2018).

A NGSI implementa basicamente três conceitos básicos, que são os de entidades (*Entity*), atributos (*attributes*) e elementos do contexto (*Context Elements*) (ELMANGOUSH et al., 2013). A Figura 5 apresenta o modelo de informação da especificação NGSI, ou seja, estrutura de dados com informações associadas às entidades de contexto.

Figura 5 – Modelo de informação da especificação NGSI.



Fonte – FIWARE (2018).

Conforme apresentado na Figura 5, um elemento de contexto possui um conjunto de atributos de contexto (*Context Element Attributes*) e metadados (*Metadata*). As entidades de contexto contém um identificador (*EntityId*) e um tipo (*EntityType*), formando uma entidade única em uma tupla <EntityId, EntityType> (MARTINEZ et al., 2017; LEA; BLACKSTOCK, 2014). Considerando o seguinte exemplo hipotético, tem-se uma identidade chamada "dengue" do tipo "doença".

As entidades são a personificação de objetos físicos, ou seja, se trata de uma descrição que contém atributos dos objetos e metadados relacionados a detalhes das características. Esses atributos, por sua vez representam uma instância de dados quanto a uma característica. No NGSI, os atributos possuem uma tupla contendo o nome, tipo e valor (ELMANGOUSH et al., 2013; FIWARE, 2018). No exemplo hipotético, a entidade "dengue" pode conter um atributo de localização, do tipo geolocalização e de valor "39000.000,-39000.000" indicando que a doença da dengue foi localizada em Aracaju.

O valor do atributo contém os dados reais e metadados. Os metadados podem descrever as propriedades do valor do atributo, como por exemplo a precisão, dentre outros (FIWARE, 2018). Os metadados possuem uma tupla contendo o nome, o tipo e o valor. Voltando ao exemplo hipotético, a representação do metadado do atributo de contexto "localização" da entidade "dengue", pode ser representado com o nome "precisão" que indica o valor do metadado "100" e tipo "inteiro", indicando que a localização está sujeita a margens de erro de 100 metros. A seguir o Código 1 apresenta um exemplo em *JavaScript Object Notation* (JSON) da entidade de contexto, utilizada no exemplo hipotético.

Código 1 – Exemplo JSON.

```
{
   "json": {
       "contextElements": [
               {
                   "atributes": [
                       {
                       "name": "localizacao",
                       "type": "geolocalizacao",
                       "value": "39000.000, -39000.000",
                           "metadatas": [
                               {
                                   "name": "precisao",
                                   "type": "inteiro",
                                   "value": "100",
                               }
                           ]
                       }
                   ]
                    "id": "dengue",
                    "type": "doenca"
               }
       ]
   }
}
```

Fonte – Autora (2019).

2.5.1.2 *Orion Context Broker*

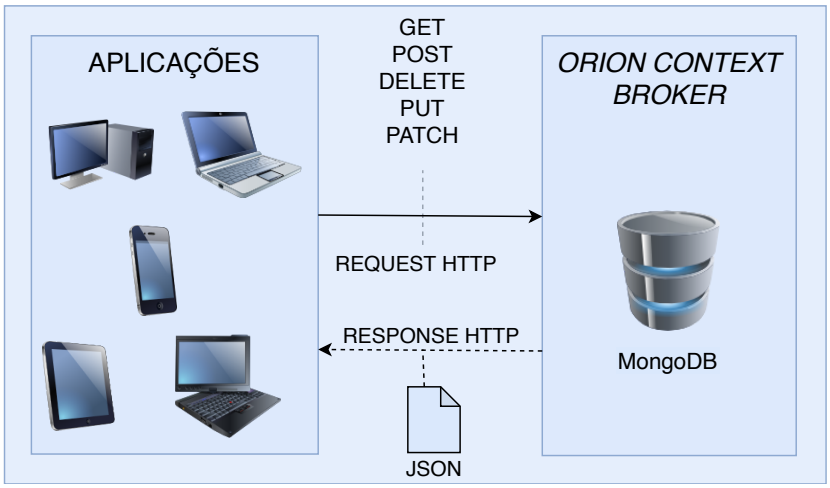
É uma implementação do NGSIv2 que faz parte da plataforma FIWARE, ou seja, é a base da plataforma FIWARE (KOVACS et al., 2016; ALONSO et al., 2018). Contém o objetivo de tratar diferentes tipos de dados durante todo o seu ciclo de vida, baseando-se em um contexto personalizável. Portanto, ao fazer uso dessa ferramenta é possível elaborar um contexto para realizar alterações dos dados durante seu ciclo de vida por meio de requisições (SUN et al., 2016).

O *Orion Context Broker* (ORION) oferece operações como as de atualizações, consultas, registros e assinaturas (*subscription*)[3]. O principal objetivo é fornecer diferentes acessos. O ORION contém um banco de dados de código aberto, denominado de MongoDB, que armazena informações de dados de contexto, como entidades de dados, assinaturas e registros. A comunicação do ORION é feita por meio de mensagens REST, utilizando o *JavaScript Object Notation*

[3] São assinaturas utilizadas para notificar as informações alteradas. Desta forma, quando acontece uma alteração nos valores de atributos das entidades, uma notificação é enviada para a aplicação externa (FIWARE, 2018)

(JSON). Esse banco armazena de forma normalizada as entidades. Essas entidades contém pelo menos dois campos, identificador (ID) e tipo (FIWARE, 2018; SALHOFER, 2018). A Figura 6 apresenta o funcionamento do ORION.

Figura 6 – Visão geral do funcionamento do ORION.



Fonte – Autora (2019).

A Figura 6 ilustra a visão geral do funcionamento do ORION. As aplicações requisitam o ORION por meio de requisições HTTP, como por exemplo *GET*, *POST*, *DELETE*, *PATCH* e *PUT*. O ORION por sua vez traduz o NGSI e solicita ao banco de dados MongoDB. A resposta do banco é recebida pelo ORION, que converte os dados para um arquivo JSON. Por fim, esses dados são passados como resposta, via HTTP, para a requisição feita pelo aplicativo. O ORION provê dados de forma independente e padronizados. Desse modo, mesmo que o dispositivo pare de funcionar ou se comunicar, é possível recuperar a informação com base no contexto que este se encontrava (FIWARE, 2018).

Uma vez que o ORION trata de diferentes tipos de dados baseados em um contexto, se torna importante para as aplicações IoT ou sistemas que tratem diferentes tipos de dados (ZAHARIADIS et al., 2014).

2.6 Requisitos de *Middlewares* para Cidades Inteligentes

No contexto de CI, uma plataforma *middleware* fornece um conjunto de requisitos para o paradigma de IoT (ATZORI; IERA; MORABITO, 2010). Tendo em vista que as CI integram o escopo de IoT, uma plataforma *middleware* para CI contém os mesmos requisitos de um *middleware* voltado para IoT, incluindo demais requisitos inerentes da aplicação (SCHAFFERS et al., 2011). Esses requisitos são provenientes da interação dos usuários com os aplicativos em uso ou que possivelmente podem modificar os parâmetros iniciais. Os requisitos são indispensáveis para o pleno funcionamento do *middleware*. A seguir são descritos esses requisitos.

a) **Adaptação Dinâmica:** é um requisito importante para aplicações, ao assegurar a qualidade e a acessibilidade das aplicações no desempenho da sua execução. Caso haja um mal funcionamento ou falhas nessas aplicações, pode resultar em ameaça à vida ou à saúde das pessoas, perda ou danos graves de equipamentos ou ambientais, entre outros. Os *middlewares* IoT precisam oferecer um funcionamento adequado para as aplicações heterogêneas, com a finalidade de evitar erros ou inconsistências no sistema. Consequentemente, é necessário analisar corretamente as mudanças na estrutura ou no comportamento do sistema e nos dispositivos que estão sendo introduzidos (CAVALCANTE et al., 2015; RAZZAQUE et al., 2016).

b) ***Big Data*:** é um requisito indispensável para as plataformas IoT. Com o crescimento de dispositivos, é necessário gerenciar, localizar, buscar e transferir o imenso volume dos dados provendo respostas rápidas em um tempo hábil. Para solucionar esse problema, é necessário acompanhar e analisar essa demanda, obtendo novos métodos para o tratamento dos grandes volumes de dados (BANDYOPADHYAY et al., 2011; SOLDATOS; SERRANO; HAUSWIRTH, 2012).

c) **Ciência de Contexto:** o contexto é representado por pessoas, lugares ou objetos pertinentes a troca de informações entre o usuário e um aplicativo (DEY, 2001). Esses contextos precisam ser utilizados da melhor maneira possível para ajudar decidir quais dados precisam ser processados, com base nos extraídos (ZIMMERMANN; LORENZ; OPPERMANN, 2007; ATZORI; IERA; MORABITO, 2010). Esse requisito é um componente central de um *middleware* IoT. Eles trabalham em ambientes inteligentes sendo responsáveis pela extração, gerenciamento e processamento dos dados provenientes dos dispositivos, por meio dos sensores e os extraem dos contextos necessários definindo o melhor esforço a ser tomado (ATZORI; IERA; MORABITO, 2010; KRCO; POKRIC; CARREZ, 2014; PFLANZNER; KERTÉSZ, 2016).

d) **Conectividade:** fazem o uso de diferentes meios para a troca de informações, proporcionando o aumento do volume de dados provenientes dos dispositivos, que utilizam a Internet para sua conectividade total e interoperabilidade via tecnologias, tais como: *Wireless Fidelity* (Wi-Fi)[4], *Near Field Communication* (NFC)[5], ZigBee[6], *Radio-Frequency*

[4] É uma tecnologia sem fio que pode se comunicar uns com os outros, por meio de uma rede local de computadores ou Internet. Essa comunicação contém uma alta qualidade de desempenho (VIJAYAKUMAR et al., 2018).

[5] É uma tecnologia *wireless* que permite o envio de dados por meio de ondas eletromagnéticas, eliminando o uso de cabos ou fios. Somente é necessário a aproximação física entre os dispositivos (MYNY; STEUDEL, 2016).

[6] É uma tecnologia que disponibiliza uma rede com extrema baixa potência de operação, buscando reduzir o consumo de energia nos dispositivos, a fim de estender a vida útil de suas baterias, podendo as mesmas durarem anos (RONEN et al., 2018).

IDentification (RFID)[7], 3G/4G e *Bluetooth*[8] (ATZORI; IERA; MORABITO, 2010; NITTI et al., 2016; KASSAL; STEINBERG; STEINBERG, 2018).

e) **Descoberta e gerenciamento de dispositivos:** possibilita a detecção de todos os dispositivos existentes em um ambiente IoT, tornando a sua presença conhecida na rede e realizando dinamicamente o atendimento dos requisitos das aplicações. É essencial para (i) desenvolver soluções para descobrir e gerenciar os dispositivos, no que se refere ao fornecimento das informações sobre a sua localização e do seu estado; (ii) permitir localizar ou desconectar dispositivos roubados, perdidos ou não reconhecidos; (iii) configurar/modificar os dispositivos automaticamente; e por fim, (iv) proporcionar a integração entre os dispositivos (BANDYOPADHYAY et al., 2011; CAVALCANTE et al., 2015; AMARAL et al., 2016).

f) **Escalabilidade:** refere-se à capacidade de suportar grandes números de dispositivos, serviços e funções sem afetar a qualidade dos serviços existentes em um uso intenso (GOMES; RIGHI; COSTA, 2014; RABBAH et al., 2016; PALADE et al., 2017). Escalar sistemas é uma dificuldade enfrentada pelas plataformas *middlewares*, especialmente na presença das plataformas de *hardware* e comunicações com os protocolos (GÓMEZ-GOIRI; IPIÑA, 2010). Para uma plataforma se tornar confiável, é necessário gerenciar problemas de escalabilidade para as aplicações operarem de forma eficiente, em pequena ou grande escala de ambientes (MATTERN; FLOERKEMEIER, 2010). Devido a essa dificuldade, surgiu o paradigma de Computação nas Nuvens com o objetivo de solucionar os problemas enfrentados pelas plataformas IoT (SOLDATOS; SERRANO; HAUSWIRTH, 2012).

g) **Ferramentas de Desenvolvimento:** é uma característica essencial com a finalidade de resolver desafios provenientes dos requisitos dos *middlewares* IoT. Para essa criação, são necessárias ferramentas, API ou interfaces gráficas que auxiliem no desenvolvimento das aplicações (BATISTA et al., 2016).

h) **Flexibilidade**: os *middlewares* passarão por constantes avanços, pertinentes ao surgimento de novas implementações, protocolos, padrões, *hardware*, dentre outros (FARAHZADI et al., 2017). A flexibilidade é umas das características importantes que precisa oferecer aos seus sistemas, pois terá que ser flexível para se adequar ao surgimento das novas tecnologias (ZHILIANG et al., 2011; MARQUES; GARCIA; POMBO, 2017).

[7] É uma identificação automática que emite sinais de rádio frequência, que armazena e recupera os dados remotamente. Para isso é utilizado dispositivos denominados de RFID, que foram criados para substituir a leitura de código de barras. Essa tecnologia permite que objetos comuns sejam identificados de maneira única, por meio de *tags* (GAMA; TOUSEAU; DONSEZ, 2012).

[8] É uma tecnologia que conecta e troca informações com segurança entre dois ou mais dispositivos, como por exemplo: *smartphones*, *notebooks*, computadores, dentre outros. Essa tecnologia é realizada por meio de uma frequência de rádio curta (MOHAMMADI et al., 2018).

i) **Interoperabilidade:** esse requisito é de suma importância, pois possibilita a criação eficiente das aplicações por meio da troca de dados ou serviços de forma significativa e maior agregação aos usuários sem quaisquer problemas e esforços adicionais de programação, sendo capaz de utilizar domínios de dispositivos. Atualmente, grande parcela dos dispositivos são criados com protocolos fechados, impedindo que outros dispositivos possam ser inseridos a suas aplicações. Há desafios referente a utilização da interoperabilidade. Para superar esse desafio, foram criados modelos semânticos que proporcionam a integração entre essas aplicações (SANTOS; REINHARD, 2010; DELICATO; PIRES; BATISTA, 2013).

j) **Portabilidade:** evita a restrição apenas de um *hardware* e tecnologias de sistemas operacionais, podendo ser acessado em qualquer hora ou lugar. Obter a portabilidade é um desafio crítico, pois há uma ampla heterogeneidade de dispositivos em sistemas operacionais diferentes e *hardwares* (FARAHZADI et al., 2017). Porém, há uma solução para esse desafio que são plataformas que utilizam, por exemplo, a linguagem de programação Java, pois nessa linguagem, além de outras, também existe uma portabilidade de alto nível chamada de portabilidade distribuída (ou paralela). Logo, o *Java Virtual Machine* (JVM) possibilita que uma mesma aplicação possa ser desenvolvida e executada em diferentes sistemas operacionais (TELES, 2018).

k) **Reusabilidade:** é a possibilidade de reutilizar diferentes *softwares* ou *hardwares*, que não há tecnologia específica para compor novas soluções (ATZORI; IERA; MORABITO, 2010). A reutilização reduz os custos e minimiza os desafios enfrentados pelos desenvolvedores ao criar seus novos projetos (FARAHZADI et al., 2017).

l) **Segurança e Privacidade:** são requisitos cruciais para as funções de uma solução IoT, pois a maioria das transmissões de dados (privados ou não) ocorrem por meio deles (BANDYOPADHYAY et al., 2011). Na maioria das vezes, as coletas de dados são transportadas por meio da rede insegura. Por esse motivo, um *middleware* IoT deve assegurar segurança nos dados trafegados em redes e permitir que somente os usuários com permissão acessem as informações (AL-JAROODI et al., 2010). Para que um sistema seja seguro, devem conter os seguintes princípios: suportar as autenticações de dados, controle de acesso e a privacidade do usuário (BANDYOPADHYAY et al., 2011).

m) **Usabilidade:** é um atributo de qualidade que avalia a facilidade de utilização das interfaces dos usuários facilitando o gerenciamento dos objetos e a interação com as informações disponibilizadas. Nas soluções *middlewares* para IoT, os usuários com alto grau de conhecimento técnico manipulam as soluções possibilitando a adaptabilidade de diferentes ambientes que utilizam tecnologia IoT (TELES, 2018).

2.7 Arquitetura de *Software*

Provavelmente, a primeira referência ao termo arquitetura de *software* ocorreu em 1969 em uma conferência sobre Engenharia de *Software* (KRUCHTEN; OBBINK; STAFFORD, 2006). Entre meados dos anos 1968 e 1972, os primeiros artigos foram publicados. Em 1968 Dijkstra (1968) publicou seu artigo sobre a estrutura do sistema multiprogramação (DIJKSTRA, 1983). Em 1972, o autor David Lorge Parnas publicou dois artigos, um sobre uma técnica para especificação de módulo de *software* com exemplos (PARNAS, 1972a; PARNAS, 1983) e outro sobre critérios a serem usados em sistemas de decomposição em módulos (PARNAS, 1972b; PARNAS, 2002).

Existem diferentes maneiras de definir esse termo (KRUCHTEN; OBBINK; STAFFORD, 2006). As arquiteturas de *softwares* são componentes arquiteturais essenciais para a organização de um determinado sistema, incluindo relacionamentos com o ambiente (ISO, 2011). Esses componentes são estabelecidos na tomada de decisão do arquiteto de *software*, que conduzem a concepção e a evolução de um sistema de *software* (PERRY; WOLF, 1992; ISO, 2011). Arquitetura de *Software* envolve uma estrutura e organização de componentes e subsistemas modernos, interagindo na formação desses (KRUCHTEN; OBBINK; STAFFORD, 2006).

O gerenciamento adequado da arquitetura de *software* é um dos fatores mais importantes para o sucesso no desenvolvimento e na evolução dos componentes de *software*. Para obter qualidade em um produto e no processo de desenvolvimento é necessário que a arquitetura esteja bem definida, evitando falhas e problemas em seu produto final (BASS; CLEMENTS; KAZMAN, 2012). Portanto, a arquitetura é indispensável no decorrer do desenvolvimento de um *software*, contemplando e especificando a estrutura geral do mesmo (VLIET, 2008).

Uma arquitetura bem definida/elaborada propicia o desenvolvimento de sistemas bem aruitetados. Outrossim, traz benefícios para as etapas de desenvolvimento, testes, manutenção do *software*. Já os *softwares* mal estruturados são difíceis de serem corrigidos, modificados ou completados (BOOCH, 2007). Nesse contexto, existem decisões que devem ser tomadas pelo arquiteto a fim de propiciar a adequada documentação da arquitetura (VLIET, 2008). A arquitetura proporciona a realização de atributos de qualidade desejados do sistema, tais como requisitos funcionais, requisitos não funcionais, evolução, reusabilidade, baixa complexidade e aprimoramento dos *stakeholders* (BASS; CLEMENTS; KAZMAN, 2012).

Segundo ISO (2011), uma arquitetura de *software* contém múltiplas visões. Essas visões são compostas por diferentes modelos, que por sua vez são representados por meio de diagramas em uma linguagem de modelagem. Existem diferentes linguagens de modelagens, a *Unified Modeling Language* (UML) é uma delas.

3

Plataformas de *Middlewares* para Cidades Inteligentes

Atualmente existem plataformas que são utilizadas com maior frequência no desenvolvimento de aplicações IoT ou em CI. Essas plataformas solucionam desafios propostos no desenvolvimento de aplicações IoT, porém dada a complexidade de tais soluções, acabam se tornando um grande desafio do ponto de vista da usabilidade.

Este trabalho buscou detalhar o resultado do objetivo específico **"Investigar a utilização de outras plataformas de IoT abertas voltadas para Cidades Inteligentes"**, respectivamente apresentado na Seção 1.3 do do Capítulo 1 deste trabalho. A fim de investigar e identificar essas plataformas, foi realizado um mapeamento sistemático, apresentado na Seção 1.6 do Capítulo 1.

Além deste trabalho, outros autores também analisaram plataformas para CI ou IoT comparando cada uma delas com os requisitos de *middleware* (apresentados na Seção 2.6 do Capítulo 2), como Bandyopadhyay et al. (2011), Perera et al. (2015), Pires et al. (2015), Teles (2018), dentre outros. Dentre as 19 plataformas aqui mencionadas, 11 foram baseadas do trabalho de Teles (2018) que são ASPIRE, Ebbits, EcoDiF, GSN, LinkSmart, OpenHAB, RestThing, SM4All, SOCRADES, *Task Computing* e Ubiware. No presente trabalho, foram mencionadas outras 8 plataformas, como por exemplo Carriots, City-Hub, FIWARE, Kaa, OpenIoT, S^3OIA, SIRENA e UbiSOAP.

3.1 ASPIRE

A plataforma *Advanced Sensors and lightweight Programmable middleware for Innovative Rfid Enterprise applications* (ASPIRE) é um *middleware* de IoT criado pela Comissão Europeia. Ela contém uma infraestrutura *open source* para a criação de soluções em ambientes industriais, que utilizam a tecnologia RFID (ASPIRE, 2018). ASPIRE é totalmente compatível com o conjunto de padrões *Eletronic Product Code* (EPC), que se preocupam com o processamento de dados em RFID nas arquiteturas centralizadas. Esses padrões são responsáveis

pela interoperabilidade e gerenciamento entre as aplicações que utilizam a tecnologia RFID (ANGGORJATI et al., 2010).

No que diz respeito à facilidade no desenvolvimento e implementação, esse *middleware* especifica a existência de uma *Integrated Development Environment* (IDE), que também pode ser chamada de AspireRFID IDE. Essa IDE permite aos desenvolvedores especialistas o acesso a todas as funcionalidades da plataforma, com a finalidade de definir, criar ou melhorar soluções para atender as necessidades desejadas (TELES, 2018).

A comunicação entre esses elementos ocorre pelos protocolos: *Hypertext Transfer Protocol* (HTTP) para transportar os dados, e o *Simple Object Access Protocol* (SOAP)[1] para o envio e encapsulamento dos dados (ANGGORJATI et al., 2010).

3.2 Carriots

É uma plataforma proprietária de aplicativos em nuvem para projetos relacionados a IoT, fornecendo um ambiente de desenvolvimento, APIs e hospedagens para o desenvolvimento de novas soluções. Ela permite que o usuário colete os dados de diferentes dispositivos, armazene-os e crie novas soluções para aplicativos, como Cidade Inteligente e energia inteligente, implantando e dimensionando serviços para uma variedade de dispositivos. Ademais, possibilita a integração com sistemas externos e gerencia os dados oriundos dos dispositivos (ZDRAVKOVIĆ et al., 2016). Essa plataforma fornece API REST para coletar e armazenar fluxos de dados de qualquer dispositivo por meio de uma conectividade (*gateway* ou 3G). A troca de dados nos dispositivos é realizada por meio do *Message Queue Telemetry Transport* (MQTT)[2] ou enviando para um HTTP/HTTPS API RESTful no formato XML ou *JavaScript Object Notation* (JSON) (MINERAUD et al., 2016; SINGH; KAPOOR, 2017). Outrossim, ela pode integrar com outros sistemas e extrair ou enviar dados com qualquer API, como por exemplo o Dropbox. Esses dados podem ser acessados por meio das estratégias PUSH e PULL da API (MINERAUD et al., 2017).

Carriots foi desenvolvida em Java, utilizando o Java Carriots SDK. Fornece uma interface amigável que permite o registro de novos dispositivos. O Carriots (i) permite coletar e armazenar qualquer tipo de dados de seus dispositivos; (ii) possibilita construir poderosos aplicativos com o mecanismo SDK que oferece; e (iii) permite a conexão de qualquer dispositivo, sendo necessário somente estar com acesso à Internet. Por outro lado, não contém o requisito de adaptação dinâmica; e não especifica se pode ser reutilizado e se possui ferramentas de desenvolvimento (CARRIOTS, 2018).

[1] É um protocolo criado devido a necessidade de um protocolo capaz de propiciar a troca de informações em um ambiente distribuído de forma descentralizada. Ele está relacionado ao uso do XML no processo de troca de mensagens, buscando como referência os protocolos distintos da camada de aplicação, em especial o RPC e o HTTP que possibilitam a negociação e transmissão dos arquivos XML (SHELBY, 2010).

[2] É um protocolo de mensagens *publish/subscribe* projetado para trabalhar com dispositivos restritos. Está sendo amplamente usado para entrega de dados em aplicativos IoT. Porém, esse protocolo apresenta suporte de segurança limitado (AL-FUQAHA et al., 2015).

3.3 City-Hub

É uma plataforma móvel que proporciona soluções inteligentes, auxiliando no gerenciamento e acesso ao fluxo da cidade. Pode ser aplicado por diferentes partes interessadas, como a própria cidade, cidadãos ou empresas que fornecem serviços ao cidadão de modo geral. Essa plataforma proporciona aos interessados a facilidade em desenvolver e implantar suas soluções inteligentes para a cidade. Contém uma infraestrutura de nuvem disponível para os desenvolvedores, abrangendo todos os serviços ali oferecidos (LEA; BLACKSTOCK, 2014; HEDDEBAUT; CIOMMO, 2018).

City-Hub foi desenvolvido para proporcionar a interoperabilidade entre sistemas heterogêneos, atuando diretamente na solução desta demanda. Desta forma, torna-se possível o reaproveitamento de sistemas e aplicações que atuem em diferentes cidades. Esse *middleware* consegue realizar tal demanda devido a possibilidade de abstrair nuvens híbridas possibilitando a interoperabilidade dos sistemas (PIRES et al., 2015). É uma plataforma essencial para projetos de CI colaborando para a melhora da qualidade da cidade. Ela contém os requisitos de conectividade, escalabilidade, flexibilidade e interoperabilidade (READIY; GUNASEKARAN; SPALANZANI, 2015). Porém, contém uma grande desvantagem, não consegue gerenciar os dados contidos em tempo real, oriundos dos sistemas de alta confiabilidade que devido a suas características de produção de dados em tempo real passa a oferecer grande volume de dados. Outrossim, não oferece segurança e privacidade dos dados (LEA; BLACKSTOCK, 2014; HEDDEBAUT; CIOMMO, 2018).

3.4 Ebbits

O *Enabling Bussiness-Based Internet of Things and Services* (EBBITS) é um *middleware* IoT financiado pela Comissão Europeia para ambientes agrícolas e industriais. O EBBITS foi uma evolução do *middleware* LinkSmart (BRIZZI et al., 2013). Esse projeto visa desenvolver arquitetura, tecnologias e processos que permitem às empresas integrar semanticamente a Internet das Coisas em sistemas empresariais tradicionais (KOSTELNIK; SARNOVSK; FURDIK, 2011). Essa plataforma é baseada em uma arquitetura orientada a serviços para automatização e monitoramento de processos, que dá suporte aos aplicativos interoperáveis oferecendo ambientes mais dinâmicos e interativos. Além de ser interoperável, ele é reutilizável, escalável e flexível (VAJDA et al., 2011).

O EBBITS define os recursos de rede que permitem que ela se conecte com o mundo, por meio de uma grande variedade de dispositivos físicos, etiquetas RFID, sensores e atuadores com uma infraestrutura de comunicações aberta, permitindo que sejam identificados, ou configurados de forma automática (BRIZZI et al., 2013). Porém, essa comunicação entre os dispositivos pode degradar ruídos, afetando a comunicação em uma rede sem fio (VAJDA et al., 2011). Outrossim, pode ocorrer perda de pacotes ou até mesmo parar de funcionar. Esse *middleware* aprimora essa

comunicação, possibilitando maior disponibilidade e confiabilidade (TELES, 2018).

A arquitetura assegura a conectividade dos dispositivos e a entrega confiável dos dados sem ruídos por meio de troca dinâmica de canais, gerenciamento de multi rádio e redes tolerantes a atraso. As desvantagens dessa plataforma são (i) somente usuários especialistas estão aptos a utilizar essa plataforma; (ii) a ferramenta é voltada somente para área comercial; e (iii) não permite ao usuário inserir novas funcionalidades com a finalidade de criar novas soluções de acordo com suas necessidades (BRIZZI et al., 2013; CONZON et al., 2015; KHALEEL et al., 2017).

Essa plataforma permite as seguintes características inovadoras, (i) sensores e redes do mundo físico, (ii) virtualização de entidade, (iii) Gerenciamento de Dados e eventos, (iv) inteligência centrada e distribuída, (v) infraestrutura de conhecimento semântico, e (vi) *Frameworks* para Gerenciamento do Ciclo de Vida do Processo de Negócio incluindo métricas e otimização de processos (BRIZZI et al., 2013; CONZON et al., 2015).

3.5 EcoDiF

É uma plataforma *Web* aberta para ecossistema de dispositivos físicos, que representa o núcleo de um ecossistema para IoT. Ela estabelece a interoperabilidade entre os diferentes dispositivos com os usuários e/ou aplicações, conectados na Internet, permitindo o desenvolvimento de novas soluções e/ou produtos atendendo as necessidades dos especialistas. Outrossim, possibilita serviços de aplicações, serviços de apoio, controle de dados em tempo real, visualização, processamento e armazenamento de dados provenientes desses dispositivos, dentre outros (PIRES et al., 2014).

A arquitetura e implementação dessa plataforma é baseada em *Representational State Transfer* (REST)[3] e se baseia em padrões e protocolos da *Web*, como HTTP e *Uniform Resource Identifier* (URI). O HTTP além de ser utilizado para comunicação, também é usado para suportar todas as interações com os diferentes dispositivos. Essas interações ocorrem por meio das principais operações HTTP (por exemplo, *GET*, *POST*, *PUT* e *DELETE*), que fornecem uma interface bem definida para expor a funcionalidade dos objetos na *Web*. Essa interface está em conformidade com os princípios REST, permitindo assim que os dispositivos sejam acessados por meio da Internet e seus dados ou serviços sejam (re)utilizados em diferentes aplicações com qualquer recurso da *Web* (DELICATO et al., 2013; PIRES et al., 2015).

Com a utilização do REST, essa plataforma padroniza e facilita o desenvolvimento de novas soluções. Outrossim, reduz a compatibilidade entre diferentes dispositivos, formato dos dados e protocolos proprietários (PAUTASSO; ZIMMERMANN; LEYMANN, 2008). A EcoDiF

[3] É um estilo de arquitetura abrangente para aplicativos descentralizados. Essa arquitetura utiliza o protocolo HTTP e suas operações disponíveis, tais como: *GET*, *POST*, *PUT* e *DELETE*. Esse protocolo é utilizado para desenvolver *middlewares* IoT (CATARINUCCI et al., 2015).

proporciona uma solução convergente de acesso consolidado e serviços de alto nível, assim como permite que os especialistas colaborem de maneira mais fácil no desenvolvimento de novas aplicações para IoT (PIRES et al., 2015).

Essa plataforma possui vantagens, tais como flexibilidade, conectividade, interoperabilidade, portabilidade, suporta grande volume de dados, e reusabilidade. No entanto, permite que somente os usuários com conhecimento técnico desenvolvam soluções, atendendo diferentes domínios e dispositivos. Contém desafios com usabilidade e escalabilidade e não faz o uso de interfaces gráficas que auxiliem os usuários não especialistas a adaptar suas soluções com as suas necessidades (PIRES et al., 2015).

3.6 GSN

O *Global Sensor Network* (GSN) é um *middleware* IoT que oferece uma plataforma genérica para implantar redes de sensores e processar dados produzidos pela rede de sensores de maneira distribuída (PERERA et al., 2014; BABU; GEORGE; SAMUEL, 2016). Essa plataforma oferece uma implementação flexível e uma integração de diferentes sistemas por meio de redes de sensores virtuais em geral (AHSAN et al., 2016). Esses sensores virtuais podem ser lógicos, com abstrações de um ou mais sensores reais, que capture dados. Eles também podem ser agregadores ou filtros aplicados a outros sensores virtuais, que pode ser implantados localmente ou remotamente (CALBIMONTE et al., 2014).

Esse *middleware* fornece a capacidade de integrar, descobrir e consultar dados dos sensores por meio da linguagem baseada em XML (PERERA et al., 2013). Ele adota a arquitetura baseada em *containers*, que as redes de sensores podem ser facilmente implementados ou (re)configurados em tempo de execução. Esses *containers* têm como objetivo oferecer serviços às demais partes da arquitetura, mesmo que não suporte tais serviços. A comunicação da GSN é realizada em *peer-to-peer* (P2P), protocolo padrão da Internet e da *Web* (ABERER; HAUSWIRTH; SALEHI, 2007).

O GSN (i) permite que os aplicativos interoperem grandes quantidades de dados usando fluxos em tempo real de praticamente qualquer lugar, com qualquer tecnologia de *hardware* e *software*; (ii) oferece roteamento, agregação de dados e energia de otimização, além da tradicional rede de sensores; (iii) pode ser aplicado em ambientes heterogêneos; (iv) utilizado para diferentes aplicabilidades em IoT; (v) possibilita que os usuários manipulem criando novas soluções; e (vi) escalável. Por outro lado, o GSN continha interfaces que possibilitavam aos usuários adaptarem a sua realidade (ABERER; HAUSWIRTH; SALEHI, 2007; TU'N et al., 2012; PERERA et al., 2013; PERERA et al., 2013; BABU; GEORGE; SAMUEL, 2016).

3.7 Kaa

É uma plataforma *middleware* IoT aberta que introduz métodos padronizados que permitem a interoperabilidade entre os dispositivos de qualquer tipo (KERSTING et al., 2017). Ela é capaz de conectar e gerenciar qualquer dispositivo por meio de um acesso remoto, além de coletar e analisar dados em tempo real e ser implantado em qualquer lugar (HUACARPUMA et al., 2017). Também, oferece soluções IoT inteligentes para os setores da saúde, energia, logística, indústria, agricultura, lazer, automobilístico, dentre outros. Por fim, fornece uma interface para armazenar dados de dispositivos e interfaces da IoT para usar dados de aplicativos e outros sistemas (KAA, 2018).

A comunicação realizada pelo Kaa é por meio da utilização do SDK suportados nas linguagens de programação Java, C e C++, que produzem aplicações que podem ser incorporadas a dispositivos (KERSTING et al., 2017). As aplicações desenvolvidas nos SDKs são incorporadas ao dispositivo conectado e implementa a troca de dados em tempo real e bidirecional com o servidor (MALCHE; MAHESHWARY, 2015; PFLANZNER; KERTÉSZ, 2016).

A plataforma Kaa promove a possibilidade de comunicar, controlar e monitorar dispositivos heterogêneos, por meio de uma estrutura escalável, em relação aos dispositivos conectados. Por outro lado, é possível abstrair detalhes da comunicação realizada entre outras ferramentas, o que facilita na análise dos dados provenientes dessa integração onde há grande amplitude na diversidade dos dados. Como vantagens é uma (i) plataforma segura e confiável para o desenvolvimento de soluções IoT; (ii) garante a segurança e privacidade dos dados; (iii) escalabilidade; (iv) portabilidade; e (v) é flexível. Como desvantagem não possui grande utilização no correr do tempo (KIM-HUNG et al., 2017; MADUKWE; EZIKA; ILOANUSI, 2018).

3.8 LinkSmart

LinkSmart, anteriormente chamada de HYDRA, é uma plataforma *middleware* IoT completa que baseia-se em uma arquitetura orientada a serviços. Essa arquitetura proporciona aos desenvolvedores de *software* um conjunto de componentes para o gerenciamento de dispositivos e uma infraestrutura para acessar esses dispositivos. A plataforma fornece recursos garantindo a interoperabilidade entre os dispositivos heterogêneos por meio de uma rede *peer-to-peer* (P2P). Outrossim, possibilita comunicações seguras e confiáveis, permitindo operações criptografadas para proteção de mensagens a fim de garantir a confidencialidade entre as partes (PALADE et al., 2017; LINKSMART, 2018).

Esse *middleware* é recomendado para soluções inteligentes em sistemas embarcados, porém atualmente é aplicado no desenvolvimento de casas inteligentes, agricultura e energia (KOSTELNIK; SARNOVSK; FURDIK, 2011). O LinkSmart apresenta uma solução voltada às demandas dos usuários que possuem conhecimento do processo, ou seja, aqueles que detém

domínio teórico e prático de uma área específica, sendo assim o suporte a usuários não especialistas se torna deficitário. Desse modo, mesmo ao utilizar um modelo semântico que propicia flexibilidade, o LinkSmart possui restrições em relação aos usuários que o utilizam (PATTI et al., 2016).

A plataforma fornece finalidades importantes para os ambientes IoT, tais como conectividade, descoberta e gerenciamento de dispositivos, flexibilidade, interoperabilidade, portabilidade, reusabilidade, usabilidade, e segurança e privacidade. Outrossim, suporta uma grande variedade de dispositivos que podem ser controlados por uma interface de comunicação, como *Bluetooth*, ZigBee e RFID (RAZZAQUE et al., 2016; PALADE et al., 2017; TELES, 2018).

3.9 openHAB

Open Home Automation Bus (OpenHAB) é um *middleware* desenvolvido para interoperar sistemas heterogêneos na solução para todos os aspectos relacionados à automação residencial, como aquecedores, luzes, dentre outros (OPENHAB, 2018). Por ser interoperável, ele consegue se comunicar e gerenciar diferentes sistemas que utilizam protocolos como *Bluetooth*, Wi-Fi, MQTT, TCP/UDP, dentre outros (HEIMGAERTNER et al., 2017). É uma plataforma *open source*, desenvolvida em Java baseada no *framework Eclipse SmartHome*, oferecendo aos usuários uma interface gráfica para administrar seu ambiente residencial (SALIHBEGOVIC et al., 2015). Contém uma arquitetura modular baseada em *Open Service Gateway Initiative* (OSGi)[4], proporcionando administrar o ambiente em tempo de execução, sem prejudicar os demais serviços designados pela arquitetura (SEIGER; HUBER; SCHLEGEL, 2018; HEIMGAERTNER et al., 2017).

Esse *middleware* contém uma versão em Android, que o *smartphone* por meio da sua rede Wi-Fi, se conecta no servidor do OpenHAB utilizando a interface RESTful. Pode ser executado em qualquer dispositivo que seja capaz de executar uma *Java Virtual Machine* (JVM), como por exemplo *Linux*, Mac e *Windows*. A automação residencial é baseada em eventos, regras, *Scripts* e ações. Um evento pode ser caracterizado como a mudança de um determinado estado para outro, podendo envolver diferentes tipos de dados, portanto é necessário executar algoritmos reativos a eventos. Esses algoritmos devem apresentar cunho genérico para que possam permitir o reuso de códigos em cenários diferentes. Desse modo, estes algoritmos podem ser elaborados por uma linguagem própria de *Scripts* que utilizam parâmetros similares à linguagem de programação Java (RAMLJAK, 2017). Um evento demanda uma ação que é traduzida em métodos Java que podem ser utilizados pelos *Scripts* que foram desenvolvidos orientados a eventos (SMIREK; ZIMMERMANN; ZIEGLER, 2014).

O openHAB (i) atende a todas as necessidades de automação; (ii) fornece APIs para

[4] Fornece um ambiente para modularizar aplicativos da *Web* em pacotes configuráveis, que podem registrar dinamicamente no contexto de execução do pacote (ZHONG; ZHAO; DAI, 2014; JI et al., 2014).

ser integrado em outros sistemas; (iii) contém interfaces para iOS e Android; (iv) possibilita o controle do ambiente por meio de regras de negócios; (v) permite a interoperabilidade de diferentes tecnologias; e (vi) pode integrar diferentes tecnologias de automação residencial em uma só. Porém, ele só pode ser utilizado em ambientes domésticos (MICLAUS; RIEDEL; BEIGL, 2014; OPENHAB, 2018).

3.10 OpenIoT

OpenIoT é uma plataforma *middleware* IoT aberta fundada pela Comissão Europeia, permitindo a integração e descoberta de diferentes dispositivos IoT conectados à Internet e serviços *Web*. Essa integração ocorre por meio de uma interface gráfica amigável, trabalhando em ambientes *Cloud Computing* ou servidor local. É totalmente descentralizada, além de permitir que o usuário solicite e processe os dados conforme necessário para fornecer serviços IoT. A infraestrutura desse *middleware* suporta implementação de algoritmo para coletar e filtrar as informações necessárias dos objetos conectados à Internet (KIM; LEE, 2014; MOHAMMED; KIRAN, 2015).

Esse projeto explora maneiras eficientes de usar e gerenciar ambientes de nuvem para entidades e recursos de IoT, como sensores, atuadores e dispositivos inteligentes. Permite obter facilmente informações sobre nuvens de sensores e conecta essas informações a serviços da *Web* sem se preocupar com exatamente quais sensores diferentes estão sendo usados. Proporciona escalabilidade das redes de sensores e a adição de novos sensores flexíveis, fornecendo um *middleware* essencial para aplicativos IoT. Os usuários podem configurar, gerenciar e usar a nuvem de sensores (SERRANO et al., 2015).

O *middleware* está divido em três camadas importantes, que são as de aplicação, virtualização e física. A primeira camada proporciona o gerenciamento de dispositivos e a integração de serviços de terceiros para a análise de dados. Na segunda camada, efetua as requisições de serviços assegurando segurança nos recursos e armazenando os dados. Por fim, a última camada irá filtrar todos os dados essenciais dos dispositivos (GEORGAKOPOULOS; JAYARAMAN, 2016). O OpenIoT tem a capacidade de suportar implementações em grande escala, além de ser flexível e permitir a segurança. Porém, é somente uma solução essencial para ambientes industriais e empresas, não podendo ser reutilizável.

3.11 RestThing

RestThing é *middleware* fortemente ligado a IoT, assim como o EcoDiF, é um servidor *Web Service* que permite ocultar a interoperabilidade entre sistemas heterogêneos conectados na Internet manipulando por meio das requisições REST (PIRES et al., 2015). O *middleware* permite que os usuários gerenciem suas aplicações, além de proporcionar a criação das suas soluções

para a integração por meio da Internet. A troca dos dados por meio do REST é realizada pelo protocolo HTTP e seus métodos (*GET*, *POST*, *PUT* e *DELETE*). Outro importante identificador de recurso universal utilizado pela API RESTful e a *Web* é a URI. Esse identificador único de recurso é fundamental para permanecer o acesso das informações na *Web* (QIN et al., 2011).

Esse *middleware* suporta três formatos, como por exemplo: JSON, XML e *Comma-Separated Values* (CSV). Contém uma arquitetura com 4 camadas: aplicação, provedor de serviços, adaptação e dispositivos embutidos. A primeira camada refere-se a todos os dispositivos ou aplicações que irão se integrar ou colaborar auxiliando na criação de soluções inteligentes. A segunda camada é o núcleo para o acesso do *middleware*, acessada pela API RESTful. Ainda nessa camada, os dados são coletados pela camada de adaptação. Essa camada irá transferir os dados obtidos pela camada de dispositivos embarcados, que irá ser integrado na API RESTful. A última camada pode corresponder aos *softwares* (redes sociais, dentre outros) e *hardware* (sensores, dentre outros) (QIN et al., 2011).

A principal vantagem contida pelo *middleware* RestThing é a capacidade de ocultar a diferença dos dispositivos, proporcionando a interoperabilidade dos mesmos, ainda que estes possuam interfaces de comunicação distintas. Em contrapartida, não é escalável; não garante a segurança e privacidade dos dados, e nem permite adaptação dinâmica.

3.12 S^{3}OIA

Smart Spaces and Smart Objects Interoperability Architecture (S^{3}OIA) é uma arquitetura IoT distribuída que gerencia diferentes tipos de dados ou objetos e trata a interoperabilidade de qualquer dispositivo entre grandes sistemas heterogêneos. Os 3S significam que é uma arquitetura semântica, sintática e SOA, arquitetura orientada a serviço. A integração entre os objetos é utilizada pelo OSGi, propondo unificar diferentes protocolos de gerenciamento e descoberta de dispositivos. Essa arquitetura cria um novo protocolo comum padronizado abstraindo as inconsistências e utiliza o padrão RESTful (VEGA-BARBAS et al., 2012).

Essa arquitetura descentralizada contém 3 níveis: física, serviços e virtual. O nível físico é o nível em que se encontra todos os dispositivos. Já o segundo nível proporciona a descoberta dos dispositivos desconsiderando o fornecimento dos seus serviços. Por fim, o último nível apresenta a cooperação das soluções inteligentes (VEGA-BARBAS et al., 2012).

S^{3}OIA contém os requisitos de interoperabilidade, adaptação dinâmica, ciência de contexto e conectividade. No entanto, não contém requisitos importantes de um *middleware*, tais como tratamento de grande volume de dados, escalabilidade e segurança e privacidade das informações.

3.13 SIRENA

Service Infrastructure for Real-time Embedded Networked Applications (SIRENA) é um serviço de infraestrutura aberta para a integração de dispositivos IoT em tempo real para soluções industriais. Foi projetado pela Comissão Europeia em 2003 e desenvolvido em Java. Em 2005, esse serviço foi demonstrado na *Microsoft* lançando uma versão para o *Windows* Vista. Essa infraestrutura utiliza uma arquitetura orientada a serviços para integrar com os dispositivos heterogêneos que contém restrições de recursos (BOHN; BOBEK; GOLATOWSKI, 2006).

Identifica-se diversos *middlewares* que são orientados a serviços, ou seja, que podem ser dotados da tecnologia SIRENA, tais como OSGi, *Universal Plug and Play* (UPnP)[5] e *Devices Profile for Web Services* (DPWS)[6]. O SIRENA utiliza os componentes RFID, *Bluetooth* e Sensores. OSGi estabelece uma plataforma de serviço que os usuários implantam e desenvolvem suas soluções. Com o DPWS, os dispositivos irão detectar automaticamente a presença de outros dispositivos inteligentes. Porém, quando existir diferentes dispositivos do mesmo tipo, haverá a necessidade de configurá-los. O UPnP é uma arquitetura de serviços simples que utiliza o protocolo IPv4. Logo, ele não é recomendado para uma grande quantidade de dispositivos (JAMMES; SMIT, 2005).

O SIRENA proporciona adaptação dinâmica e conectividade, possibilita a descoberta e gerenciamento dos dispositivos, é flexível, permite a interoperabilidade entre os dispositivos inteligentes, e oferece a portabilidade. Em contrapartida, só pode ser utilizado no setor industrial, automatização, telecomunicação e residencial. Outrossim, não oferece segurança e privacidade dos dados, e contém desafios com escalabilidade.

3.14 SM4All

O *Smart Home For All* (SM4All) é uma plataforma *middleware* inovadora projetado pela Comissão Europeia em 2008, contendo serviços integrados inteligentes interoperando em ambientes imersos, focando em casas inteligentes. Esse *middleware* gerencia os dispositivos oferecendo uma configuração dinâmica e estruturação dos serviços, assegurando a privacidade. Atende as necessidades dos usuários oferecendo uma infraestrutura altamente interativa e adaptável às necessidades de diferentes usuários e ambiente, melhorando o conforto (KALDELI et al., 2013a).

Essa plataforma está dividida em 3 camadas: camada de usuário, camada de composição e camada física. A primeira camada oferece interfaces gráficas para seus usuários conforme

[5] É um protocolo de conectividade mundial que permite diferentes tipos de dispositivos de rede cooperem e comunicar-se com outro dispositivo de rede de maneira flexível usando o recurso encontrado na rede (SANTOS; ALMEIDA; PERKUSICH, 2015).

[6] É um protocolo de Serviços da *Web*, que contém um conjunto de restrições de implementação para habilitar serviços seguros de mensagens, descoberta, descrição e eventos em dispositivos com recursos limitados (HAN et al., 2015).

as suas necessidades. Já a segunda camada administra os dispositivos e coleta as informações necessárias, satisfazendo os objetivos desejados pelos usuários. Por fim, última camada com infraestrutura que possibilita a descoberta automática e conexão de dispositivos com diferentes protocolos de rede, como por exemplo, *Bluetooth*, ZigBee, dentre outros. Ela também utiliza os padrões UPnP e OSGi (WARRIACH et al., 2011).

O SM4All (i) permite adaptação dinâmica; (ii) suporta grande volume de dados; (iii) conectividade; (iv) proporciona a descoberta e gerenciamento dos dispositivos; (v) é escalável; (vi) é flexível; (vii) é reutilizável; (viii) oferece segurança e privacidade nas informações; e (ix) oferece interfaces atendendo as necessidades de diferentes usuários. Por outro lado, essa plataforma foi descontinuada em 2011.

3.15 SOCRADES

Service-Oriented Cross-Layer Infrastructure for Distribued smart Embedded devices (SOCRADES) é uma plataforma *middleware* IoT criado pela Comissão Europeia. Essa plataforma é orientada a serviços para rede incorporada e contém o principal objetivo de desenvolver soluções inteligentes para a integração dos dispositivos, proporcionando uma conectividade confiável (SOUZA et al., 2008). A integração visa promover benefícios para as tecnologias existentes por meio do uso do DPWS (CANNATA; GEROSA; TAISCH, 2008).

Esse projeto é semelhante ao LinkSmart, podendo ser aplicado na área de automação residencial além da automação industrial (JAHN et al., 2010). Uma alternativa viável para implementar IoT em industrias é dada por meio do uso de uma camada xMII, possibilitando a interoperabilidade de diferentes tipos de dispositivos que integram a regra de negócio da empresa. Esse *middleware* foca na conexão dos dispositivos para satisfazer os requisitos de heterogeneidade, gerenciamento de falhas, adaptação, plataforma independente e composição de serviços (CANNATA; GEROSA; TAISCH, 2008).

O SOCRADES é uma interessante solução IoT para ambientes industriais, contendo os requisitos de adaptação dinâmica, tratamento de grande volume de dados, conectividade, escalabilidade, flexibilidade, portabilidade, reusabilidade, usabilidade, interoperabilidade, e segurança e privacidade aos usuários. Portanto, é uma plataforma que permite apenas aos usuários especialistas criar novas soluções adequando as suas necessidades.

3.16 *Task Computing*

Task Computing, também chamado de *Many-Task Computing* (MTC), é um *middleware* para soluções inteligentes que atua de modo a criar uma interação entre objetos e serviços heterogêneos por meio da criação de tarefas personalizadas para cada usuário. Permite a criação de fluxos de tarefas por meio de uma interface gráfica altamente amigável. A fim de dinamizar

a interface gráfica o *middleware* tenta descobrir os serviços disponibilizados na rede onde cria objetos que modificam a interface com o usuário e criam fluxos de trabalho lógico. Por fim, permite que o usuário combine serviços, sejam eles internos ou externos (RAICU; FOSTER; ZHAO, 2008).

O MTC visa permitir o monitoramento e execução de tarefas por meio da composição de serviços em uma interface gráfica amigável aos usuários. Esta plataforma não se limita a mera interface gráfica, esta oferece ferramentas para possibilitar a implementação de um ambiente responsivo a tecnologias IoT por meio do processo da busca automática de novos serviços e dispositivos, além de extrair características que permitem melhor nomeação e usabilidade. De fato, esse *middleware* apresenta soluções úteis, mas que podem ainda ser melhoradas, como a criação de um modelo que permita o usuário criar novas tarefas de forma mais realística com as suas necessidades, como a descrição comportamental de um ambiente que analisa tarefas predeterminadas pelo usuário por meio de um modelo conceitual. A implementação desse modelo pode permitir ao usuário a criação de regra de modo que a execução siga uma lógica baseada em fluxos de dados necessários para regra de negócios utilizadas pelo usuário (TELES, 2018).

3.17 UbiSOAP

UbiSOAP é um *middleware* orientado a serviços para redes ubíquas. Foi elaborado pela Comissão Europeia com o objetivo de dispor de redes sem fio para capacitar sua infraestrutura de comunicação. Essa comunicação ocorre por meio do protocolo SOAP, suportando serviços *Web* legados, à medida que observa a sua conectividade ubíqua. A conectividade pode ocorrer por meio de tecnologias sem fio, como *Bluetooth*, Wi-Fi, dentre outros. Também fornece uma rede de serviços *Web* que podem ser implantados em diferentes dispositivos, incluindo dispositivos móveis (CAPORUSCIO; RAVERDY; ISSARNY, 2012).

A sua arquitetura está dividida em duas camadas, que estabelecem a conectividade independente de rede explorando todas as redes existentes em uma única vez. A primeira camada é a de conectividade, que os usuários estabelecem suas políticas para a utilização de uma determinada rede. Já a segunda camada é a de comunicação, que utiliza o protocolo SOAP para enviar mensagens para todos os serviços, integrando componentes heterogêneos dentro do ambiente distribuído. A UbiSOAP propõe que esses dispositivos funcionem via multi-rádio, e que os servidores *proxy* funcionem como pontes para uma integração essencial destes dispositivos. (CAPORUSCIO et al., 2008).

Esse *middleware* é uma interessante solução IoT para os usuários que necessitam de serviços em redes *Wireless* e precisam da integração de redes sem fio heterogêneas, utilizando a computação ubíqua. A principal vantagem é gerenciar os dispositivos permitindo a conectividade entre eles. Outras vantagens podem ser destacadas, como possibilita a portabilidade, é escalável e possui adaptação dinâmica. Porém, a grande desvantagem é que não oferece segurança e

privacidade dos dados (BANDYOPADHYAY et al., 2011).

3.18 Ubiware

Ubiware é uma plataforma *middleware* baseada na arquitetura orientada a serviço. É utilizada para o desenvolvimento de novas soluções IoT de sistemas industriais complexos que requerem distribuição, compartilhamento e componentes reutilizáveis (NIKITIN; KATASONOV; TERZIYAN, 2009). A arquitetura requer uma plataforma central confiável, que forneça meios para a construção de sistemas flexíveis, solucionando diferentes cenários e adaptando as necessidades dos usuários ou ambientes (KATASONOV et al., 2008). Ele é baseado nos requisitos de automação, gerenciamento de dados, descoberta de dispositivos, integração, interoperabilidade, dentre outros (PALADE et al., 2017). Outrossim, o Ubiware contém multi-agentes, entidades computacionais autônomas que gerenciam o desenvolvimento de sistemas complexos e executam os seus principais requisitos (NAGY et al., 2009).

Esse *middleware* contém objetos conectados a um agente de *software*, proporcionando aos usuários um conhecimento sobre o estado de cada dispositivo, possibilitando-os a serem inteligentes. O usuário terá o controle do ambiente fornecido pelos agentes e os objetos ganharão recursos de comunicação, autocontrole e auto-monitoramento. Diante do fato de cada objeto estar conectado a um agente, pressupõe-se que o mesmo tem o devido conhecimento em relação ao seu estado. Desta forma, os agentes podem trocar informações entre si, melhorando a execução de seus objetos e explorando os dados obtidos a fim de encontrar as informações relevantes para sua utilização (PIRES et al., 2015).

Essa plataforma foi implementada no *Java Agent Develpment Framework* (JADE). Ubiware (i) permite ser reutilizável; (ii) proporciona a interoperabilidade; (iii) possibilita que seja modificada atendendo as necessidades dos usuários e de diferentes domínios; (iv) propicia ambientes inteligentes. Por outro lado, somente usuários com grau elevado de conhecimento conseguem manusear essa plataforma podendo adequar ou expandir novas soluções propostas no Ubiware (PALADE et al., 2017; TELES, 2018).

3.19 *Middlewares versus* Requisitos

Considerando a complexidade aplicada no desenvolvimento das plataformas de *middlewares*, se faz necessário verificar quais requisitos são atendidos por essas plataformas presentes no mercado. Portanto, para apresentar de forma mais coesa e direta, foi elaborada as Tabelas 3, 4 e 5. Essas tabelas foram criadas para confrontar as plataformas *middlewares* que estão sendo mais utilizadas no mercado com os requisitos necessários, verificando se os mesmos suportam tais requisitos. Essas tabelas foram refeitas do trabalho de Teles (2018), adicionando novos requisitos e plataformas. Pode-se observar que três novos requisitos foram adicionados, como

Adaptação Dinâmica, Ferramentas de Desenvolvimento e Interfaces. Outrossim, também foram adicionados oito *middlewares*/plataformas, como Carriots, City-Hub, FIWARE, Kaa, OpenIoT, S^3OIA, SIRENA e UbiSOAP.

Esta tabela possui uma coluna indicando as dezenove "**plataformas ou *middlewares***" que estão sendo avaliadas. A coluna "**requisitos**" apresenta os quatorze requisitos IoT que se caracterizam como sendo relevantes no desenvolvimento de novas aplicações, que são adaptação dinâmica, *big data*, ciência de contexto, conectividade, descoberta e gerenciamento de dispositivos, escalabilidade, ferramentas de desenvolvimento, flexibilidade, interfaces, interoperabilidade, portabilidade, reusabilidade, segurança e privacidade e usabilidade. O campo "**referência**" apresenta quais são as fontes que foram utilizadas para verificar até que ponto os requisitos haviam sido atendidos. Cada requisito foi avaliado com base nos quatro critérios (●) requisito completamente atendido (◗) requisito parcialmente atendido, (○) requisito não atendido e (⊗) requisito não especificado.

De acordo com as Tabelas 3, 4 e 5 constata-se que existem *middlewares* sendo utilizados para CI ou IoT (BATISTA et al., 2016; GUTH et al., 2016; PALADE et al., 2017). Dentre os quatorze requisitos, a interoperabilidade e a conectividade foram os únicos atendidos por todas as plataforma. Já os demais foram insuficientemente abordados. Diante do exposto, verifica-se que dentre as dezenove plataformas, somente a FIWARE contempla a todos os requisitos elencados. As demais plataformas encontram-se em estado inicial, devido a ausência de requisitos importantes que não estão sendo contemplados.

Os principais requisitos devem ser contemplados nos projetos/soluções *middlewares*, tais como interoperabilidade, portabilidade, usabilidade, privacidade e segurança, conectividade e escalabilidade (TELES, 2018). As plataformas FIWARE, OpenIoT, SOCRADES, MTC e ASPIRE contemplam esses principais requisitos. Dentre essas cinco plataformas, a plataforma FIWARE é a única que suporta tecnologias com dispositivos inteligentes domésticos[7] e dispositivos legados[8]. Portanto, é possível concluir que a FIWARE é a plataforma que engloba todos os requisitos necessários para a criação de soluções IoT.

[7] São dispositivos que se conectam/comunicam com os serviços pela Internet, como por exemplo *Smartphone*, *Smart* TV, câmeras, dentre outros (PERERA et al., 2012).

[8] São dispositivos que não se conectam/comunicam com os serviços pela Internet (TELES, 2018).

Tabela 3 – *Middlewares versus* Requisitos (parte 1).

Middlewares/ Plataformas	Requisitos														Referências
	Adaptação Dinâmica	*Big Data*	Ciência de Contexto	Conectividade	Descoberta e Gerenciamento de Dispositivos	Escalabilidade	**Ferramentas de Desenvolvimento**	Flexibilidade	**Interfaces**	Interoperabilidade	Portabilidade	Reusabilidade	Segurança e Privacidade	Usabilidade	
ASPIRE	●	○	⊗	●	●	●	●	●	●	●	●	●	●	●	(PERERA et al., 2015) (TELES, 2018) (ASPIRE, 2018)
Carriots	⊗	●	◗	●	●	●	○	●	●	●	○	○	●	●	(FARAHZADI et al., 2017) (MINERAUD et al., 2016) (CARRIOTS, 2018)
City-Hub	◗	●	◗	●	◗	●	●	●	●	●	○	○	⊗	○	(LEA; BLACKSTOCK, 2014) (CITY-HUB, 2018) (BATISTA et al., 2016)
Ebbits	○	○	●	●	○	●	○	●	●	●	○	●	●	●	(TELES, 2018) (VAJDA et al., 2011)
EcoDiF	⊗	●	◗	●	◗	◗	◗	●	●	●	●	●	●	◗	(TELES, 2018) (DELICATO et al., 2013) (BATISTA et al., 2016)
FIWARE	●	●	●	●	●	●	●	●	●	●	●	●	●	●	(PREVENTIS et al., 2016) (BATISTA et al., 2016) (FARAHZADI et al., 2017) (FAZIO et al., 2015) (SANTANA et al., 2018) (BARRETO et al., 2015) (STRAVOSKOUFOS et al., 2014) (KOVACS et al., 2016)
GSN	●	○	⊗	●	●	●	○	●	●	●	●	●	●	○	(CHAQFEH; MOHAMED, 2012) (TELES, 2018) (PERERA et al., 2015) (AHSAN et al., 2016) (PERERA et al., 2013)
Kaa	◗	●	◗	●	◗	●	●	●	●	●	●	○	●	○	(BATISTA et al., 2016) (KAA, 2018)

Fonte – Autora (2019).

Tabela 4 – *Middlewares versus* Requisitos (parte 2).

Middlewares/ Plataformas	Requisitos														Referências
	Adaptação Dinâmica	*Big Data*	Ciência de Contexto	Conectividade	Descoberta e Gerenciamento de Dispositivos	Escalabilidade	**Ferramentas de Desenvolvimento**	Flexibilidade	**Interfaces**	Interoperabilidade	Portabilidade	Reusabilidade	Segurança e Privacidade	Usabilidade	
LinkSmart	⊗	●	●	●	●	⊗	○	●	⊗	●	●	●	●	●	(PALADE et al., 2017) (TELES, 2018) (KRYLOVSKIY; JAHN; PATTI, 2015)
OpenHAB	○	⊗	⊗	●	●	◗	○	●	●	●	●	●	●	●	(OPENHAB, 2018) (RAMLJAK, 2017) (HEIMGAERTNER et al., 2017) (WERNER; PALLAS; BERMBACH, 2018)
OpenIoT	⊗	●	⊗	●	⊗	●	○	●	●	●	●	○	●	●	(SANIAT et al., 2015) (OPENIOT, 2018)
RestThing	⊗	⊗	◗	●	⊗	⊗	○	○	●	●	●	○	⊗	◗	(VENTURA et al., 2014) (TELES, 2018)
S^{3}OIA	●	⊗	●	●	●	⊗	⊗	○	⊗	●	○	○	⊗	○	(VEGA-BARBAS et al., 2012) (VEGA-BARBAS et al., 2015)
SIRENA	●	○	⊗	●	●	◗	●	●	●	●	●	○	⊗	○	(KYUSAKOV et al., 2013) (PERERA et al., 2015)
SM4All	●	●	●	●	●	●	○	●	●	●	○	●	●	◗	(CATARCI et al., 2009) (AIELLO et al., 2011) (LI, 2013) (KALDELI et al., 2013b) (TELES, 2018)
SOCRADES	●	●	⊗	●	●	●	○	●	●	●	●	●	●	●	(PERERA et al., 2015) (TELES, 2018)
MTC	●	●	⊗	●	●	●	○	●	●	●	●	●	●	●	(RAICU; FOSTER; ZHAO, 2008) (TELES, 2018) (SADOOGHI et al., 2014) (VALERO-LARA et al., 2016)

Fonte – Autora (2019).

Tabela 5 – *Middlewares versus* Requisitos (parte 3).

Middlewares/ Plataformas	Requisitos													Referências
	Adaptação Dinâmica	*Big Data*	Ciência de Contexto	Conectividade	Descoberta e Gerenciamento de Dispositivos	Escalabilidade	**Ferramentas de Desenvolvimento**	Flexibilidade	**Interfaces**	Interoperabilidade	Portabilidade	Reusabilidade	Segurança e Privacidade	Usabilidade	
UbiSOAP	●	●	⊗	●	●	●	●	○	●	●	●	○	⊗	○	(PERERA et al., 2015) (BANDYOPADHYAY et al., 2011) (MOHAMED; AL-JAROODI, 2011) (CAPORUSCIO; RAVERDY; ISSARNY, 2012)
Ubiware	⊗	●	⊗	●	●	●	○	○	●	●	●	○	⊗	●	(BATISTA et al., 2016) (PALADE et al., 2017) (BANDYOPADHYAY et al., 2011) (SCUTURICI et al., 2012) (PERERA et al., 2015)

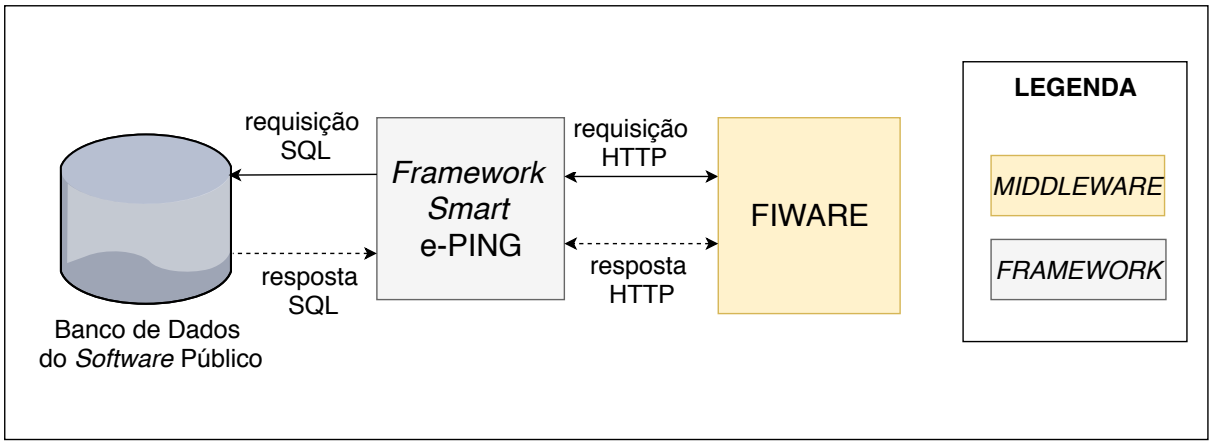
Fonte – Autora (2019).

4

Framework Smart e-PING

A fim de contribuir para o GEB, o presente trabalho propôs o *Framework Smart* e-PING. O objetivo do *Framework Smart* e-PING é prover a interoperabilidade entre um *Software* Público que adota a arquitetura e-PING e a plataforma FIWARE. Ademais, o *Framework Smart* e-PING possibilita que o *Software* Público e a FIWARE interajam de maneira rápida, uma vez que a arquitetura do *Framework Smart* e-PING abstrai as camadas inferiores responsáveis pelo fornecimento dos seus serviços. Portanto, o usuário não necessita de prévio conhecimento, restando somente entender a interface do *Framework Smart* e-PING para efetuar o seu uso.

Figura 7 – Visão da comunicação do *Framework Smart* e-PING.



Fonte – Autora (2019).

De forma resumida, a Figura 7 apresenta uma visão da comunicação de como o *Framework Smart* e-PING interage com o *Software* Público e a FIWARE. O *Framework Smart* e-PING foi desenvolvido na linguagem PHP e a sua comunicação está dividida em três camadas. A primeira camada corresponde ao Banco de Dados do *Software* Público, que é responsável por disponibilizar as informações a serem utilizadas por quaisquer aplicações e/ou *Web Services*,

que necessitem dessas informações. A segunda camada é o próprio *Framework Smart* e-PING, que é responsável pelo gerenciamento da interoperabilidade entre as demais camadas. Por fim, a terceira camada corresponde à plataforma FIWARE, que é representada pelo *Generic Enable* ORION. Outrossim, a FIWARE recebe requisições enviadas pelas aplicações e/ou *Web Services* que necessitem das informações provenientes do *Software* Público.

Conforme a Figura 7, a comunicação entre o *Framework Smart* e-PING e o *Software* Público é realizada por meio de requisições SQL (*SELECT*, *UPDATE*, *DELETE* ou *INSERT*) no Banco de Dados, com envio e resposta de arquivos no formato SQL. Por outro lado, a comunicação entre o *Framework Smart* e-PING e a FIWARE é realizada por meio de requisições HTTP (*GET*, *POST*, *DELETE* ou *UPDATE*). As requisições HTTP tem como objetivo enviar e receber arquivos no formato *JavaScript Object Notation* (JSON), formato aceito pela FIWARE.

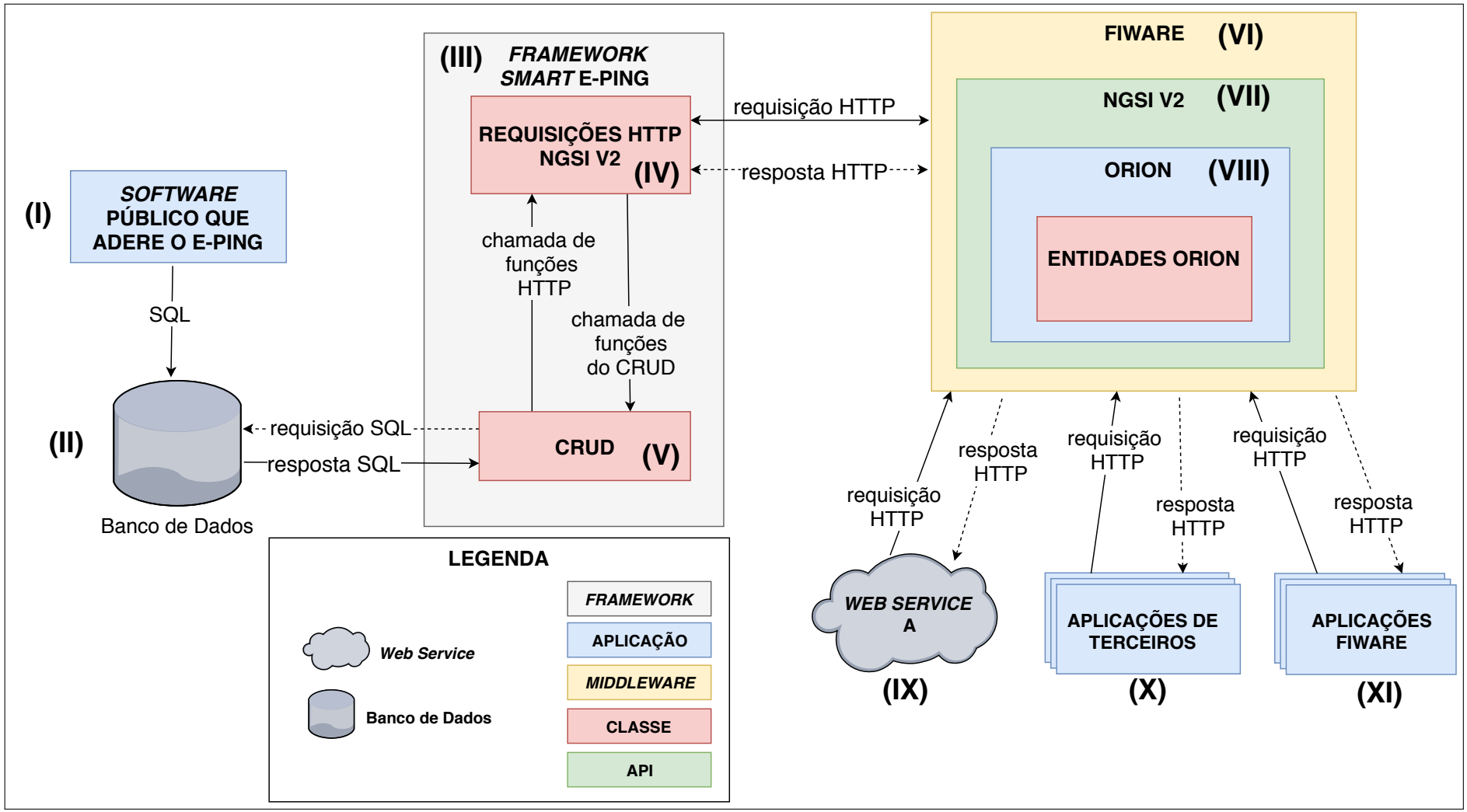
A Figura 8 apresenta uma visão mais detalhada do *Framework Smart* e-PING. O *Framework Smart* e-PING divide-se em duas classes, que são a Requisição HTTP **(IV)** e o CRUD **(V)**. A Requisição HTTP **(IV)** é responsável pela comunicação com o CRUD **(V)** e com a FIWARE **(VI)**. Outrossim, recebe as informações no formato SQL provindas da classe CRUD, por meio de chamadas de funções HTTP e recebe as informações provindas da FIWARE, por meio de requisição HTTP. Outrossim, a Requisição HTTP é responsável pelo tratamento e padronização dos dados, para que a rota possibilite funcionalidades como adição, exclusão, consulta e edição. Essas funcionalidades alimentam o consumo de dados provenientes das requisições realizadas pela FIWARE **(VI)**. De forma adicional, a classe Requisição HTTP envia os dados e/ou requisições realizadas pelas aplicações e/ou *Web Services* para a classe CRUD, que é responsável por emitir uma resposta ao *Software* Público.

Por fim, a classe CRUD **(V)** contém as seguintes funções: ***C**reate*, ***R**ead*, ***U**pdate* e ***D**elete*. O CRUD é uma classe responsável pelas funções de criação, alteração, exclusão ou edição de uma nova informação/ entidade[1] na Base de Dados do *Software* Público. Estas funções são realizadas por meio de requisições de *Web Services*, que são solicitadas pelos usuários das aplicações externas que utilizam a FIWARE **(IX, X** ou **XI)**. O CRUD é responsável por se comunicar diretamente com a Base de Dados do *Software* Público **(II)**, por meio destas funções. Esse processo funciona com base na camada de aplicação da rede por meio do protocolo HTTP, que é possível obter respostas mediante os arquivos no formato JSON. Outrossim, o CRUD é responsável por receber as informações tanto da classe Requisições HTTP **(VI)**, quanto da Base de Dados do *Software* Público **(II)**.

Conforme a Figura 8, pode-se exemplificar a comunicação da FIWARE para o *Software* Público do seguinte modo. Supondo que uma aplicação **(X** ou **XI)** e/ou *Web Service* **(IX)** deseje uma determinada localização de uma cidade, fornecida pelo *Software* Público **(I)**. Essa aplicação

[1] A entidade se trata de uma descrição que contém atributos dos objetos e meta dados relacionados a detalhes das características. Esses atributos, por sua vez representam uma instância de dados quanto a uma característica (ELMANGOUSH et al., 2013; FIWARE, 2018).

Figura 8 – Visão Geral do *Framework Smart* e-PING.



Fonte – Autora (2019).

realiza uma requisição HTTP, diretamente na base ORION, da plataforma FIWARE. O ORION verifica se existe em seu Banco de Dados, essa localização. Se existir, retorna a aplicação **(X** ou **XI)** e/ou *Web Service* **(IX)** a localização desejada. Caso não exista, o ORION realiza uma requisição HTTP para a classe Requisição HTTP do *Framework Smart* e-PING.

A classe Requisição HTTP verifica qual a operação provinda da FIWARE, como por exemplo *GET*, *POST*, *DELETE* ou *UPDATE*. Em seguida, a classe Requisição HTTP realiza uma chamada de função HTTP para a classe CRUD, passando como parâmetro as informações solicitadas.

A classe CRUD por sua vez, irá se comunicar diretamente com a Base de Dados do *Software* Público e obter a localização desejada pela aplicação externa. Uma vez que a classe CRUD obtiver essas informações, realiza uma chamada de função CRUD passando como parâmetros as informações que obteve no Banco de Dados do *Software* Público para a classe Requisição HTTP. Nessa classe, serão traduzidas as informações do Banco de Dados (no formato SQL) para o formato JSON. Em seguida, a classe Requisição HTTP envia ao FIWARE as informações no formato JSON por meio de uma determinada URL parametrizada pela NGSI versão 2. Com isso, a FIWARE se encarrega de enviar as informações para as aplicações que solicitaram.

Conforme a Figura 8, pode-se exemplificar a comunicação do *Software* Público para a FIWARE do seguinte modo. O *Framework Smart* e-PING por meio da classe CRUD, acessa a Base de Dados do *Software* Público e obtem as informações desejadas. Em seguida, a classe CRUD realiza uma chamada de função CRUD para a classe Requisição HTTP, passando como parâmetro as informações desejadas. Na classe Requisição HTTP, essas informações serão traduzidas no formato JSON. Após essa tradução, a classe Requisição HTTP envia a FIWARE as informações no formato JSON por meio de uma determinada URL parametrizada pela NGSI versão 2. Com isso, a FIWARE se encarrega de notificar as aplicações que possuem interesse nas informações disponibilizadas pelo *Software* Público.

Vale destacar que a cada nova informação inserida/removida/atualizada na Base de Dados do *Software* Público **(II)**, é necessária a atualização dessas informações na FIWARE **(VI)**.

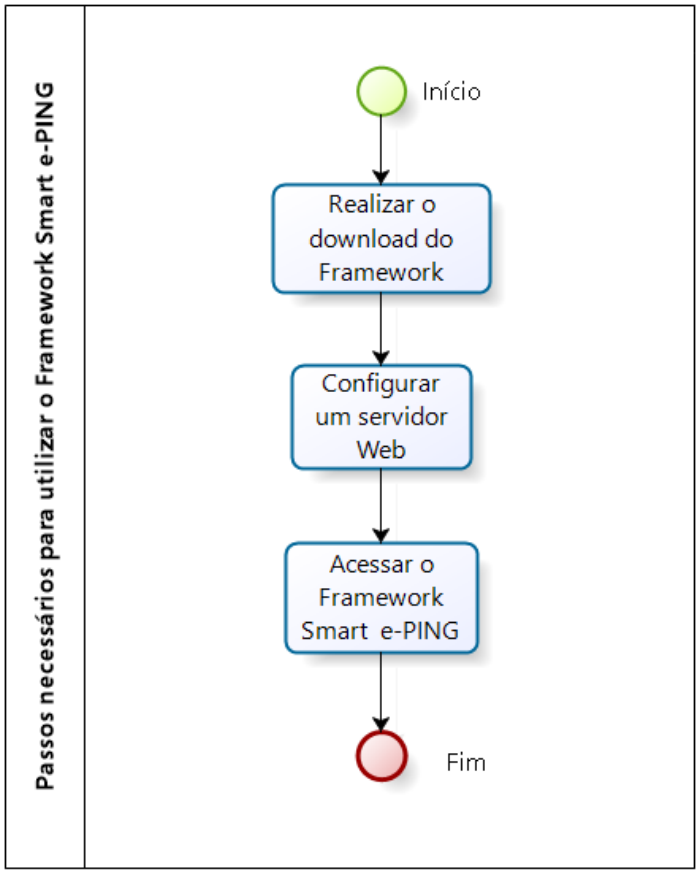
A função do *Framework Smart* e-PING é estabelecer a interoperabilidade entre um *Software* Público e a plataforma FIWARE. Desse modo, a estratégia de comunicação foi dividida em duas etapas, que são apresentadas nas seções a seguir.

4.1 Interoperabilidade do *Software* Público para a FIWARE

Nesta seção é apresentada como é realizada a interoperabilidade do *Software* Público para a FIWARE, por meio do ORION. Para isso, é necessário que o usuário das organizações, do GEB, tenham um entendimento em aplicações *Web*. Em outras palavras, o usuário precisa ter um

entendimento em instalações de pacotes no *Linux*/*Windows* e configuração básica do Apache, PHP e MySQL. Com esse entendimento, é possível configurar o *Framework Smart* e-PING corretamente. Outrossim, o usuário deverá saber qual é a Base de Dados que ele deseja tornar interoperável com a plataforma FIWARE.

Figura 9 – Passos para utilizar o *Framework Smart* e-PING.



Fonte – Autora (2019).

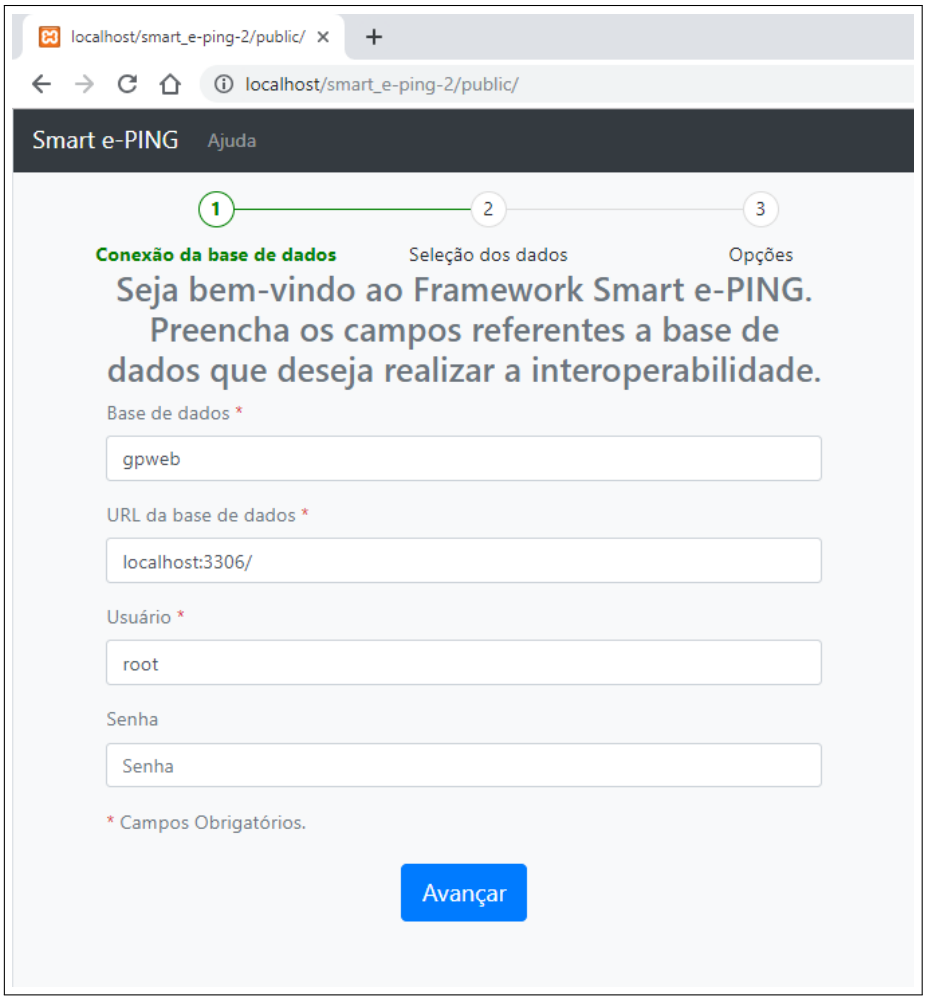
Conforme apresentado na Figura 9, a interoperabilidade do *Software* Público para o ORION é realizada da seguinte forma: o primeiro passo para utilizar o *Framework Smart* e-PING é realizado pelo usuário. O usuário realiza o *download* do *Framework Smart* e-PING, disponível no GitHub[2]. Em seguida, configura um servidor *Web* que contenha os serviços Apache e MySQL, como por exemplo o XAMPP[3].

Após a configuração correta apresentada no Apêndice A, o usuário acessa o *Framework Smart* e-PING, de qualquer navegador *Web*, por meio do endereço previamente configurado "<http://localhost/smart_e-ping-2/public/>". Ao acessar esse endereço, exibe-se a tela de conexão, conforme apresentado na Figura 10.

[2] <https://github.com/ThauaneDcomp/e-smart>.

[3] É um ambiente de desenvolvimento da linguagem PHP (XAMPP, 2019). Disponível para *download* em: <https://www.apachefriends.org/pt_br/download.html>.

Figura 10 – Tela de Conexão do *Framework Smart* e-PING.



Fonte – Autora (2019).

A fim de realizar a conexão no *Framework Smart* e-PING, o usuário deve informar todos os dados necessários da sua base, que são:

a) **Base de Dados**: nome do Banco de Dados do *Software* Público que deseja utilizar para realizar a interoperabilidade. Conforme apresentado na Figura 10, utilizou-se a Base de Dados do *Software* Público GP*Web*[4].

b) **URL da Base de Dados**: endereço/local em que se encontra a Base de Dados do *Software* Público. Conforme apresentado na Figura 10, a Base de Dados do *Software* Público GP*Web* está localizada na máquina *localhost* na porta 3306.

c) **Usuário**: usuário do Banco de Dados do *Software* Público.

d) **Senha**: senha do Banco de Dados do *Software* Público.

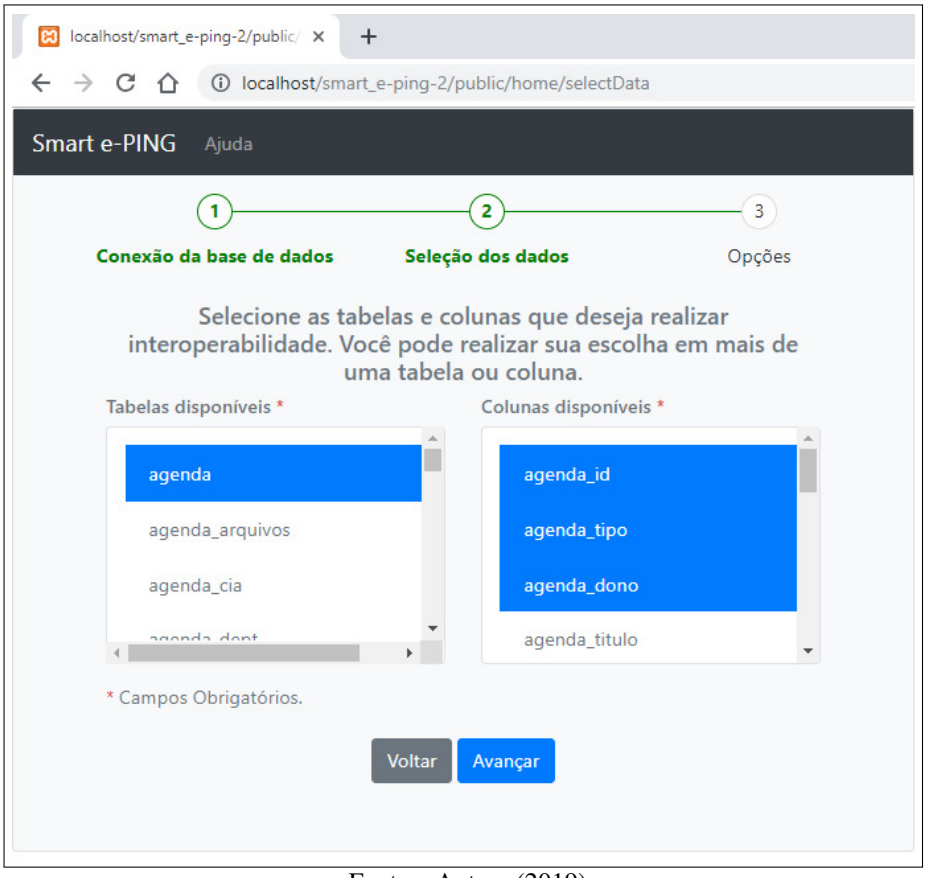
[4] <https://softwarepublico.gov.br/social/gpweb>.

Estes dados são fundamentais para que o *Framework Smart* e-PING tenha acesso ao Banco de Dados do *Software* Público e realize a interoperabilidade. Após fornecer todos os dados necessários, o usuário deve clicar em "*Avançar*". Em seguida, o *Framework Smart* e-PING verifica se existe a conexão com a Base de Dados do *Software* Público. Caso não encontre, exibe-se uma mensagem na tela informando que ocorreu falha de conexão com o banco.

Vale destacar que a conexão no Banco de Dados do *Software* Público, somente será estabelecida se as informações fornecidas forem condizentes com a autenticação da Base de Dados informada.

Após a autenticação no Banco de Dados do *Software* Público, o *Framework Smart* e-PING apresenta uma listagem dos dados disponíveis neste banco. O *Framework Smart* e-PING exibe todas as tabelas e suas respectivas colunas disponíveis, conforme apresentado na Figura 11.

Figura 11 – Tela de Listagem das Tabelas e Colunas.

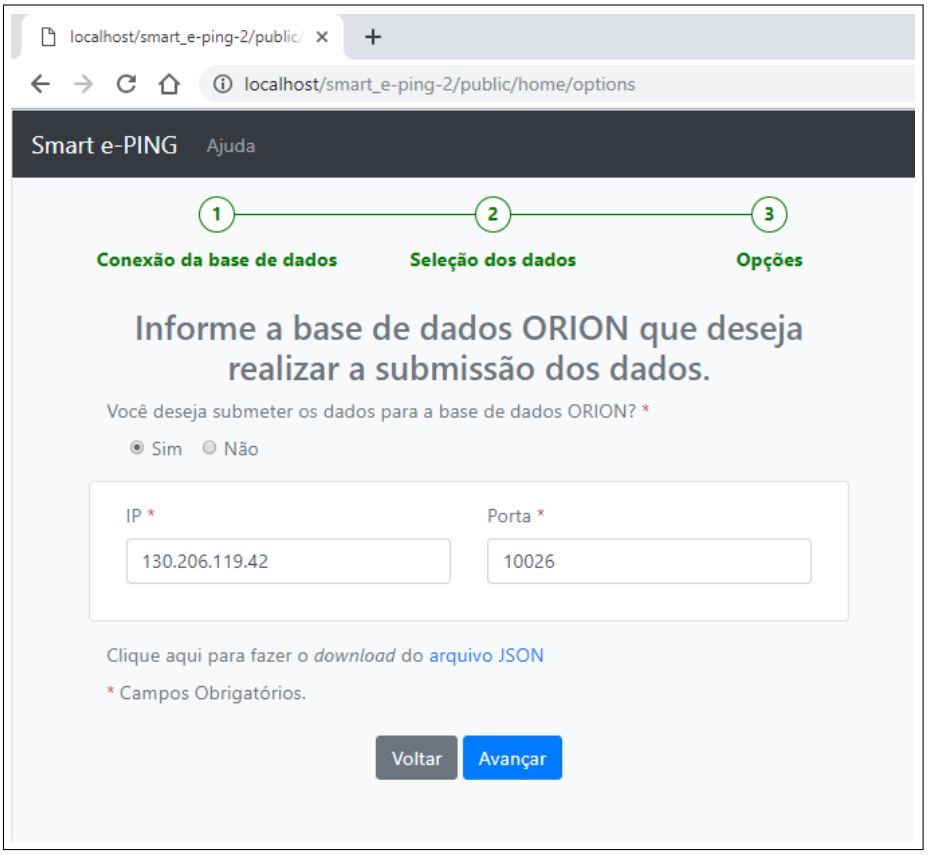


Fonte – Autora (2019).

O *Framework Smart* e-PING permite que o usuário selecione mais de uma tabela e/ou coluna, conforme desejar. Caso o usuário deseje retornar a tela anterior, deve clicar no botão “Voltar”. Após selecionar a(s) tabela(s) e a(s) coluna(s) que deseja realizar a interoperabilidade, o usuário deve clicar em "Avançar". Em seguida, o *Framework Smart* e-PING converte todos os

dados selecionados para o formato JSON. Finalizada a conversão, o usuário é redirecionado para a tela de opções, conforme apresentado na Figura 12.

Figura 12 – Tela de Opções.



Fonte – Autora (2019).

Conforme ilustrado na Figura 12, o usuário terá as seguintes opções de ações a realizar: efetuar o *download* do arquivo JSON e/ou enviar os dados para uma Base de Dados ORION. Caso selecione a opção "enviar para a Base de Dados ORION", é necessário que seja informado o endereço *Internet Protocol* (IP) e a porta em que a base ORION se encontra. Caso deseje "efetuar *download* do arquivo JSON", o usuário deve clicar em "arquivo JSON" e automaticamente será realizado o *download* do arquivo do tipo ".json". Outrossim, o usuário pode realizar as duas ações, caso desejar.

Caso o usuário deseje retornar a tela anterior, deve clicar no botão “Voltar”. Após selecionar a opção desejada, o usuário deve clicar em "Avançar". Em seguida, o *Framework Smart* e-PING envia todos os dados selecionados para o IP e porta em que a base ORION se encontra.

Vale destacar que o *Framework Smart* e-PING está se conectando somente com o Banco de Dados MySQL. Posteriormente, serão implementados novos módulos para que os demais Bancos de Dados possam utilizá-lo.

4.2 Interoperabilidade da FIWARE para o *Software* Público

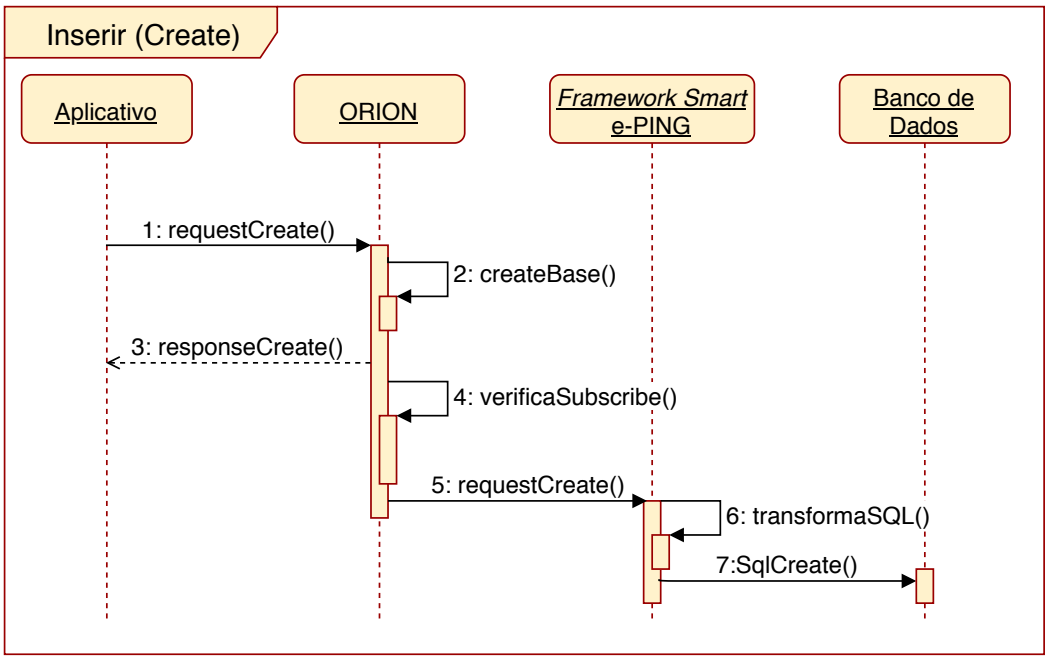
Nesta seção é apresentada como é realizada a interoperabilidade da FIWARE, por meio do ORION, para o *Software* Público. Essa interoperabilidade é realizada pelas seguintes funções CRUD: ***C**reate*, ***R**ead*, ***U**pdate* e ***D**elete*. O CRUD consiste em uma etapa essencial para a interoperabilidade entre sistemas heterogêneos, que serão descritos a seguir.

4.2.1 Inserir (*Create*)

A Figura 13 apresenta como um usuário por meio do aplicativo externo, que utiliza a FIWARE, pode realizar inserções dos dados no Banco de Dados do *Software* Público. O usuário, por meio do aplicativo externo, realiza uma requisição HTTP do tipo "*POST*" ao ORION. Ao receber essa requisição, o ORION verifica se o identificador (ID) informado na requisição existe em sua Base de Dados MongoDB. Se não há dados com o mesmo ID, o ORION insere os dados em seu banco e verifica se existem *subscribes*. Se existirem dados com o mesmo ID, o ORION informa ao aplicativo externo que esses dados existem.

Se existirem *subscribes*, o ORION envia os dados da solicitação para o *Framework Smart* e-PING. Ao receber essa solicitação, o *Framework Smart* e-PING transforma os dados JSON em SQL. Em seguida, o *Framework Smart* e-PING insere os dados no Banco de Dados do *Software* Público. Se não existirem *subscribes*, a requisição termina com a resposta à aplicação externa informando se os dados foram inseridos ou não na Base de Dados MongoDB e consequentemente na Base de Dados do *Software* Público.

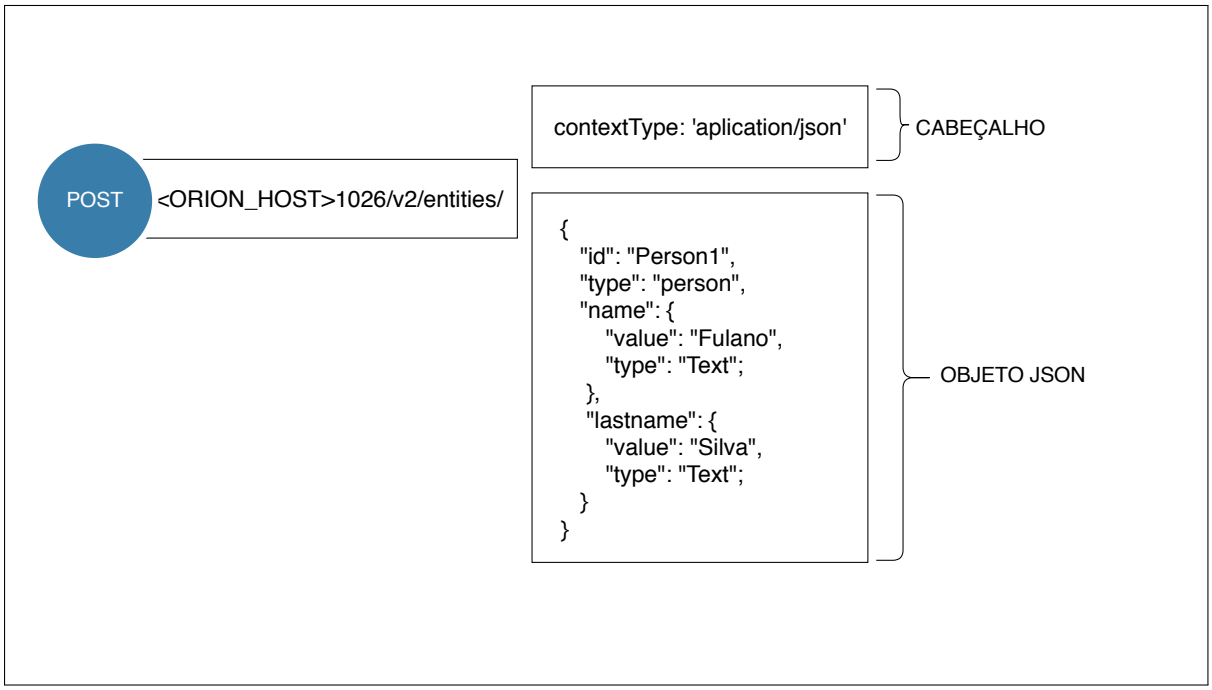
Figura 13 – Diagrama de Sequência da função *Create*.



Fonte – Autora (2019).

No exemplo a seguir, um usuário por meio do aplicativo externo realiza a inserção de uma pessoa "*person*" no Banco de Dados do *Software* Público. O usuário, por meio do aplicativo externo, realiza uma requisição HTTP do tipo "*POST*" ao ORION, com base no padrão de comunicação NGSIv2. Essa requisição contém o servidor, porta, versão do NGSI, cabeçalho, o corpo da mensagem do objeto JSON e gatilhos, conforme apresentado na Figura 14.

Figura 14 – Exemplo de uma requisição do tipo "*POST*".



Fonte – Autora (2019).

Na Figura 14, o usuário realizou uma requisição do tipo "*POST*", para o servidor "*ORION_HOST*" que está localizado na porta "1026". Ademais, informou a versão do NGSI "v2" e o gatilho "*entities*" que é recebido pelo ORION. Uma vez que o ORION recebeu essa requisição, é verificado se o ID "*Person1*", do objeto JSON, existe na sua Base de Dados MongoDB. Se o ID existir, o ORION informa ao aplicativo externo que esse dado existe. Caso esse ID não exista, o ORION realiza a inserção em seu Banco de Dados e verifica se existem *subscribes*.

Se não existirem *subscribes*, o ORION informa ao aplicativo externo uma mensagem de inserção concluída ou erro na inserção. Se existirem *subscribes*, o ORION envia os dados da solicitação para o *Framework Smart* e-PING, por meio da URL "<http://smarteping:8080/listen?action=insert>", conforme apresentado no Código 2. Ao receber a solicitação, o *Framework Smart* e-PING transforma os dados JSON em SQL. Em seguida, insere os dados no Banco de Dados do *Software* Público.

Código 2 – *Subscribe* da função *Create*.

```
1  {
2    "description": "A signature to insert data on the Public Software database",
3    "subject": {
4      "entities": [
5        {
6          "type": "person"
7        }
8      ]
9    },
10   "notification": {
11     "http": {
12       "url": "http://smarteping:8080/listen?action=insert"
13     },
14     "attrs": [
15       "name",
16       "lastname"
17     ]
18   }
19 }
```

Fonte – Autora (2019).

A seguir, são explicados em detalhes os diferentes elementos contidos no *subscribe* apresentado no Código 2:

a) O subcampo "*entities*" **(linha 4)** dentro de "*subject*" **(linha 3)** é o parâmetro a ser analisado como filtro do *subscribe*. Neste caso, o ORION filtra as requisições recebidas do tipo "*person*" **(linha 6)**. Em outras palavras, se enquadram neste *subscribe* o objeto JSON que contenha o campo "*type*: "*person*"" **(linha 6)**.

b) O subcampo "*attrs*" **(linha 14)** dentro de "*notifications*" **(linha 10)** define o conteúdo das mensagens de notificação. Neste exemplo, é especificado que a notificação deve incluir os atributos "*name*" **(linha 15)** e "*lastname*" **(linha 16)** na entidade "*person*".

c) O subcampo "URL" **(linha 12)** é responsável por enviar notificação na URL aqui definida. Neste exemplo, foi utilizada a URL do *Framework Smart* e-PING. Apenas uma URL pode ser incluída por assinatura. No entanto, pode ter várias assinaturas nos mesmos elementos de contexto, ou seja, mesma entidade e atributo sem nenhum problema.

d) O subcampo "*description*" **(linha 2)** descreve o que a *subscribe* realiza.

Dentro deste *subscribe*, se a entidade do tipo "*person*" for inserida no Banco de Dados do ORION, é criada uma requisição para a URL, descrito no Código 2, com os dados descritos

no campo "*attrs*" (atributos). Em seguida, o ORION envia ao aplicativo uma notificação que os dados foram inseridos com sucesso. Paralelamente a essa resposta ao ORION, o *Framework Smart* e-PING recebe a solicitação enviada pelo ORION e realiza o processo de tradução de JSON para SQL. Em seguida, insere esses dados na Base de Dados do *Software* Público. Esta inserção no Banco de Dados é feita pelo comando SQL apresentado no Código 3.

Código 3 – Exemplo do comando da função *Insert*.

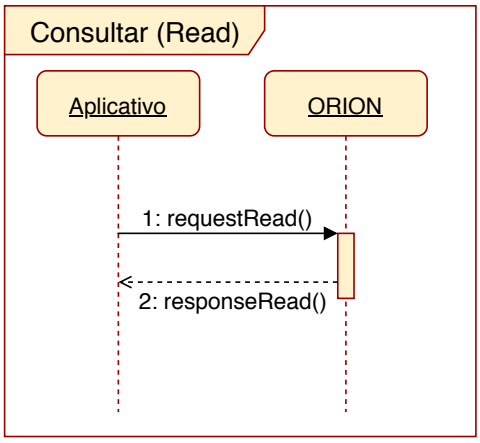
```
1 INSERT INTO person (name, lastname)
2 VALUES ("Fulano", "Silva");
```

Fonte – Autora (2019).

4.2.2 Consultar (*Read*)

A Figura 15 apresenta como um usuário por meio do aplicativo externo que utiliza a FIWARE, pode consultar informações dos dados no Banco de Dados do *Software* Público. O usuário, por meio do aplicativo externo, realiza uma requisição HTTP do tipo "*GET*" ao ORION. Ao receber essa requisição, o ORION verifica se o ID informado na requisição existe em sua Base de Dados MongoDB. Se não existirem dados com o mesmo ID, o ORION informa que os dados não existem em seu Banco de Dados. Se existirem dados com o mesmo ID, o ORION informa à aplicação externa que esse dado não existe.

Figura 15 – Diagrama de Sequência da função *Read*.



Fonte – Autora (2019).

No exemplo a seguir, um usuário por meio do aplicativo externo realiza a consulta de uma pessoa "*person*" no Banco de Dados do *Software* Público. O usuário, por meio do aplicativo externo, realiza uma requisição HTTP do tipo "*GET*" ao ORION, com base no padrão

de comunicação NGSIv2. Assim, a comunicação pode ocorrer de quatro formas, conforme apresentado nos itens abaixo.

a) Para consultar todas as entidades registradas no banco do ORION (MongoDB), é necessário realizar o seguinte comando no *prompt* de comando:

```
curl <orion_host>:1026/v2/entities -s -S -H 'Context-Type: application/json' -d @-
↪  <<EOF
```

b) Para filtrar as entidades por tipo "*person*", é necessário realizar o seguinte comando:

```
curl <orion_host>:1026/v2/entities?type=person -s -S -H 'Context-Type:
↪  application/json' -d @- <<EOF
```

c) Para consultar uma única entidade, por meio da identidade (ID), é necessário realizar o seguinte comando:

```
curl <orion_host>:1026/v2/entities/person1 -s -S -H 'Context-Type: application/json'
↪  -d @- <<EOF
```

d) Para requisitar apenas um atributo de uma entidade, é necessário realizar o seguinte comando:

```
curl <orion_host>:1026/v2/entities/person1/attrs/name -s -S -H 'Context-Type:
↪  application/json' -d @- <<EOF
```

A seguir, são explicados em detalhes os diferentes elementos contidos nessa consulta:

a) **curl:** ferramenta de linha de comando para requisições HTTP. Utilizado em qualquer sistema operacional.

b) **<orion_host>:1026/v2/entities:** “orion_host" é o endereço da FIWARE, que está localizado na porta "1026". Ademais, informou a versão do NGSI "v2" e o gatilho "entities" que é recebido pelo ORION.

c) **-s -S:** exibe mensagem de erro ao falhar.

d) **-H:** passa o(s) cabeçalho(s) personalizado(s) para o servidor.

e) **'context-type: application/json':** cabeçalho utilizado para especificar o formato JSON.

f) **-d:** retorna a data que realizou a consulta.

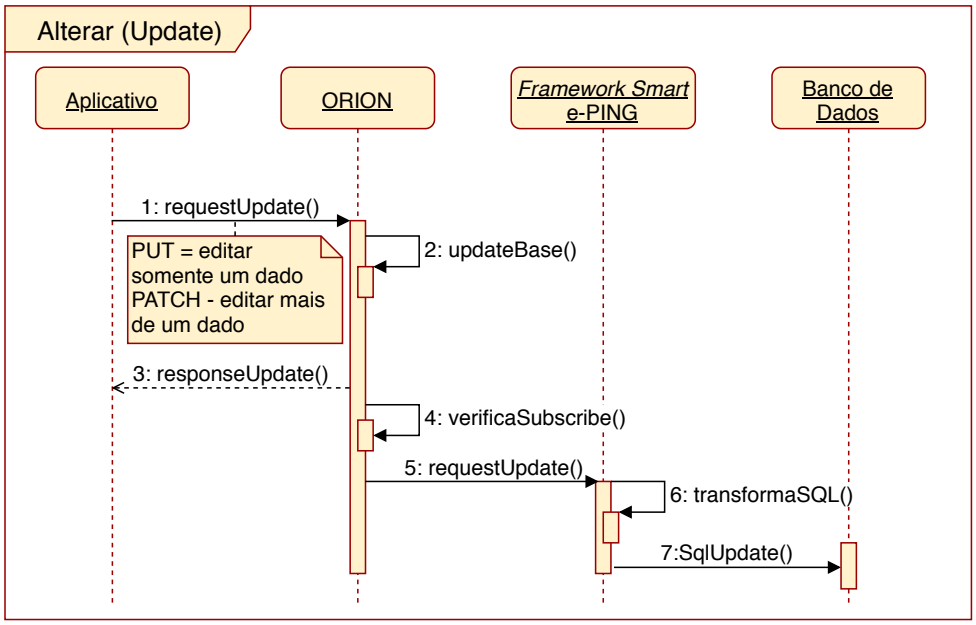
g) **@- « EOF:** fim do comando curl.

4.2.3 Alterar (*Update*)

Para uma aplicação externa, que utiliza a FIWARE, realizar uma alteração na Base de Dados do *Software* Público, deve-se seguir os passos que são apresentados na Figura 16. O usuário, por meio do aplicativo externo, realiza uma requisição HTTP do tipo "*PATCH*" ou "*PUT*" ao ORION. Ao receber essa requisição, o ORION verifica se o ID informado na requisição existe em sua Base de Dados MongoDB. Se existirem dados com o mesmo ID, o ORION altera os dados em seu banco e verifica se existem *subscribes*. Caso esse ID não exista, o ORION informa ao aplicativo externo que esse ID já existe.

Se existirem *subscribes*, o ORION envia os dados da solicitação para o *Framework Smart* e-PING. Ao receber essa solicitação, o *Framework Smart* e-PING transforma os dados JSON em SQL e altera os dados no Banco de Dados do *Software* Público. Se não existirem *subscribes*, a requisição termina com a resposta à aplicação externa informando se os dados foram alterados ou não na Base de Dados MongoDB.

Figura 16 – Diagrama de Sequência da função *Update*.



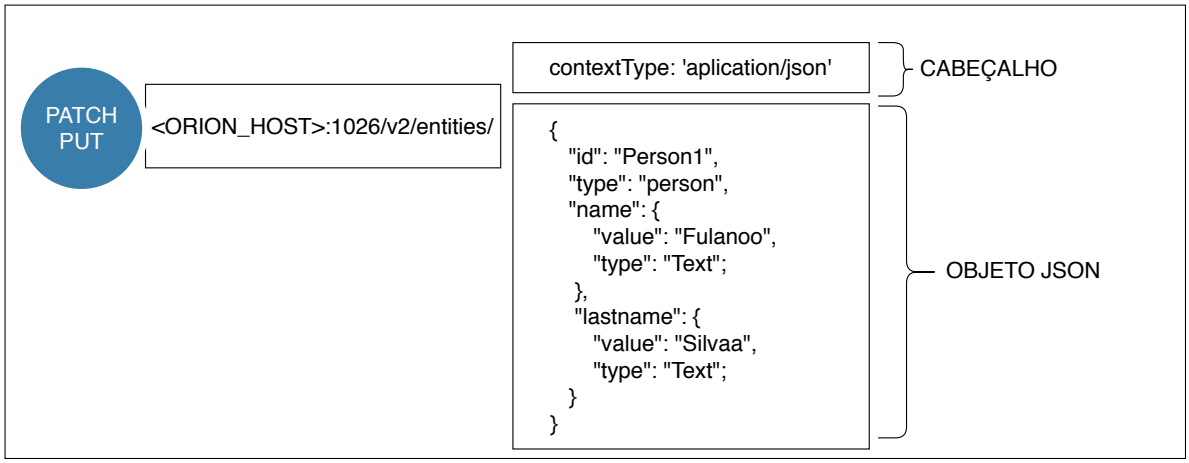
Fonte – Autora (2019).

Como mostrado na Figura 16, o aplicativo externo pode executar a mudança usando dois métodos, como "*PUT*" e "*PATCH*". O método "*PUT*" solicita o armazenamento/atualização de

uma única informação/entidade. Finalmente, o método "*PATCH*", diferente do "*PUT*", é usado para armazenar ou atualizar um recurso. No entanto, o "*PATCH*" é feito para alterar mais de uma informação/entidade.

No exemplo anterior, um usuário por meio do aplicativo externo realiza a alteração de uma pessoa "*person*" ("*name*" e "*lastname*") no Banco de Dados do *Software* Público. O usuário, por meio do aplicativo externo, realiza uma requisição HTTP do tipo "*PATCH*" ao ORION, com base no padrão de comunicação NGSIv2. Essa requisição contém o servidor, porta, versão do NGSI, cabeçalho, o corpo da mensagem do objeto JSON e gatilhos, conforme ilustrado na Figura 17.

Figura 17 – Exemplo de uma requisição do tipo "*PATCH*".



Fonte – Autora (2019).

Como por exemplo na Figura 17, o usuário realizou uma requisição do tipo *PATCH*, para o servidor "*ORION_HOST*" que está localizado na porta "1026". Ademais, informou a versão do NGSI "v2" e o gatilho "*entities*" que é recebido pelo ORION. Uma vez que o ORION recebeu essa requisição, é verificado se o ID "*Person1*", do objeto JSON, existe na sua Base de Dados MongoDB. Caso esse ID não exista, o ORION retorna mensagem de erro, informando que não existe esse ID. Caso esse ID exista, o ORION realiza a alteração em seu Banco de Dados e verifica se existem *subscribes*.

Se existirem *subscribes*, o ORION envia os dados da solicitação para o *Framework Smart* e-PING, por meio da URL "<http://smarteping:8080/listen?action=update>", conforme apresentado no Código 4. Ao receber a solicitação, o *Framework Smart* e-PING transforma os dados JSON em SQL. Em seguida, altera os dados no Banco de Dados do *Software* Público. Se não existirem *subscribes*, o ORION informa ao aplicativo externo uma mensagem de alteração concluída ou erro na alteração.

Uma vez que a requisição é realizada, ORION recebe os dados e verifica se o campo identidade (ID) "*Person*1" existe no Banco de Dados, se ele existir, ele altera os dados e verifica se existe *subscribe*, conforme apresentado no Código 4.

Código 4 – *Subscribe* da função *Update*.

```
{
  "description": "A signature to update data on the  Public Software database",
  "subject": {
    "entities": [
      {
        "type": "person"
      }
    ]
  },
  "notification": {
    "http": {
      "url": "http://smarteping:8080/listen?action=update"
    },
    "attrs": [
      "name",
      "lastname"
    ]
  }
}
```

Fonte – Autora (2019).

Considerando este *subscribe*, se a entidade do tipo "*person*" for alterada no Banco de Dados do ORION, é criada uma requisição para a URL, apresentada no Código 4, com os dados descritos no campo "*attrs*" (atributos). Em seguida, o ORION envia ao aplicativo uma notificação que os dados foram modificados com sucesso. Em paralelo a esta resposta ao ORION, o *Framework Smart* e-PING recebe a solicitação enviada pelo ORION e realiza o processo de tradução de JSON para SQL. Em seguida, altera esses dados na Base de Dados do *Software* Público. Esta alteração no Banco de Dados é feita pelo comando SQL, apresentado no Código 5.

Código 5 – Exemplo do comando da função *Update*.

```
UPDATE person SET name = "Fulanoo", lastname = "Silvaa"
WHERE id = "person1";
```

Fonte – Autora (2019).

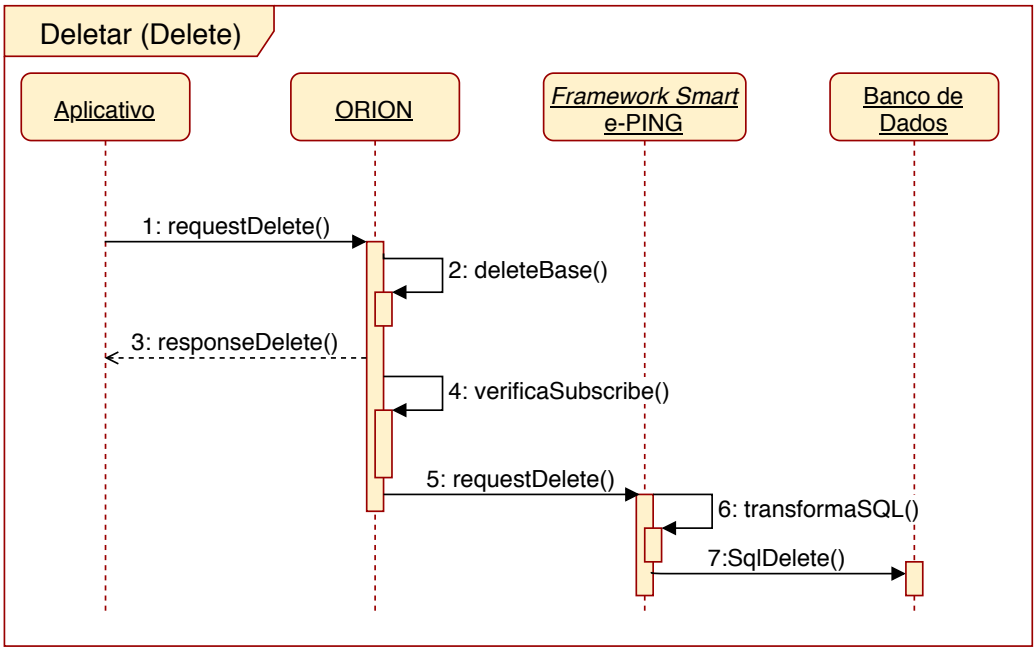
4.2.4 Deletar (*Delete*)

Na Figura 18 é exemplificado como uma aplicação externa, que utiliza a FIWARE, realiza a remoção de dados na base do *Software* Público, demonstrando os passos necessários

para realizar essa ação. O usuário, por meio do aplicativo externo, realiza uma requisição HTTP do tipo "*DELETE*" ao ORION. Ao receber essa requisição, o ORION verifica se o ID informado na requisição existe em sua Base de Dados MongoDB. Se não existirem os dados com o mesmo ID, o ORION informa ao aplicativo externo que a informação/entidade não existe. Se existirem dados com o mesmo ID, o ORION apaga os dados em seu banco e verifica se existem *subscribes*.

Se existirem *subscribes*, o ORION envia os dados da solicitação para o *Framework Smart* e-PING. Ao receber essa solicitação, o *Framework Smart* e-PING transforma os dados JSON em SQL e apaga os dados no Banco de Dados do *Software* Público. Se não existirem *subscribes*, a requisição termina com a resposta à aplicação externa informando se os dados foram apagados ou não na Base de Dados MongoDB.

Figura 18 – Diagrama de Sequência da função *Delete*.



Fonte – Autora (2019).

No exemplo anterior, um usuário por meio do aplicativo externo realiza a remoção de uma pessoa "*person*" no Banco de Dados do *Software* Público. O usuário, por meio do aplicativo externo, realiza uma requisição HTTP do tipo "*DELETE*" ao ORION, com base no padrão de comunicação NGSIv2. Assim, a remoção é da seguinte forma, conforme ilustrado no Código 6.

Código 6 – Exemplo de Comunicação da função *Delete*.

```
1 curl <orion_host>:1026/v2/entities/person1 -s -S -H 'Context-Type: application/json'
  ↪  -X DELETE @- <<EOF
```

Fonte – Autora (2019).

No Código 6, o usuário realizou uma requisição do tipo *DELETE*, para o servidor "*ORION_HOST*" que está localizado na porta "1026". Ademais, informou a versão do NGSI "v2" e o gatilho "*entities*" que é recebido pelo ORION. Uma vez que o ORION recebeu essa requisição, é verificado se o ID "*person1*", do objeto JSON, existe na sua Base de Dados MongoDB. Se não existir esse ID, o ORION informa a aplicação externa que não existe a informação/entidade. Caso esse ID exista, o ORION realiza a remoção em seu Banco de Dados e verifica se existem *subscribes*.

Se existirem *subscribes*, o ORION envia os dados da solicitação para o *Framework Smart* e-PING, por meio da URL "<http://smarteping:8080/listen?action=delete>", conforme apresentado no Código 7. Ao receber a solicitação, o *Framework Smart* e-PING transforma os dados JSON em SQL. Em seguida, remove os dados no Banco de Dados do *Software* Público. Se não existirem *subscribes*, o ORION informa ao aplicativo externo uma mensagem de remoção concluída ou erro na remoção.

Código 7 – *Subscription* da função *Delete*.

```
{
  "description": "A signature to delete data on the  Public Software database",
  "subject": {
    "entities": [
      {
        "type": "person"
      }
    ]
  },
  "notification": {
    "http": {
      "url": "http://smarteping:8080/listen?action=delete"
    },
    "attrs": [
      "id"
    ]
  }
}
```

Fonte – Autora (2019).

Considerando este *subscribe*, se a entidade do tipo "*person*" for removida no Banco de Dados do ORION, é criada uma requisição para a URL, apresentado no Código 7, com os dados descritos no campo "*attrs*" (atributos). Em seguida, o ORION envia ao aplicativo uma notificação que os dados foram removidos com sucesso. Paralelamente a essa resposta ao ORION, o *Framework Smart* e-PING recebe a solicitação enviada pelo ORION e realiza o processo de tradução de JSON para SQL. Em seguida, remove esses dados na Base de Dados do

Software Público. Esta remoção no Banco de Dados é feita pelo comando SQL, apresentado no Código 8.

Código 8 – Exemplo do comando da função *Delete*.

```
1 DELETE FROM person
2 WHERE id = "person1";
```

Fonte – Autora (2019).

Desta forma, foi possível transformar os dados em formato JSON para uma plataforma de Cidades Inteligentes para que outras aplicações consigam acessar os dados e utilizá-los. Além dessa vantagem, *softwares* como o *Comprehensive Knowledge Archive Network* (CKAN) poderão conseguir fazer proveito destes dados. Esses dados são traduzidos pelo *Framework Smart* e-PING e disponibilizados aos usuários de aplicações externas e/ou *Web Services*, sem que estes usuários tenham um conhecimento de computação. Dessa maneira, o GEB por meio do *Framework Smart* e-PING poderá disponibilizar seus dados dos *Softwares* Públicos, sem a necessidade de implementação de componentes em suas aplicações. Ademais, não é necessário prévio conhecimento da FIWARE, restando somente entender a interface do *Framework Smart* e-PING para efetuar o seu uso. Portanto, o *Framework Smart* e-PING dá suporte ao GEB a reduzir os seus custos, otimizar e melhorar os seus serviços públicos por meio de soluções inteligentes e a reutilização de recursos inteligentes.

Vale destacar que a cada nova informação que é inserida/removida/atualizada na Base de Dados do *Software* Público, faz-se necessária a atualização dessas informações em tempo real na base ORION, da plataforma FIWARE. Essa atualização é realizada por meio de funções que serão criadas no Banco de Dados do *Software* Público utilizando *triggers*[5]. Estas *triggers* serão acionadas com a operação da atualização dos dados no *Framework Smart* e-PING para que a *trigger* atualize/remova/insira os dados no ORION também. Este acionamento notifica o *Framework* para que estes dados sejam atualizados na Base de Dados do ORION.

No entanto, para que estas *triggers* sejam acionadas será necessário que a organização do Governo Brasileiro, que esteja utilizando o *Framework Smart* e-PING, autorize a criação de *triggers* em sua base de dados. Será uma *trigger* para cada tabela escolhida para interoperar, em que serão acionadas e em seguida irá enviar uma requisição HTTP para o *Framework Smart* e-PING, que recebe a requisição e envia as informações ao ORION.

Desta forma, ao executar o *Framework Smart* e-PING e passar pelo processo de configuração da aplicação, deve haver uma tela de autorização. Esta tela terá que exibir todas as

[5] Uma *trigger* é formada por ações em SQL, armazenadas no Banco de Dados. Essas ações são executadas automaticamente pelo Banco de Dados, sempre que há uma alteração, remoção ou verificação dos dados de uma determinada tabela (GEHANI; JAGADISH; SHMUELI, 1992; CERI; COCHRANE; WIDOM, 2000; FRATANTONIO et al., 2016).

triggers que serão criadas na base de dados do *Software* Público e solicitar que o usuário permita a criação das *triggers* em sua base. Desta forma, o usuário pode analisar as *triggers* antes de executá-la na base.

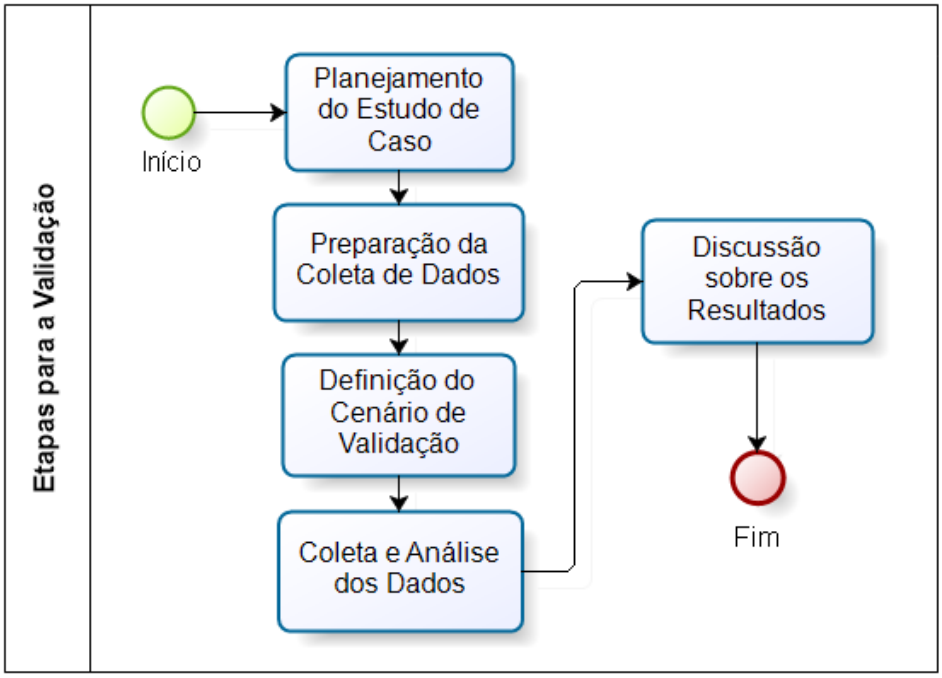
Por conseguinte, tanto as aplicações que trabalham com o *Software* Público quanto às aplicações FIWARE podem interagir com os dados em tempo real e sem quaisquer alterações no código fonte, devido ao *Framework Smart* e-PING.

5

Validação do *Framework Smart* e-PING

Este capítulo tem como objetivo validar a plataforma utilizando um estudo de caso, em forma de provas de conceito (JR.; CHEN; PURDIN, 1991). Um estudo de caso é um tipo de experimento planejado que objetiva analisar fenômenos de um determinado ambiente, por meio de múltiplas fontes de evidências (YIN, 2015; GIL, 2017).

Figura 19 – Etapas realizadas na validação.



Fonte – Autora (2019).

Conforme a Figura 19, para a realização do estudo de caso foram necessárias cinco etapas. O **planejamento do estudo de caso** define o objetivo a ser alcançado durante a execução desse estudo. A **preparação da coleta de dados** define as métricas referentes à norma ISO/IEC 25010. e as questões de pesquisa associadas a essas métricas. A **definição do cenário de validação**

consiste na descrição dos *softwares* utilizados, assim como as configurações necessárias para a realização da validação. A **coleta e análise dos dados** descreve como foi realizada a coleta dos dados, bem como das respostas às questões de validação. Por fim, **a discussão sobre os resultados** analisa os resultados da pesquisa.

5.1 Planejamento do Estudo de Caso

Conforme mencionado na Seção 1.6, essa validação visa avaliar a capacidade que o *Framework Smart* e-PING tem de interoperar um *Software* Público que adere ao e-PING com a plataforma FIWARE.

Para esta validação, foi utilizado o método *Goal/Question/Metric* (GQM) proposto por Basili (CALDIERA; ROMBACH, 1994; BASILI et al., 2014). Com base no GQM, o experimento teve como objetivo **analisar** o *Framework Smart* e-PING, **com o propósito de** avaliar, **com respeito a** interoperabilidade entre um *Software* Público e plataforma FIWARE, **do ponto de vista da** norma ISO/IEC 25010 (ISO/IEC, 2011), **no contexto das** próprias métricas e derivações da escala *Likert* (LIKERT, 1932).

Desta forma, as seguintes questões de validação foram definidas para a condução do experimento:

a) Questão de Validação 1 - Qual a frequência que o *Framework Smart* e-PING **traduz corretamente o arquivo do Banco de Dados** do *Software* Público para a plataforma FIWARE?

   Esta questão avalia se o *Framework Smart* e-PING traduz corretamente o arquivo lido do Banco de Dados para o formato JSON. Assim, é necessário validar se as traduções dos dados estão conforme o esperado, respeitando os formatos e a consistência dos mesmos.

b) Questão de Validação 2 - Com que frequência o *Framework Smart* e-PING **causa perda de informação** na tradução do arquivo do Banco de Dados do *Software* Público para a plataforma FIWARE?

   Esta questão avalia o número de perdas, paradas, erros ou traduções incorretas de informações ocorridas durante a tradução no *Framework Smart* e-PING. Outrossim, compara esse valor com o número de testes realizados.

c) Questão de Validação 3 - Qual a medida de **tempo** estimada para que o *Framework Smart* e-PING permita a interoperabilidade entre o *Software* Público e a plataforma FIWARE?

   Esta questão avalia o tempo em que o *Framework Smart* e-PING foi capaz de traduzir, com sucesso, o arquivo do Banco de Dados do *Software* Público para a plataforma FIWARE.

5.2 Preparação da Coleta de Dados

A experimentação foi realizada por meio das métricas propostas na norma ISO/IEC 25010 (ISO/IEC, 2011). As seguintes métricas foram definidas com base na documentação referente à norma ISO/IEC 25010. Elas estão associadas às questões de validação relatadas na Seção 5.1.

Métrica 1 - Validação da tradução dos dados com base nas tentativas de sucesso. Essa métrica contém o objetivo de avaliar a porcentagem em que a tradução dos dados entre o *Software* Público e a plataforma FIWARE é bem-sucedida. Esta métrica está associada à **Questão de Validação 1**, apresentada na Seção 5.1. O método de aplicação é contar o número total de testes que foram realizados e o número de testes em que a tradução foi bem-sucedida, conforme apresentado na Tabela 6.

Tabela 6 – Métrica 1 de validação durante a fase de desenvolvimento do *Framework Smart* e-PING.

Fórmula	Interpretação	Tipo de Escala[1]
$M_C = \frac{C}{NT} * 100$ M_C = Porcentagem de tradução bem-sucedida. C = Número de testes em que a tradução entre o *Software* Público e a plataforma FIWARE foi bem-sucedida. NT = Número de testes em que se tentou traduzir os dados do *Software* Público para a plataforma FIWARE.	$0 \leq M_C \leq 1$ Se o valor obtido estiver próximo a 100% é considerado como sendo mais favorável.	**Ótimo** = 100% 100% > **Muito Bom** $\geq$ 75% 75% > **Bom** $\geq$ 50% 50% > **Razoável** $\geq$ 25 % **Ruim** < 25%

Fonte – Elaborada com base na norma ISO/IEC (2011).

[1] Conforme a escala original *Likert* de 5 pontos (LIKERT, 1932).

Na métrica 1 foram comparados os dados já existentes por meio do *plugin* de comparação de arquivos da ferramenta Notepad++[2]. Esse *plugin* compara e mescla arquivos nos Sistemas Operacionais *Windows, Linux* ou OS X. Outrossim, compara linha por linha, caractere por caractere e palavra por palavra a cada linha. Ademais, possibilita a contagem de caracteres, linhas e palavras (NOTEPADD++, 2019). Portanto, utilizou-se essa ferramenta para realizar a checagem da integridade dos dados em cada campo, comparando caracteres, palavras e linhas. Uma vez verificada uma falha, é considerada como uma tradução inválida, relacionando-se à **questão de validação 2**.

O critério utilizado para essa métrica foi somente considerar o dado/conteúdo traduzido, verificando se este dado/conteúdo estava condizente com o arquivo do formato SQL. Portanto, foram desconsiderados os nomes dos campos e os campos, deixando de forma mais clara que o mais importante foram os dados/conteúdos presentes nesses campos.

Métrica 2 - Validação de detecção de falhas. É uma métrica importante para mostrar a estabilidade do *Framework*. Essa métrica verifica a porcentagem de falhas detectadas durante a execução do *Framework Smart* e-PING. O método de aplicação é contar o número de falhas detectadas durante a execução do *Framework Smart* e-PING e o número de testes realizados, conforme apresentado na Tabela 7.

Tabela 7 – Métrica 2 de validação durante a fase de desenvolvimento do *Framework Smart* e-PING.

Fórmula	Interpretação	Tipo de Escala[3]
$M_F = \frac{F}{NT} * 100$ M_F = Porcentagem de falhas detectadas. F = Número de falhas detectadas. NT = Número de testes realizados.	$0 \leq M_F$ Um valor de M_F reduzido significa qualidade do produto. M_F = 0 não indica necessariamente a ausência de falhas.	**Ótimo** = 0% 0% > **Muito Bom** $\geq$ 25% 25% > **Bom** $\geq$ 50% 50% > **Razoável** $\geq$ 75% **Ruim** < 75%

Fonte – Elaborada com base na norma ISO/IEC (2011).

Essa métrica 2 visa a identificação de falhas durante a execução do *Framework Smart* e-PING. Ela é obtida analisando a quantidade de paradas, erros ou traduções incorretas du-

[2] <https://sourceforge.net/projects/npp-compare/>.
[3] Conforme a escala original *Likert* de 5 pontos (LIKERT, 1932).

rante a execução do *Framework*. A métrica 2 é bem semelhante à métrica 1, porém dessa vez contabilizou-se as falhas verificadas nas palavras, caracteres e linhas. Esta métrica 2 está associada à **Questão de Validação 2**, apresentada na Seção 5.1.

Métrica 3 - Validação do tempo de execução. É uma métrica que contém um aspecto importante a ser considerado com relação a validação de um *Framework*. Essa métrica tem como objetivo calcular o tempo que o *Framework Smart* e-PING realiza uma tradução. Outrossim, verifica o tempo em que o *Framework* reage/conclui uma determinada operação. Desta forma, possibilita que essa métrica seja comparada com outras soluções que venham a existir. Esta métrica está associada à **Questão de Validação 3**, apresentada na Seção 5.1.

Para validar essa métrica utilizou-se a ferramenta *Wireshark*[4]. O *Wireshark* é uma ferramenta de análise em tempo real dos dados, em uma determinada rede. Outrossim, exibe todos os registros de data e hora em tempo absoluto (segundos), desde o início da captura desses dados. Esta ferramenta é amplamente utilizada por administradores de rede, para o gerenciamento/análise do comportamento do tráfego de pacotes que transita em uma determinada rede (WIRESHARK, 2019). Portanto, utilizou-se essa ferramenta para validar o tempo de execução do *Framework Smart* e-PING.

Tabela 8 – Métrica 3 de validação durante a fase de desenvolvimento do *Framework Smart* e-PING.

Fórmula	Interpretação
X = Tempo	$0 \leq X$ Quanto menor for o valor de X mais favorável é o resultado.

Fonte – Elaborada com base na norma ISO/IEC (2011).

5.3 Cenário de Validação

Esta validação foi composta por dois sistemas e o *Framework Smart* e-PING. A seguir são descritos os *softwares* escolhidos para realizar a interoperabilidade:

a) O primeiro *software*, GP*Web*, foi desenvolvido em março de 2008 e foi disponibilizado no Portal do *Software* Público Brasileiro[5] em 2017. O GP*Web* é uma aplicação *Web* desenvolvida em PHP e baseada no padrão de interoperabilidade do governo eletrônico que tem

[4] <https://www.wireshark.org/download.html>.

[5] Disponível em <https://softwarepublico.gov.br/social/gpweb> ou <https://www.sistemagpweb.com/>.

como objetivo gerenciar projetos, incorporando as melhores práticas de gestão. Ademais, possui módulos de gestão de recursos, gestão de documentos, agendas, colaboração social com controle de calendários, dentre outros. O GP*Web* segue a metodologia do *Balanced Score Card* (BSC) com o planejamento estratégico, ferramentas de gestão, dentre outros. Por fim, inclui módulos de comunicação e tramitação de documentos, o que possibilita à organização centralizar as informações em um único sistema.

Esse *software* foi escolhido por ser um *Software* Público que contém o Banco de Dados MySQL, banco aceito, até o momento, pelo *Framework Smart* e-PING. Ademais, por disponibilizar uma base de dados para a realização de testes, assim como, por ser um *software* com interface de fácil entendimento tanto para o seu uso quanto para a instalação.

b) O segundo *software*, *Aedes Tracker* foi desenvolvido em PHP. *Aedes Tracker* é um *software* de geoprocessamento que auxilia os agentes epidemiológicos no combate ao mosquito *Aedes Aegypti*. Esse *software* foi implementado para dar suporte aos projetos de Cidades Inteligentes, utilizando a plataforma FIWARE, com o objetivo de minimizar custos governamentais e introduzir melhorias na forma de acesso aos dados decorrentes do controle dessa doença. Ademais, contém funcionalidades como o processo de coleta e a supervisão de dados da doença *Aedes Aegypti*. Foi criado em 2018 por alunos de Iniciação Científica em Ciência da Computação da Universidade Federal de Sergipe (UFS).

   Esse projeto foi escolhido por se tratar de uma aplicação *open source* que foi desenvolvida para dar suporte aos projetos de Cidades Inteligentes, utilizando a plataforma FIWARE. Outrossim, por ser a primeira aplicação, recente, que está sendo desenvolvida pelos alunos da UFS e que aceitou contribuir com o nosso trabalho, oferecendo pleno suporte quanto ao funcionamento e configurações do *software*.

Para a execução do experimento foi utilizada uma máquina *desktop* com a seguinte configuração:

Tabela 9 – Configuração.

Configurações do Sistema	
Sistema Operacional	*Windows* 10 *Home Single Language*
Processador	Intel Core i3-5000U 2.00 GHz - 5ª Geração
Memória RAM	4,00 GB
Tipo de Sistema	Sistema Operacional de 64 bits, processador com base em x64
Informações da Internet	
Velocidade	5MB
Informações Adicionais	
Mouse e Teclado	

Fonte – Autora (2019).

5.4 Coleta e Análise dos Dados

Esta seção apresenta como foi efetuada a coleta e análise dos dados dessa validação. Para coletar os dados, foi realizada uma simulação com dados reais disponibilizados para testes pelo *Software* Público GP*Web*, no portal do *Software* Público. Estes dados serviram como base para realizar a interoperabilidade do *Software* Público GP*Web* com a plataforma FIWARE.

Para coletar os dados durante a tradução do arquivo do Banco de Dados do *Software* Público GP*Web* para a plataforma FIWARE, foi realizado um plano de testes utilizando o *plugin* de comparação da ferramenta Notepad++[6]. Com esses testes, buscou-se comparar os elementos caracteres, palavras e linhas. Essa comparação teve como objetivo analisar a confiabilidade dos dados, relacionado à **Questão de Validação 1**. Uma vez verificada uma falha, foi considerada como uma tradução inválida, relacionando-se à **Questão de Validação 2**.

Para promover maior confiabilidade dos dados, foram realizados 35 testes, sendo que cada teste equivale à consulta e execução de uma tabela distinta. Estas quantidades de testes foram relevantes para aprimorar a diversidade dos tipos de dados analisados (números, datas, textos, dentre outros), proporcionando maior confiabilidade.

Após a coleta dos dados, todos esses dados foram organizados na planilha *Microsoft Excel*[7]. Ademais, utilizou-se o *Excel*[8] para analisar os dados e elaborar os gráficos. Em seguida, com base na coleta dos dados, foram verificadas as ocorrências em relação aos acertos e erros dos elementos (caracteres, palavras e linhas). Por fim, foram elaborados os gráficos de acordo com os levantamentos propostos nas questões de validação.

A seguir os resultados da validação são apresentados como respostas às questões de validação, apresentadas na Seção 5.1 deste trabalho.

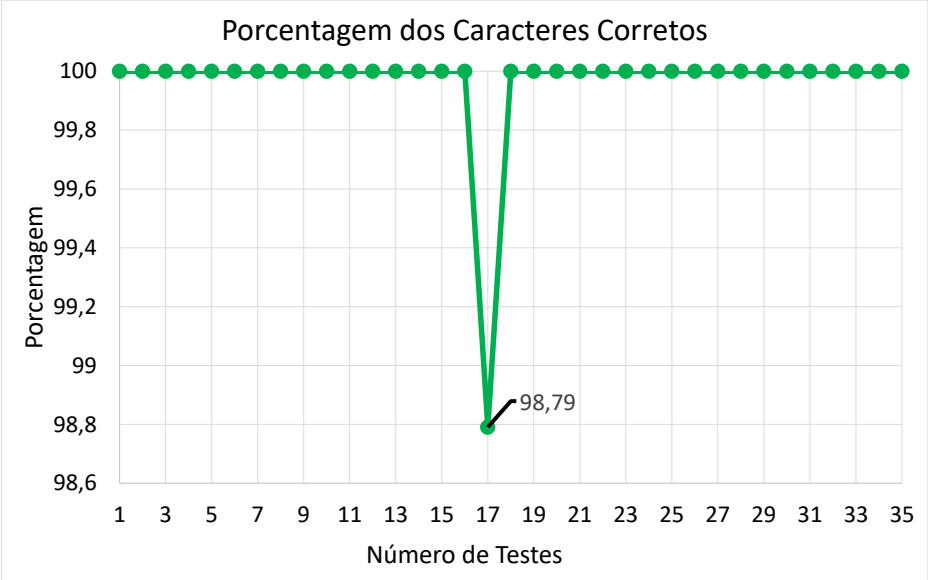
5.4.1 Análise dos Dados com relação à Questão de Validação 1

Com base na **Questão de Validação 1** e considerando especificamente a análise do elemento "**caractere**", pode-se inferir que usualmente o percentual relativo ao elemento "caractere" apresenta taxa de acerto mais próxima ou igual a 100%. Conforme pode ser observado na Figura 20, dos 35 testes analisados, somente o teste 17 não apresentou o percentual de 100% e sim de 98,79%. Com base nessa análise, pode-se inferir que a codificação do elemento "caractere" apresentou forma divergente de uma representação para a outra.

Considerando especificamente a análise do elemento "**linhas**", dos 35 testes analisados era esperado que em ao menos uma das linhas ocorresse erro, visto que, na análise realizada baseada nos caracteres um ou mais caracteres apresentaram falhas. Portanto, há de se inferir que

[6] <https://sourceforge.net/projects/npp-compare/>.
[7] <https://products.office.com/pt-br/excel>.
[8] <encurtador.com.br/jqFQR>.

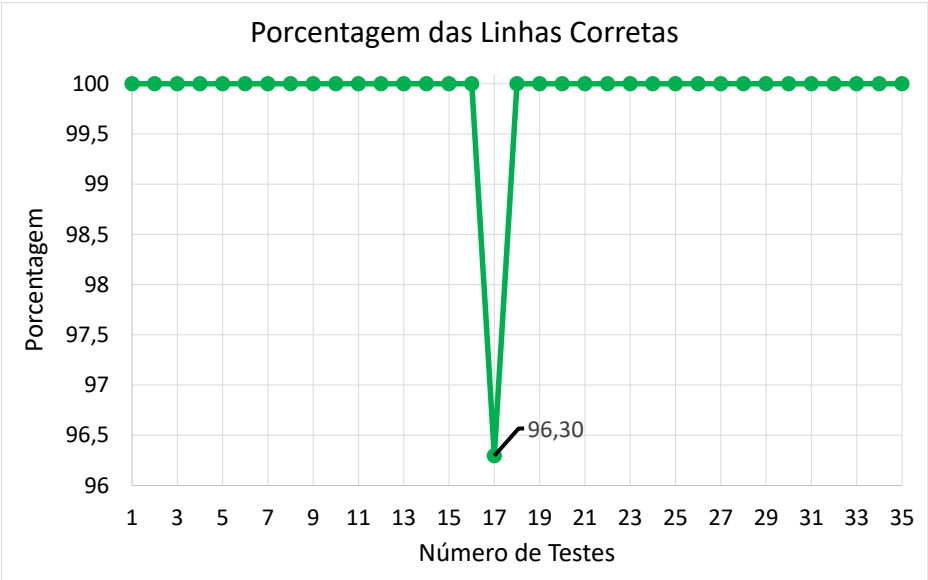
Figura 20 – Gráfico da Porcentagem dos Caracteres Corretos.



Fonte – Autora (2019).

existiria ao menos uma linha incorreta. Conforme pode ser observado na Figura 21, somente o teste 17 não apresentou o percentual de 100% e sim de 96,30%.

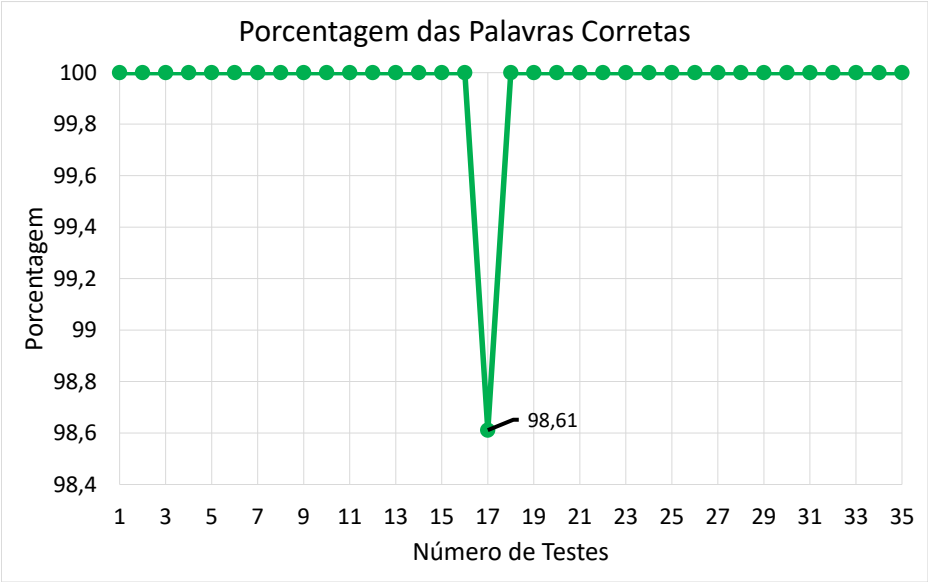
Figura 21 – Gráfico da Porcentagem das Linhas Corretas.



Fonte – Autora (2019).

Por fim, pelo fato de ter ocorrido erro nos elementos "linhas" e "caracteres", pode-se esperar que houvesse erro em ao menos um elemento "palavra". Desse modo, considerando especificamente a análise do elemento "**palavra**", dos 35 testes analisados, somente o teste 17 não apresentou o percentual de 100% e sim de 98,61%, conforme apresentado na Figura 22.

Figura 22 – Gráfico da Porcentagem das Palavras Corretas.



Fonte – Autora (2019).

Nas Figuras 20, 21 e 22, para cada teste foi utilizada a seguinte Fórmula 5.1.

$$P_C = \frac{E_C}{T_E} * 100 \tag{5.1}$$

em que, P_C = porcentagem dos elementos corretos; E_C = número de elementos corretos e T_E = número total de elementos corretos.

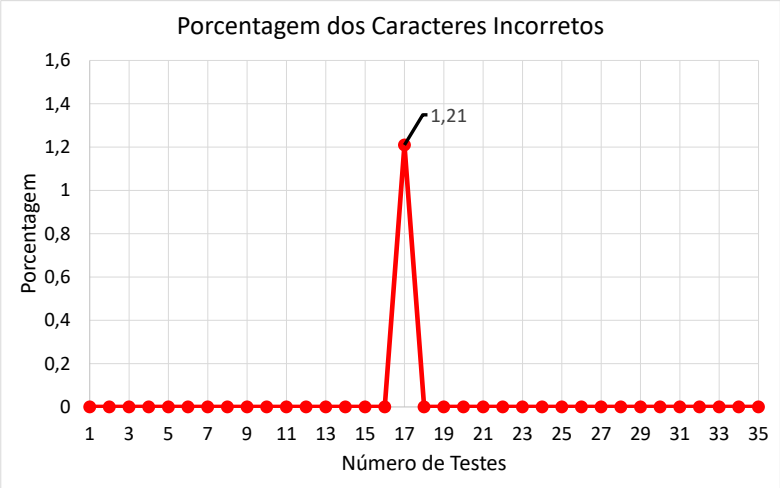
5.4.2 Análise dos Dados com relação à Questão de Validação 2

Com base na **Questão de Validação 2** e considerando especificamente a análise do elemento "**caracteres**", pode-se inferir que usualmente o percentual relativo ao elemento "caractere" apresenta taxa de acerto mais próximo ou igual a 0%. Conforme pode ser observado na Figura 23a, dos 35 testes analisados, somente o teste 17 não apresentou o percentual de 0% e sim de 1,21%.

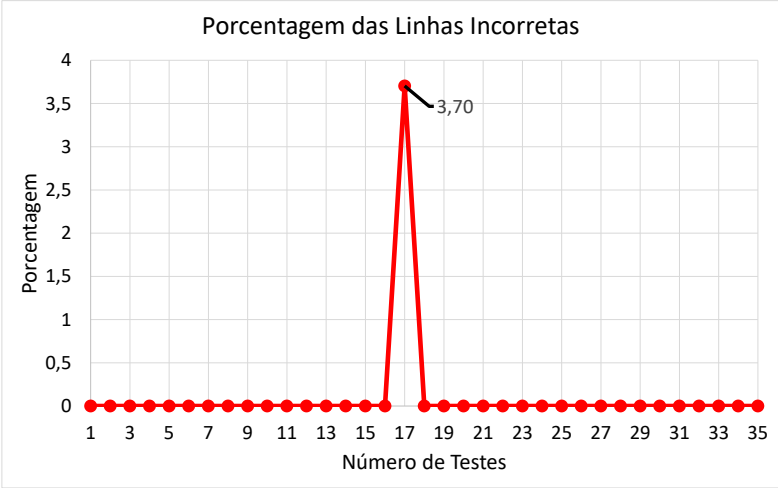
Considerando especificamente a análise do elemento "**linha**", dos 35 casos analisados, somente a amostra 17 não apresentou o percentual de 0% e sim de 3,70%, conforme apresentado na Figura 23b. Por fim, considerando especificamente a análise do elemento "palavra", dos 35 casos analisados, somente a amostra 17 não apresentou o percentual de 0% e sim de 1,39%, conforme apresentado na Figura 23c.

Baseado nessas análises, pode-se inferir que a codificação dos elementos "caracteres", "linhas" e "palavras" apresentaram formas divergentes de uma representação para a outra, conforme pode ser observado na Figura 23.

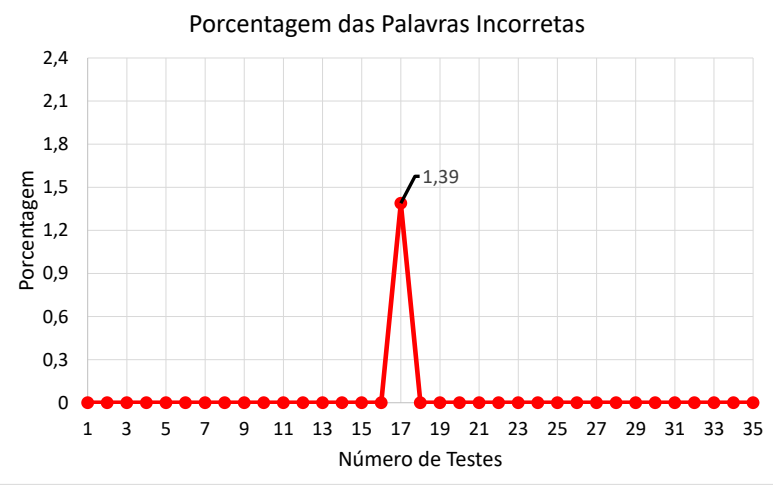
Figura 23 – Gráfico da Porcentagem dos Elementos Incorretos.



(a) Porcentagem dos Caracteres Incorretos.



(b) Porcentagem das Linhas Incorretas.



(c) Porcentagem das Palavras Incorretas.

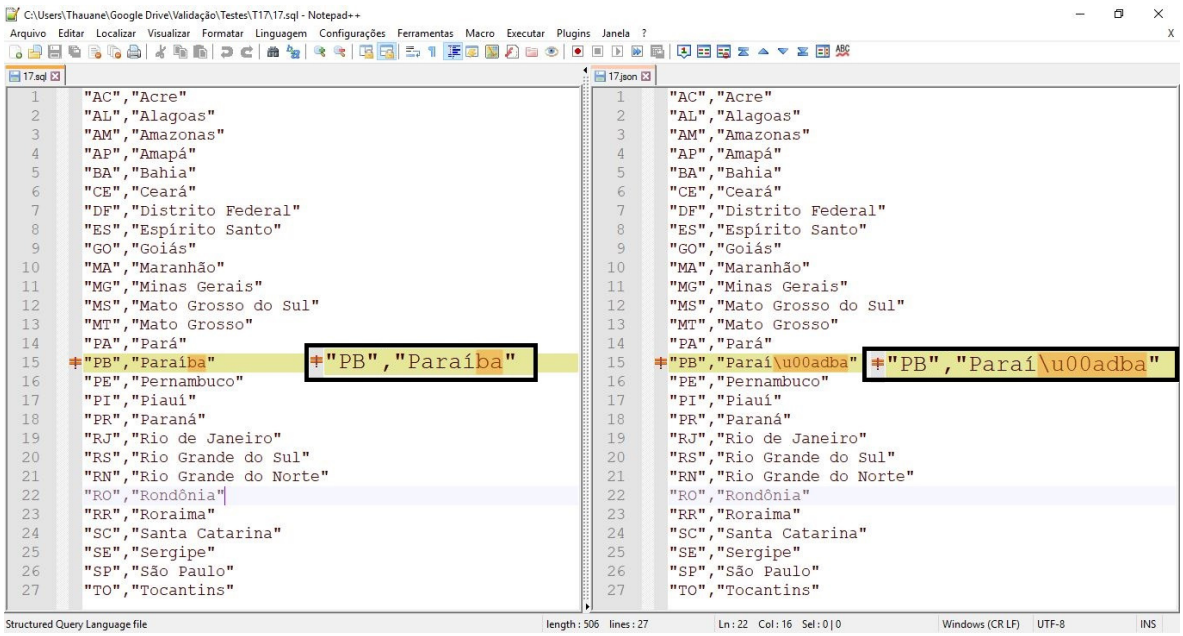
Na Figura 23, para cada teste realizado na análise da Questão de Validação 2, foi utilizada a seguinte Fórmula 5.2.

$$\boxed{P_I = \frac{E_I}{T_E} * 100} \tag{5.2}$$

em que, P_I = porcentagem de elementos incorretos; E_I = número de elementos incorretos e T_E = número total dos elementos incorretos.

Conforme mencionado na Figura 23, o teste 17 apresentou falhas nos elementos "caracteres", "linhas" e "palavras". Como pode ser observado na Figura 24, a comparação entre os arquivos SQL e JSON do teste 17, apresentou uma única falha em sua tradução **(linha 15)**. Esperava-se que nesta linha contivesse a palavra "Paraíba", porém, não houve equivalência entre as representações dos dados nos dois arquivos. A representação do formato JSON apresentou caracteres a mais em relação ao arquivo SQL, como pode ser notado na palavra "Paraí\u00adba".

Figura 24 – Erro do Teste 17.



Fonte – Autora (2019).

5.4.3 Análise dos Dados com relação à Questão de Validação 3

Inicialmente é importante compreender a metodologia adotada, para realizar o teste da questão de validação. Tendo em vista que o *Framework Smart* e-PING se comunica com a plataforma FIWARE por meio do protocolo HTTP, é possível por intermédio da análise de pacotes identificar e determinar quanto tempo leva para que a interoperabilidade seja concluída. Entretanto, é importante analisar o comportamento/processamento das requisições HTTP sob

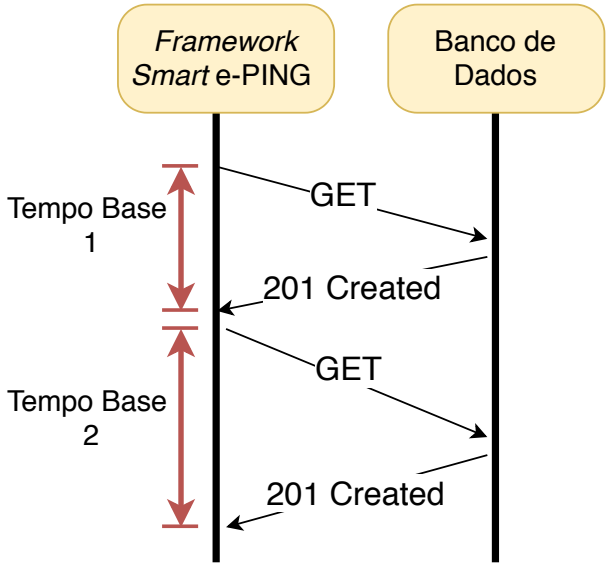
a perspectiva da camada de aplicação da arquitetura *Transmission Control Protocol/Internet Protocol* (TCP/IP).

Com a finalidade de calcular o tempo de requisição-resposta é importante observar que as requisições compõem-se de chamadas do método *GET*, aguardando respostas de pacotes do tipo *CREATE* que terão em seu conteúdo o formato JSON. Nesse ponto, é importante salientar que a camada de aplicação receberá diferentes pacotes referentes a uma mesma consulta realizada pelo *Framework Smart* e-PING. Desta forma, podem ser realizadas diferentes requisições *GET* e a camada de aplicação será responsável por remontá-las, obtendo assim a resposta completa da consulta.

Portanto, foi necessário realizar a análise do tráfego de pacotes da rede, por meio da ferramenta *Wireshark*. Utilizou-se essa ferramenta para capturar os pacotes que estavam associados ao IP do ORION, possibilitando a análise dos pacotes e consequentemente permitindo o cálculo do tempo das requisições e respostas utilizadas na interoperabilidade do *Framework Smart* e-PING para a plataforma FIWARE.

Como métrica, foi considerado o tempo base de cada requisição-resposta. Ou seja, a análise do tempo entre cada método *GET* e sua resposta, conforme apresentado na Figura 25. Desta forma, foi verificada a interação de cada requisição enviada. Após isso, foi analisado o intervalo total de tempo que a tradução ocorreu, desde o seu envio da consulta no Banco de Dados do *Software* Público até a sua chegada no ORION.

Figura 25 – Tempo Base.



Fonte – Autora (2019).

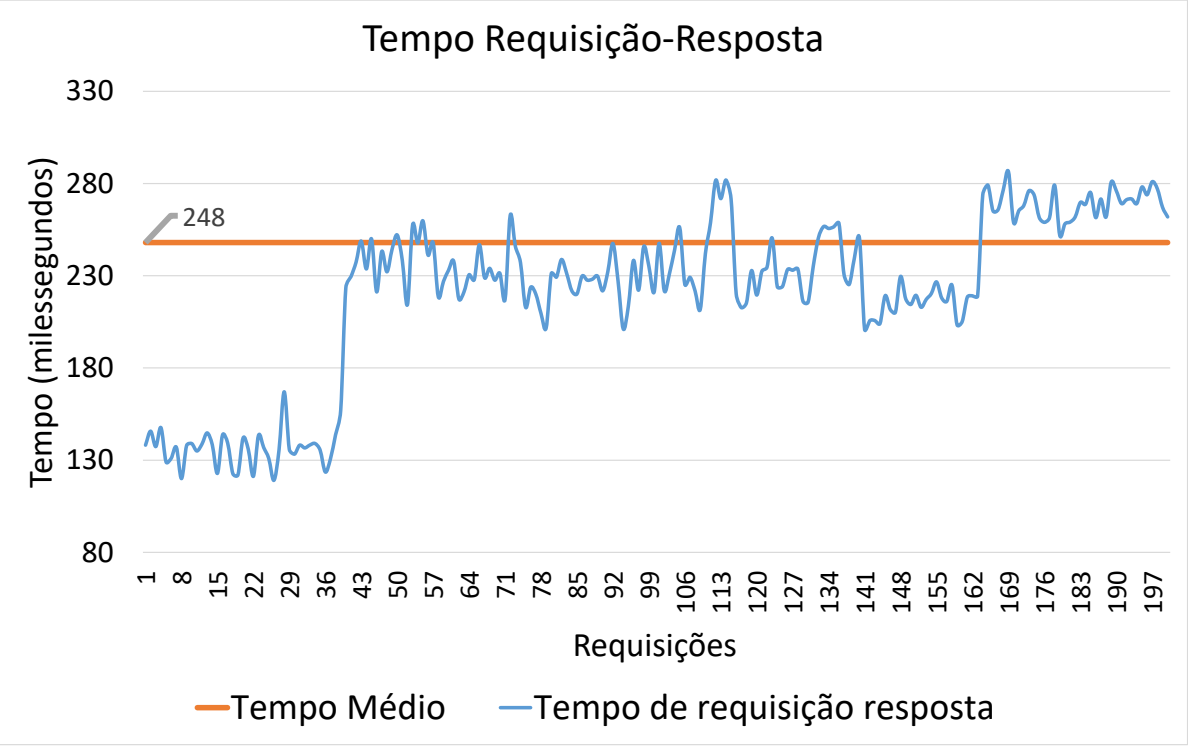
A fim de apresentar uma visão mais detalhada em relação ao comportamento das requisições e suas respostas observando o tempo destas transações, pode-se exemplificar por meio de uma tabela no banco de dados do *Software* Público que possui somente um registro e duas

colunas. Se solicitar o conteúdo desta tabela é necessário realizar uma solicitação do tipo *GET* por intermédio do protocolo HTTP, utilizado pelo *Framework Smart* e-PING. Como resposta, espera-se um *CREATE* que contenha o registro da tabela com conteúdo das duas respectivas colunas. Neste exemplo, somente foi necessário a execução de um *CREATE*, entretanto, se a tabela possuir N registros, serão necessários N *GETS* e N *CREATES*, a fim de obter a resposta contendo todos os registros desta tabela.

Vale salientar que cada consulta continha diferentes requisições. Ou seja, como foram realizadas 35 consultas e como as tabelas possuíam mais de um registro, houve neste caso a necessidade de realizar 197 requisições. Com base nestas requisições, pode-se analisar o tempo médio de cada *GET-CREATE*, sendo sabido que é comum ocorrer variações do tempo base para cada uma das requisições-respostas, como pode ser observado na Figura 26. Por conseguinte, o tempo médio em relação às 197 requisições analisadas foi de 248 milissegundos, apesar de que houve uma variação abrupta entre os pontos amostrais 36 e 43, em consequência de oscilações da latência da rede de comunicações de dados utilizada.

Portanto, pode-se inferir que no mínimo é esperado que uma consulta levará 248 milissegundos para compor uma única resposta que contenha um registro. Desse modo, uma consulta pode apresentar tempos diferentes a depender de cada requisição, visto que a consulta não necessariamente deve ter somente uma tabela que possua um único registro. Consequentemente, pode aumentar o tempo de resposta da consulta.

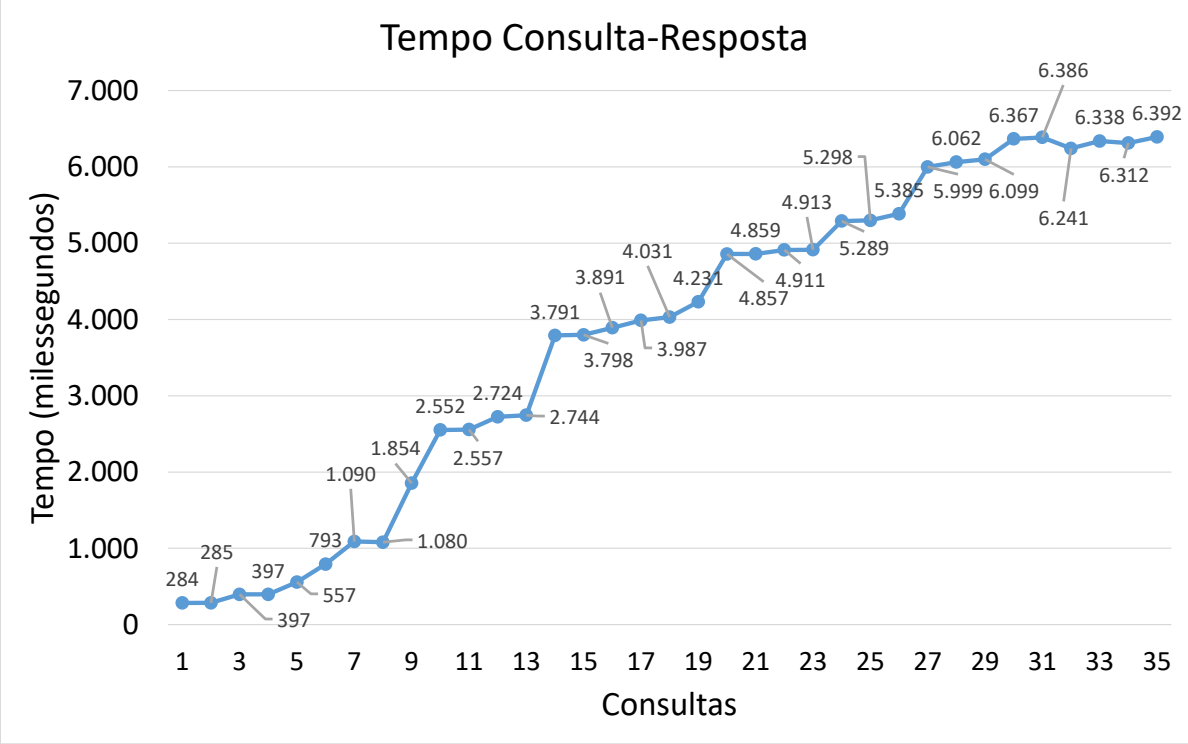
Figura 26 – Tempo Requisição-Resposta.



Fonte – Autora (2019).

Tendo como base o tempo médio apresentado na Figura 26, pode-se observar que a resposta de uma consulta pode variar de forma significativa, a depender da sua complexidade. Conforme pode ser observado, as consultas apresentam tempos de respostas diferentes uma das outras, visto que o grau de complexidade e o número de tabelas consultadas variam. Portanto, é necessário o envio de diferentes pacotes na rede para uma mesma requisição, resultando em tempos de respostas diferentes.

Figura 27 – Tempo Consulta-Resposta.



Fonte – Autora (2019).

Desse modo, pode-se inferir quais são os tempos que cada consulta irá levar para ser respondida e adequada ao arquivo JSON a partir de um Banco de Dados de um *Software* Público, conforme descrito na Figura 27. Em questão, pode-se observar que a complexidade da consulta influi diretamente no tempo de resposta. Considerando a consulta um e a consulta 35, nota-se que há uma diferença considerável de tempo de resposta, isso se dá devido a necessidade de realizar diferentes métodos *GET* e *CREATE* para a consulta 35, uma vez que esta possui 96 registros. Porém, em contraste com a consulta 35, a consulta 1 possui apenas 2 registros, em que consequentemente levará menor quantidade de tempo uma vez que será necessário somente um *GET* e um *CREATE*.

5.5 Discussão das Questões de Validação das Métricas Utilizadas

A base da análise dos testes foi pautada em três questões de validação, que foram: questão de validação 1 (tradução bem-sucedida na Seção 5.4.1), questão de validação 2 (detecção de falhas na Seção 5.4.2) e questão de validação 3 (tempo na Seção 5.4.3). Do aspecto formal em relação a essas questões, foram levantadas por meio das 35 consultas os pontos constantes entre essas questões.

Quanto à **questão de validação 1**, pode-se inferir que houve uma taxa de sucesso de 97,14%, uma vez que somente uma consulta apresentou incorreção quanto à tradução. Desse modo, observando especificamente este parâmetro e as configurações adotadas para a realização deste teste, conclui-se que o *Framework Smart* e-PING apresentou resultado "muito bom" nas "**tentativas de sucesso na tradução dos dados**". Esses resultados foram baseados segundo os critérios da Tabela 6 na Seção 5.2.

De forma análoga, observando a **questão de validação 2**, espera-se que haja valor de erro maior que 0%. Uma vez que as questões de validação 1 e 2 são complementares. Ou seja, a incompletude da não observância da margem de 100% da questão de validação 1, implica a presença de erro na questão de validação 2, como foi observado na Figura 23. Desse modo, pode-se classificar dentro dos parâmetros definidos na questão de validação 2 que o *Framework Smart* e-PING apresentou resultados "muito bom", com margem de erro de 2,85%. Esses resultados foram baseados segundo os critérios da Tabela 7 na Seção 5.2.

Por fim, a **questão de validação 3** buscou apresentar um panorama quanto ao tempo gasto em uma consulta. Desse modo, observou-se que o tempo médio de cada iteração é de 248 milissegundos. Por conseguinte, a depender do número de registros de uma determinada tabela, o tempo despendido para efetuar uma resposta vai crescer basicamente de forma proporcional.

Então, em observância aos testes realizados e as questões de validação que foram levantadas, pode-se observar que o *Framework Smart* e-PING apresenta, dentro desses condicionamentos, os resultados que atendem o objetivo da validação proposto neste trabalho, apesar de existirem pontos que podem ser aperfeiçoados. A seguir serão detalhados esses aperfeiçoamentos, bem como as dificuldades e limitações encontradas durante a validação.

5.6 Limitações e Dificuldades da Validação

No decorrer desta validação, algumas limitações foram encontradas:

a) Devido aos critérios de gerenciamento de redes que a UFS possui, por questões de segurança, a rede da UFS não permite a utilização do IP do ORION. Desse modo, como alternativa foi realizado o uso de uma máquina própria voltada para este propósito.

b) Devido a problemas nas máquinas da UFS, não foi possível utilizar o *Aedes Tracker*, uma vez que o servidor em que se encontrava o *Aedes Tracker* foi formatado sem prévio aviso. Desse modo, como alternativa foi validado com IP do ORION disponibilizado por um aluno para realizar testes. Portanto, qualquer aplicação seja *Aedes Tracker* ou qualquer outra aplicação, pode realizar a interoperabilidade de dados.

c) Durante a validação identificou-se falhas ao realizar uma requisição ao ORION contendo "strings" com caracteres especiais, como "(", ")", "<", ">". A solução deste erro não foi identificada nas pesquisas referentes nem ao ORION e nem ao JSON. Ademais, não foram encontradas soluções deste erro em páginas *Web*. Desse modo, optou-se por substituir estes caracteres por seus valores hexadecimais da tabela ASCII. Porém, não é possível utilizar um número hexadecimal, pois ao fazer isto o receptor da informação entenderia que este número poderia fazer parte do contexto (o que não é verdade). Portanto, decidiu-se padronizar a substituição de um caractere, para que as aplicações externas que recebam a informação possam retomar o valor original do hexadecimal, ficando da seguinte forma a substituição:

   a) "string" original: "Valor de um (".

   b) "string" substituída: "Valor de um .&28."

5.7 Ameaças à Validade

Por mais que se planeje e conduza, em uma validação sempre haverá ameaças à validade (WOHLIN et al., 2012). Ao final da execução desta validação foram detectadas algumas ameaças à validade:

a) **Validade interna**: como a validação foi realizada pelos autores, pode-se ter introduzido viés na seleção das métricas. Para mitigar esse tipo de problema, o processo de seleção foi revisado por três pesquisadores mais experientes, sendo dois doutores e um mestre, ambos em computação.

b) **Validade externa**: como a validação foi realizada pelos autores, pode-se influenciar os resultados da validação. Para mitigar esse tipo de problema, a validação foi realizada conforme as métricas definidas pela norma ISO/IEC 25010.

c) **Validade de conclusão**: possibilidade de ter ocorrido viés no processo de análise de dados, pelos autores. Para mitigar esse tipo de problema, a análise dos dados foi revisada por outro pesquisador mais experiente. Outra ameaça referente aos resultados é a realização da análise de dados, que ocorreu em uma única máquina evitando que alterações e configurações computacionais comprometessem os valores coletados na validação.

6
Conclusão

Neste capítulo é apresentada a conclusão desta dissertação, juntamente com os objetivos, metodologia utilizada, limitações dos resultados e os trabalhos que podem dar continuidade a esta pesquisa.

O Governo Eletrônico Brasileiro (GEB) aplicado ao Governo Federal Brasileiro vem envolvendo o uso de tecnologias para prestar serviços ao público com objetivo de melhorar a prestação desses serviços e reforçar o envolvimento dos cidadãos nos serviços públicos. Para aprimorar esses serviços, o GEB abrange a integração dos sistemas heterogêneos para reduzir o acesso a múltiplos órgãos do governo a fim de obter um único serviço, facilitando o acesso do cidadão aos serviços públicos. Portanto, o GEB criou um padrão de interoperabilidade denominado e-PING para oferecer esses serviços.

Além dessa iniciativa, o GEB está no processo de construção de Cidades Inteligentes, obtendo maior eficiência nas atividades dos Governos Federais que envolvem a gestão das cidades. Nesse sentido, surgem iniciativas governamentais com o intuito de tornar as cidades mais inteligentes.

Dessa forma, uma plataforma para Cidades Inteligentes auxilia a criação e a integração de aplicações desenvolvidas para Cidades Inteligentes. Diante dessa situação, para integrar sistemas heterogêneos é necessário utilizar um elemento mediador como um *middleware*. O *middleware* mais utilizado e citado foi a plataforma aberta FIWARE.

Dessa forma, o objetivo deste trabalho foi desenvolver o *Framework Smart* e-PING, que permitisse a interoperabilidade entre um *Software* Público que adere ao uso da e-PING e a plataforma FIWARE. A justificativa para a realização deste trabalho se deu pela necessidade da integração dos recursos de Cidades Inteligentes, facilitando o uso da plataforma para os usuários da e-PING. Portanto, os usuários não necessitarão de um prévio conhecimento da FIWARE, resta somente entender a interface do *Framework Smart* e-PING para efetuar o seu uso.

Como forma de cumprir o objetivo deste trabalho, foi estabelecida uma metodologia de pesquisa, que consistiu em oito atividades, que delimitaram o problema proposto. Vale destacar que houve a validação do *Framework Smart* e-PING que se tratou de uma das atividades que contemplaram a pesquisa.

Pode-se observar que o objetivo final foi parcialmente alcançado, por meio da junção dos três objetivos: (1) Investigar por meio de uma revisão sistemática a utilização de outras plataformas de IoT abertas voltadas para Cidades Inteligentes. Esse objetivo buscou identificar plataformas que foram utilizadas com maior frequência no desenvolvimento de aplicações IoT ou em CI, conforme descrito no Capítulo 3. E conforme descrito neste capítulo, foram analisadas essas plataformas comparando cada uma delas com os requisitos de *middleware* (apresentados na Seção 2.6 do Capítulo 2). Portanto, concluiu-se que a FIWARE atende todos os requisitos levantados.

(2) Desenvolver um *Framework* interoperável entre um *Software* Público e a plataforma FIWARE. Este objetivo foi descrito no Capítulo 4, porém foi parcialmente atendido, devido às dificuldades e ausência de tempo descritas na Seção 6.1. Para o desenvolvimento do *Framework Smart* e-PING, a sua estratégia de comunicação foi dividida em duas etapas complementares. A primeira etapa consistiu na interoperabilidade do *Software* Público para a plataforma FIWARE. Enfim, a segunda etapa consistiu na interoperabilidade da plataforma FIWARE para o *Software* Público.

Por fim, (3) Validar o *Framework Smart* e-PING por meio da execução de um processo real de um *Software* Público que adere ao e-PING com a plataforma FIWARE, conforme apresentado na Seção 5. Essa validação teve como finalidade analisar a interoperabilidade proposta do *Framework Smart* e-PING, com objetivo de obter conclusões importantes a partir dos 35 testes realizados. Portanto, esses testes analisados conseguiram avaliar positivamente a proposta do *Framework Smart* e-PING.

À vista disso, a pergunta-chave que foi definida no Capítulo 1, para direcionar a pesquisa é respondida: **Como fornecer a interoperabilidade entre um *Software* Público baseado no padrão e-PING com a plataforma FIWARE?** Para fornecer essa interoperabilidade é necessário utilizar um elemento mediador, chamado *middleware*, capaz de acessar/mediar tecnologias distintas passando a se comunicar independente de protocolos, sistemas operacionais, dentre outros. Outrossim, um *middleware* suporta arquiteturas por meio de métodos orientados a processos, permitindo que dados possam ser movidos de uma aplicação para outra.

Dessa maneira, também é possível responder a hipótese apresentada no Capítulo 1. Uma hipótese nula H_0 é apresentada e testada "Utilizando o *Framework Smart* e-PING, não será possível a interoperabilidade entre um *Software* Público que adere ao e-PING e a plataforma FIWARE". Caso sejam encontrados resultados diferentes, deve-se rejeitar a hipótese nula H_0 e aceitar a hipótese alternativa H_1 "Utilizando o *Framework Smart* e-PING, será possível a interoperabilidade entre entre um *Software* Público que adere ao e-PING e a plataforma

FIWARE". Portanto, é possível realizar a interoperabilidade por meio do *Framework Smart* e-PING. Logo, conclui-se que deve rejeitar a hipótese nula.

Este trabalho visou de fato integrar um *Software* Público que adere ao e-PING e a plataforma FIWARE, por meio do *Framework Smart* e-PING. Por conseguinte, *Framework Smart* e-PING auxilia ao GEB a poupar esforços e tempo, não necessitando de prévio conhecimento da FIWARE, entendendo somente a interface do *Framework Smart* e-PING para efetuar o seu uso. Ademais, o *Framework Smart* e-PING otimiza e melhora os seus serviços públicos por meio de soluções inteligentes e a reutilização de recursos inteligentes. Portanto, o *Framework Smart* e-PING permite a integração e facilita o uso da plataforma aberta FIWARE para as organizações que já utilizam a Arquitetura com Padrões de Interoperabilidade do Governo Brasileiro (e-PING).

Vale ainda ressaltar que um artigo científico proveniente do início deste trabalho, foi premiado como o segundo melhor artigo da trilha WPOS no ERBASE 2017 (Anexo A).

Por fim, com relação a essa pesquisa, pretende-se, com este trabalho, ampliar e diversificar o conjunto de poucos trabalhos existentes no assunto. Outrossim, colaborar para o avanço de novas aplicações na área do Governo Eletrônico Brasileiro e Cidades Inteligentes.

6.1 Dificuldades e Limitações

No decorrer deste trabalho, algumas dificuldades e limitações foram encontradas. Devido à necessidade de proteção das suas informações, as organizações apesar de fazer uso de *Softwares* Públicos que aderem ao e-PING, não disponibilizaram o acesso de fato aos seus Bancos de Dados. Todavia, para a realização da validação foi necessário utilizar a Base de Dados de testes disponibilizado pelo GP*Web*, no portal do *Software* Público.

Devido à manutenção da FIWARE, houve problema de disponibilidade de vinte dias. Durante esse período, não foi possível efetuar testes e consequentemente, proporcionou atrasos em relação ao cronograma desta dissertação.

Devido aos problemas que surgiram durante a implementação deste trabalho, não houve tempo hábil para realizar a implementação das *triggers*. Portanto, esta implementação ficará como trabalhos futuros. Com a implementação dessas *triggers* tanto as aplicações que utilizam um *Software* Público quanto as aplicações FIWARE poderão interagir com os dados em tempo real, devido ao *Framework Smart* e-PING.

Implementar *triggers* para a realização das atualizações das informações em tempo real tanto na base ORION quanto na Base de Dados do *Software* Público. Com essa implementação tanto as aplicações que utilizam um *Software* Público quanto às aplicações FIWARE poderão interagir com os dados em tempo real, sem quaisquer alterações no código fonte, devido ao *Framework Smart* e-PING.

Por fim, houve outra dificuldade em encontrar trabalhos relacionados na literatura, com a

proposta de um *Framework* similar ao *Framework Smart* e-PING. A construção do *Framework Smart* e-PING, neste aspecto, foi inovadora. E esta inovação já traz consigo uma contribuição para novos estudos na área.

6.2 Trabalhos Futuros

Nesta seção são apresentadas sugestões de melhorias e continuidade para os trabalhos futuros, que podem dar sequência a este trabalho. São elas:

Só explicar que existem alguns pontos a serem melhorados assim como podem ser adicionadas novas funcionalidades

a) Implementar a interoperabilidade do ORION para o *Software* Público. Com essa implementação um usuário por meio do aplicativo externo, que utiliza a FIWARE, poderá realizar inserções dos dados no Banco de Dados do *Software* Público. Portanto, ocorrerá a interoperabilidade bilateral.

b) Implementar novos módulos para que o *Framework Smart* e-PING possa ser utilizado por todos os bancos de dados. Com essa implementação qualquer *Software* Público que estiver utilizando outro Banco de Dados que não seja o MySQL, pode realizar a interoperabilidade de seu *Software* Público com a plataforma FIWARE.

c) Implementar *triggers* para a realização das atualizações das informações em tempo real tanto na base ORION quanto na Base de Dados do *Software* Público. Com essa implementação tanto as aplicações que utilizam um *Software* Público quanto às aplicações FIWARE poderão interagir com os dados em tempo real, sem quaisquer alterações no código fonte, devido ao *Framework Smart* e-PING.

d) Validar a interoperabilidade do *Framework Smart* e-PING em um(a) máquina/servidor com configurações superiores ao que foi apresentado no Capítulo 5. Com essa validação, pretende-se verificar que o *Framework Smart* e-PING poderá gerar resultados ainda melhores.

e) Validar a usabilidade do *Framework Smart* e-PING. Com essa validação, pretende-se verificar a capacidade de utilização do *framework* e medida do esforço necessário para que o *Framework Smart* e-PING possa ser utilizado dentro de suas prescrições.

f) Inserir o *Framework Smart* e-PING em um domínio de governo eletrônico na FIWARE, ou seja, disponibilizar habilitadores genéricos para governo eletrônico na plataforma FIWARE. Com essa inserção, os governos eletrônicos de diferentes países poderão integrar automaticamente suas plataformas. Porém, para inserir um domínio de governo eletrônico na FIWARE, o Governo Eletrônico Brasileiro tem que entrar em contato com o serviço da

FIWARE. A FIWARE irá informar uma série de requisitos de servidores e instalações a serem cumpridos. Logo após, o Governo Eletrônico Brasileiro precisará apresentar todos esses requisitos para poder entrar no domínio da FIWARE.

g) Utilização de dados abertos em colaboração com o *Framework Smart* e-PING. Com essa colaboração, os cidadãos de uma determinada sociedade saberão o que seu governo está fazendo, tornando-se transparente. Diante disso, qualquer usuário poderá acessar livremente os dados e as informações do governo e compartilhá-las com outros cidadãos.

6.3 Publicações Relacionadas à Dissertação

Nesta seção são apresentadas as contribuições relacionadas a dissertação.

6.3.1 Artigos Aprovados

a) *Public ICT Governance: A Quasi-systematic Review - In Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS), Volume 2, Porto, Portugal, April 26-29, 2017* - Qualis B1[1].

b) Proposta de Interoperabilidade da e-PING para Cidades Inteligentes Utilizando a FIWARE. Erbase 2017 - WPOS[2] - Possível publicação em revista B3. - Artigo científico proveniente do início deste trabalho e foi premiado como o segundo melhor artigo da trilha WPOS (Anexo A).

6.3.2 Artigo Submetido

a) *Smart* e-PING: *Framework* de Interoperabilidade da Arquitetura com a Plataforma FIWARE para o uso em Cidades Inteligentes. Submetido para Revista Isys[3] - Qualis B3.

[1] Disponível em: <https://www.scitepress.org/papers/2017/63146/63146.pdf>.
[2] Disponível em: <encurtador.com.br/xDI35>.
[3] Disponível em: <http://www.seer.unirio.br/index.php/isys/>.

Referências

ABERER, K.; HAUSWIRTH, M.; SALEHI, A. Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks. p. 198–205, may 2007. Conference and Custom Publishing. Citado na página 50.

AHSAN, M. et al. Ensuring Interoperability Among Heterogeneous Devices through IoT Middleware. *International Journal of Computer Science and Information Security (IJCSIS), Vol. 14, No. 4, April 2016*, v. 14, p. 251–256, 2016. Citado 2 vezes nas páginas 50 e 60.

AIELLO, M. et al. Smart Homes to Improve the Quality of Life for All. 33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2011, August 30 - Sept. 3, 2011, Boston, MA, USA, p. 1777–1780, 2011. IEEE. Citado na página 61.

AL-FUQAHA, A. I. et al. Toward Better Horizontal Integration Among IoT Services. *IEEE Communications Magazine*, v. 53, n. 9, p. 72–79, 2015. Citado na página 47.

AL-JAROODI, J. et al. Security Middleware Approaches and Issues for Ubiquitous Applications. *Computers & Mathematics with Applications*, Elsevier, v. 60, n. 2, p. 187–197, 2010. Citado na página 44.

ALBINO, V.; BERARDI, U.; DANGELICO, R. M. Smart Cities: Definitions, Dimensions, Performance, and Initiatives. *Journal of Urban Technology*, v. 22, n. 1, p. 3–21, 2015. Citado na página 33.

ALONSO, Á. et al. Industrial Data Space Architecture Implementation Using FIWARE. *Sensors*, v. 18, n. 7, p. 2226, 2018. Citado na página 40.

ALONSO, G. et al. *Web Services: Concepts, Architectures and Applications*. 1. ed. [S.l.]: Springer; Softcover Reprint of Hardcover, 2004. v. 1. 1–354 p. (Data-Centric Systems and Applications, v. 1). Citado 3 vezes nas páginas 35, 36 e 37.

ALVES, A. M.; PESSÔA, M. S. de P.; SALVIANO, C. F. Towards a Systemic Maturity Model for Public Software Ecosystems. In: SPRINGER. *International Conference on Software Process Improvement and Capability Determination - 11th International Conference, SPICE 2011, Dublin, Ireland, May 30 - June 1, 2011. Proceedings*. Berlin, Heidelberg, 2011. p. 145–156. Citado na página 32.

AMARAL, L. A. et al. Middleware Technology for IoT Systems: Challenges and Perspectives Toward 5G. In: *Internet of Things (IoT) in 5G Mobile Technologies*. [S.l.]: Springer, 2016. p. 333–367. Citado na página 43.

ANGGORJATI, B. et al. RFID Added Value Sensing Capabilities- European Advances in Integrated RFID-WSN Middleware. *SECON 2010 - 2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, p. 1–3, 2010. Citado na página 47.

ARAUJO, M. H. de; REINHARD, N. Categorization of Brazilian Internet Users and Its Impacts on the Use of Electronic Government Services. In: *Electronic Government - 13th IFIP WG*

8.5 International Conference, EGOV 2014. Dublin, Ireland: Springer, 2014. (Lecture Notes in Computer Science, v. 8653), p. 242–252. Citado na página 16.

ARMANDO, N. et al. WSNs in FIWARE–Towards the Development of People-Centric Applications. In: SPRINGER. *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Cham, 2017. p. 445–456. Citado na página 38.

ASPIRE. 2018. Disponível em: <http://www.fp7-aspire.eu/>. Citado 2 vezes nas páginas 46 e 60.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A Survey. *Computer Networks*, v. 54, n. 15, p. 2787–2805, oct 2010. Citado 5 vezes nas páginas 32, 41, 42, 43 e 44.

AZIZ, K. et al. Smart Real-Time Healthcare Monitoring and Tracking System Using GSM/GPS Technologies. In: *2016 3rd MEC International Conference on Big Data and Smart City, ICBDSC 2016*. Muscat, Oman: Institute of Electrical and Electronics Engineers Inc., 2016. p. 357–363. Citado na página 34.

BABU, K. R. R.; GEORGE, S. J.; SAMUEL, P. Optimal Sensor Selection from Sensor Pool in IoT Environment. In: *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*. Tumkuru, Karnataka, India: IEEE, 2016. p. 697–702. Citado na página 50.

BAKICI, T.; ALMIRALL, E.; WAREHAM, J. A Smart City Initiative: The Case of Barcelona. *Journal of the Knowledge Economy*, v. 4, n. 2, p. 135–148, 2013. Citado na página 34.

BANDYOPADHYAY, S. et al. Role of Middleware for Internet of Things: A Study. *International Journal of Computer Science & Engineering Survey (IJCSES)*, Academy & Industry Research Collaboration Center (AIRCC), v. 2, n. 3, p. 94–105, 2011. Citado 8 vezes nas páginas 36, 37, 42, 43, 44, 46, 58 e 62.

BARBOSA, S. A. A. et al. An Architecture Proposal for the Creation of a Database to Open Data Related to ITS in Smart Cities. In: IEEE. *Proceedings of the 8th Euro American Conference on Telematics and Information Systems (EATIS 2016)*. Colombia, 2016. v. 1, n. 4, p. 1–7. Citado na página 28.

BARRETO, L. et al. Identity management in IoT Clouds: A FIWARE Case of Study. In: *2015 IEEE Conference on Communications and Network Security, (CNS)*. Florence, Italy: Institute of Electrical and Electronics Engineers Inc., 2015. p. 680–684. Citado na página 60.

BARROS, A.; CEPIK, M. A. C.; CANABARRO, D. R. Para Além da e-Ping: o Desenvolvimento de uma Plataforma de Interoperabilidade de e-Serviços no Brasil. *Panorama da interoperabilidade no Brasil. Brasília. Ministério do Planejamento, Orçamento e Gestão*, p. 137–157, 2010. Citado na página 30.

BASILI, V. R. et al. GQM+Strategies: A Comprehensive Methodology for Aligning Business Strategies with Software Measurement. *CoRR*, abs/1402.0292, 2014. Citado na página 84.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice*. 3. ed. Boston: United States of America: Addison-Wesley Professional, 2012. Citado na página 45.

BĂTĂGAN, L. Smart Cities and Sustainability Models. *Revista de Informática Economică*, v. 15, n. 3, p. 80–87, 2011. Citado na página 16.

BATISTA et al. SmartMetropolis - Plataformas e Aplicações para Cidades Inteligentes. p. 47, 2016. Disponível em: <http://smartmetropolis.imd.ufrn.br/wp-content/uploads/2016/05/RT1-WP5.pdf>. Citado 4 vezes nas páginas 43, 59, 60 e 62.

BEHERA, R.; REDDY, K. H. K.; ROY, D. Reliability Assessment of Energy Monitoring Service for a Futuristic Smart City. In: *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies (ICTCS)*. Udaipur, India: ACM International Conference Proceeding Series, 2016. v. 4, p. 1–6. Citado na página 34.

BLAIR, G. S. et al. The Role of Ontologies in Emergent Middleware: Supporting Interoperability in Complex Distributed Systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 7049 LNCS, p. 410–430, 2011. Citado na página 18.

BOHN, H.; BOBEK, A.; GOLATOWSKI, F. SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service Oriented Framework for Different Domains. In: *Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, (ICNICONSMCL'06)*. Morne, Mauritius, Mauritius: IEEE, 2006. v. 6, p. 43–49. Citado na página 55.

BOLÍVAR, M. P. R.; LÓPEZ-QUILES, J. M. The Quest for the Quality of Life in European Smart Cities: An Empirical Research. In: ACM. *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age, DG.O*. Delft,The Netherlands, 2018. p. 24:1–24:15. Citado 2 vezes nas páginas 18 e 34.

BOOCH, G. The Economics of Architecture-First. *IEEE Software*, v. 24, n. 5, p. 18–20, 2007. Citado na página 45.

BORGIA, E. The Internet of Things Vision: Key Ffeatures, Applications and Open Issues. *Computer Communications*, v. 54, p. 1–31, 2014. Citado 2 vezes nas páginas 32 e 35.

BRASIL. *EPING Padrões de Interoperabilidade de Governo Eletrônico*. 2007. Disponível em: <www.governoeletronico.gov.br>. Citado na página 30.

BRASIL. Manual dos Dados Abertos: Governo. 2011. Disponível em: <http://www.w3c.br/pub/Materiais/PublicacoesW3C/Manual_Dados_Abertos_WEB.pdf>. Citado na página 37.

BRASIL. *EPING Padrões de Interoperabilidade de Governo Eletrônico - Documento de Referência*. 2016. Disponível em: <http://governoeletronico.gov.br/acoes-e-projetos/e-ping-padroes-de-interoperabilidade>. Citado 4 vezes nas páginas 19, 30, 31 e 32.

BRIZZI, P. et al. Bringing the Internet of Things Along the Manufacturing Line: A Case Study in Controlling Industrial Robot and Monitoring Energy Consumption Remotely. In: IEEE. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*. Torino, Italy, 2013. p. 1–8. Citado 2 vezes nas páginas 48 e 49.

BROGAN, J. P.; THUEMMLER, C. Specification for Generic Enablers as Software. In: IEEE COMPUTER SOCIETY. *Information Technology: New Generations (ITNG), 2014 11th International Conference on*. Edinburgh, United Kingdom, 2014. p. 129–136. Citado na página 17.

CALBIMONTE, J. et al. XGSN: An Open-source Semantic Sensing Middleware for the Web of Things. In: *Joint Proceedings of the 6th International Workshop on the Foundations, Technologies and Applications of the Geospatial Web, TC 2014, and 7th International Workshop on Semantic Sensor Networks, SSN 2014, co-located with 13th International Semantic Web Conference (ISWC 2014)*. Trentino, Italy: CEUR Workshop Proceedings, 2014. v. 1401, p. 51–66. Citado na página 50.

CALDIERA, V. R. B.-G.; ROMBACH, H. D. Goal Question Metric Paradigm. *Encyclopedia of software engineering*, v. 1, p. 528–532, 1994. Citado na página 84.

CANNATA, A.; GEROSA, M.; TAISCH, M. SOCRADES: A Framework for Developing Intelligent Systems in Manufacturing. In: IEEE. *IEEE International Conference on Industrial Engineering and Engineering Management, 2008. IEEM 2008*. [S.l.], 2008. p. 1904–1908. Citado na página 56.

CAPES. QUALIS: Concepção e Diretrizes Básicas. *Revista Brasileira Pós-Graduação*, p. 149–151, 2010. Citado na página 20.

CAPORUSCIO, M.; RAVERDY, P.; ISSARNY, V. ubiSOAP: A Service-Oriented Middleware for Ubiquitous Networking. *IEEE Trans. Services Computing*, v. 5, n. 1, p. 86–98, 2012. Citado na página 57.

CAPORUSCIO, M. et al. ubiSOAP: A Service Oriented Middleware for Seamless Networking. In: *International Conference on Service-Oriented Computing - ICSOC*. Sydney, Australia: [s.n.], 2008. p. 195–209. Citado na página 57.

CAPORUSCIO, M.; RAVERDY, P.-G.; ISSARNY, V. UbiSOAP: A Service-Oriented Middleware for Ubiquitous Networking. *IEEE Transactions on Services Computing*, v. 5, n. 1, p. 86–98, 2012. Citado na página 62.

CARRIOTS. 2018. Disponível em: <https://www.carriots.com/>. Citado 2 vezes nas páginas 47 e 60.

CATARCI, T. et al. Smart Homes for All: Collaborating Services in a for-All Architecture for Domotics. In: SPRINGER. *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Orlando, FL, USA, 2009. p. 56–69. Citado na página 61.

CATARINUCCI, L. et al. An IoT-Aware Architecture for Smart Healthcare Systems. *IEEE Internet of Things Journal*, v. 2, n. 6, p. 515–526, 2015. Citado na página 49.

CAVALCANTE, E. et al. An Analysis of Reference Architectures for the Internet of Things. In: ACM. *Proceedings of the 1st International Workshop on Exploring Component-based Techniques for Constructing Reference Architectures*. Montreal, QC, Canada, 2015. p. 13–16. Citado 4 vezes nas páginas 17, 19, 42 e 43.

CERI, S.; COCHRANE, R.; WIDOM, J. Practical applications of triggers and constraints: Successes and lingering issues. In: . [S.l.: s.n.], 2000. p. 254–262. Citado na página 81.

CHAQFEH, M.; MOHAMED, N. Challenges in Middleware Solutions for the Internet of Things. In: IEEE. *Proceedings of the 2012 International Conference on Collaboration Technologies and Systems, CTS 2012*. Denver, CO, USA, 2012. p. 21–26. Citado 2 vezes nas páginas 36 e 60.

CITY-HUB. 2018. Disponível em: <https://cityhub.com/>. Citado na página 60.

COCCHIA, A. Smart and Digital City: A Systematic Literature Review. In: *Smart city*. [S.l.]: Springer, 2014. p. 13–43. Citado na página 33.

CONZON, D. et al. Industrial Application Development Exploiting IoT Vision and Model Driven Programming. In: *18th International Conference on Intelligence in Next Generation Networks (ICIN)*. Paris, France: Association for Computing Machinery, 2015. p. 168–175. Citado na página 49.

COULOURIS, G. et al. *Distributed Systems: Concepts and Design*. 5. ed. USA: Addison-Wesley Publishing Company, 2011. Citado na página 36.

CURRY, E. System of Systems Information Interoperability Using a Linked Dataspace. In: IEEE. *2012 7th International Conference on System of Systems Engineering (SoSE)*. [S.l.], 2012. p. 101–106. Citado na página 29.

DELICATO, F. C.; PIRES, P. F.; BATISTA, T. *Middleware Solutions for the Internet of Things*. Berlin, Germany: Springer, 2013. 1–86 p. Citado 2 vezes nas páginas 28 e 44.

DELICATO, F. C. et al. Towards an IoT Ecosystem. In: ACM. *Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems, SESoS@ECOOP*. Montpellier, France, 2013. p. 25–28. Citado 2 vezes nas páginas 49 e 60.

DESAI, P.; SHETH, A. P.; ANANTHARAM, P. Semantic Gateway as a Service Architecture for IoT Interoperability. In: *2015 IEEE International Conference on Mobile Services (MS)*. [S.l.]: IEEE Computer Society, 2015. p. 313–319. ISSN 2329-6429. Citado na página 28.

DEY, A. K. Understanding and Using Context. *Personal and ubiquitous computing*, Springer-Verlag, v. 5, n. 1, p. 4–7, 2001. Citado na página 42.

DIJKSTRA, E. W. The Structure of "THE-Multiprogramming System. *Commun. ACM*, v. 11, n. 5, p. 341–346, 1968. Citado na página 45.

DIJKSTRA, E. W. The Structure of "THE-Multiprogramming System (Reprint). *Commun. ACM*, v. 26, n. 1, p. 49–52, 1983. Citado na página 45.

DOUZIS, K. et al. Modular and generic IoT management on the cloud. *Future Generation Computer Systems*, Elsevier B.V., p. 369–378, 2016. Citado na página 37.

ELICEGUI, I. et al. Design and Implementation of a Cloud-Based Platform for Unleashing the Personal and Communal Internet of Things. *Mobile Information Systems*, v. 2017, p. 2164072:1–2164072:14, 2017. Citado na página 37.

ELMANGOUSH, A. et al. An Approach to Expose M2M Services Over OMA Next Generation Service Interface. In: IEEE. *2013 17th International Conference on Intelligence in Next Generation Networks (ICIN)*. Venice, Italy, 2013. p. 147–154. Citado 2 vezes nas páginas 39 e 64.

ERGAZAKIS, K.; METAXIOTIS, K. S.; PSARRAS, J. E. Towards Knowledge Cities: Conceptual Analysis and Success Stories. *J. Knowledge Management*, v. 8, n. 5, p. 5–15, 2004. Citado 2 vezes nas páginas 34 e 35.

FARAHZADI, A. et al. Middleware Technologies for Cloud of Things- A Survey. *Digital Communications and Networks*, Elsevier, abs/1705.00387, 2017. Citado 3 vezes nas páginas 43, 44 e 60.

FAZIO, M. et al. Exploiting the Fiware Cloud Platform to Develop a Remote Patient Monitoring System. In: IEEE COMPUTER SOCIETY. *2015 IEEE Symposium on Computers and Communication, ISCC 2015*. Larnaca, Cyprus, 2015. p. 264–270. Citado 2 vezes nas páginas 21 e 60.

FERREIRA, C.; MEIRELLES, P.; NERI, K. Study of Barriers to Contribute with Brazilian Public Software Projects. In: IEEE. *12th Iberian Conference on Information Systems and Technologies (CISTI 2017)*. Lisbon, Portugal, 2017. p. 1–6. Citado na página 33.

FERREIRA, D. et al. Towards Smart Agriculture Using FIWARE Enablers. In: *International Conference on Engineering, Technology and Innovation (ICE/ITMC 2017)*. Stuttgart, Germany: IEEE, 2017. p. 1544–1551. Citado na página 37.

FERSI, G. Middleware for Internet of Things: A Study. In: *2015 International Conference on Distributed Computing in Sensor Systems, DCOSS 2015, Fortaleza, Brazil, June 10-12, 2015*. [S.l.]: IEEE Computer Society, 2015. p. 230–235. Citado na página 37.

FIWARE. 2018. Disponível em: <https://www.kaaproject.org/>. Citado 5 vezes nas páginas 38, 39, 40, 41 e 64.

FONTELLES, M. J. et al. Metodologia da Pesquisa Científica: Diretrizes para a Elaboração de um Protocolo de Pesquisa. *Revista Paraense de Medicina*, v. 23, n. 3, p. 1–8, 2009. Citado na página 23.

FRATANTONIO, Y. et al. Triggerscope: Towards detecting logic bombs in android applications. In: . [S.l.: s.n.], 2016. p. 377–396. Citado na página 81.

GAMA, K.; TOUSEAU, L.; DONSEZ, D. Combining Heterogeneous Service Technologies for Building an Internet of Things Middleware. *Computer Communications*, Elsevier B.V., v. 35, n. 4, p. 405–417, 2012. Citado 2 vezes nas páginas 34 e 43.

GARCIA, T. M.; FORTES, D. X.; NASCIMENTO, R. P. C. do. Mapeamento Sistemático: Adoção de Governança de TIC em Cidades Inteligentes na Administração Pública. Revista Científica da Faculdade Sete de Setembro (FASETE). Paulo Afonso - Bahia, n. 17, p. 219–235, 2018. Citado na página 33.

GEHANI, N.; JAGADISH, H.; SHMUELI, O. Event specification in an active object-oriented database. *ACM SIGMOD Record*, v. 21, n. 2, p. 81–90, 1992. Citado na página 81.

GEORGAKOPOULOS, D.; JAYARAMAN, P. P. Internet of Things:From Internet Scale Sensing to Smart Services. *Computing*, Springer, v. 98, n. 10, p. 1041–1058, 2016. Citado na página 53.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de Pesquisa*. 1. ed. Porto Alegre: Editora da Universidade Federal do Rio Grande do Sul, 2009. Citado na página 23.

GIFFINGER, R. et al. City-Ranking of European Medium-Sized Cities. *Cent. Reg. Sci. Vienna UT*, p. 1–12, 2007. Citado na página 34.

GIL, A. C. Como Elaborar Projetos de Pesquisa. *São Paulo*, v. 6, n. 61, p. 16–17, 2017. Citado 3 vezes nas páginas 22, 23 e 83.

GOMES, M. M.; RIGHI, R. da R.; COSTA, C. A. da. Internet of Things Scalability: Analyzing the Bottlenecks and Proposing Alternatives. In: IEEE. *6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, ICUMT 2014*. Petersburg, Russia, 2014. p. 269–276. Citado na página 43.

GÓMEZ-GOIRI, A.; IPIÑA, D. López-de. A Triple Space-Based Semantic Distributed Middleware for Internet of Things. In: SPRINGER. *International Conference on Web Engineering*. Vienna, Austria, 2010. p. 447–458. Citado na página 43.

GUTH, J. et al. Comparison of IoT Platform Architectures: A Field Study based on a Reference Architecture. p. 1–6, 2016. Citado na página 59.

GWEBU, K. L.; WANG, J. Adoption of Open Source Software: The Role of Social Identification. *Decision Support Systems*, v. 51, n. 1, p. 220–229, 2011. Citado na página 32.

HAN, S. N. et al. Extending the Devices Profile for Web Services Standard Using a REST Proxy. *IEEE Internet Computing*, v. 19, n. 1, p. 10–17, 2015. Citado na página 55.

HEDDEBAUT, O.; CIOMMO, F. D. City-hubs for Smarter Cities. The Case of Lille "EuraFlandres" Interchange. *European Transport Research Review*, v. 10, n. 1, 2018. Citado na página 48.

HEIMGAERTNER, F. et al. Scaling Home Automation to Public Buildings: A Distributed Multiuser Setup for OpenHAB 2. In: *Global Internet of Things Summit, GIoTS 2017, June 6-9, 2017*. Geneva, Switzerland: Institute of Electrical and Electronics Engineers Inc., 2017. p. 1–6. Citado 2 vezes nas páginas 52 e 61.

HERTEL, G.; NIEDNER, S.; HERRMANN, S. Motivation of Software Developers in Open Source Projects: an Internet-Based Survey of Contributors to the Linux Kernel. *Research policy*, Elsevier, v. 32, n. 7, p. 1159–1177, 2003. Citado na página 32.

HUACARPUMA, R. C. et al. Distributed Data Service for Data Management in Internet of Things Middleware. *Sensors*, v. 17, n. 5, p. 977, 2017. Citado na página 51.

ISO. *Systems and Software Engineering–Architecture Description*. [S.l.], 2011. Citado na página 45.

ISO/IEC. *ISO/ IEC 25010 - Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models*, 2011. Citado 4 vezes nas páginas 84, 85, 86 e 87.

JAHN, M. et al. The Energy Aware Smart Home. *Proceedings of the Fifth International Conference on Future Information Technology (FutureTech)*, p. 1–8, 2010. Citado na página 56.

JAMMES, F.; SMIT, H. Service-oriented Paradigms in Industrial Automation. *IEEE Transactions on Industrial Informatics*, v. 1, n. 1, p. 62–70, 2005. Citado na página 55.

JANSSEN, M.; ESTEVEZ, E.; JANOWSKI, T. Interoperability in Big, Open, and Linked Data-Organizational Maturity, Capabilities, and Data Portfolios. *IEEE Computer*, v. 47, n. 10, p. 44–49, 2014. Citado na página 29.

JI, Z. et al. A Cloud-Based Car Parking Middleware for IoT-Based Smart Cities: Design and Implementation. *Sensors*, v. 14, n. 12, p. 22372–22393, 2014. Citado na página 52.

JR., J. F. N.; CHEN, M.; PURDIN, T. D. M. Systems development in information systems research. *Journal of management information systems*, Taylor & Francis, v. 7, n. 3, p. 89–106, 1991. Citado na página 83.

KAA. 2018. Disponível em: <https://www.kaaproject.org/>. Citado 2 vezes nas páginas 51 e 60.

KALDELI, E. et al. Coordinating the Web of Services for a Smart Home. *ACM Transactions on the Web (TWEB)*, v. 7, n. 2, p. 10:1–10:40, 2013. Citado na página 55.

KALDELI, E. et al. Coordinating the Web of Services for a Smart Home. *TWEB*, v. 7, n. 2, p. 10:1–10:40, 2013. Citado na página 61.

KASSAL, P.; STEINBERG, M.; STEINBERG, I. Wireless Chemical Sensors and Biosensors: A Review. *Sensors and Actuators, B: Chemical*, v. 266, p. 228–245, 2018. Citado na página 43.

KATASONOV, A. et al. Smart Semantic Middleware for the Internet of Things. *Proceedings of the Fifth International Conference on Informatics in Control, Automation and Robotics Service*, v. 8, p. 169–178, 2008. Citado na página 58.

KERSTING, J. et al. Internet of Things Architecture for Handling Stream Air Pollution Data. In: INSTICC. *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security - Volume 1: IoTBDS,*. Porto, Portugal: SciTePress, 2017. p. 117–124. Citado na página 51.

KHALEEL, H. et al. Heterogeneous Applications, Tools, and Methodologies in the Car Manufacturing Industry Through an IoT Approach. *IEEE Systems Journal*, v. 11, n. 3, p. 1412–1423, 2017. Citado na página 49.

KIM, E. Smart City Service Platform Sssociated with Smart Home. In: . [S.l.: s.n.], 2017. p. 608–610. Citado na página 34.

KIM-HUNG, L. et al. An Industrial IoT Framework to Simplify Connection Process Using System-Generated Connector. In: *IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI 2017)*. Modena, Italy: IEEE, 2017. p. 1–6. Citado na página 51.

KIM, J.; LEE, J. OpenIoT: An Open Service Framework for the Internet of Things. In: IEEE COMPUTER SOCIETY. *IEEE World Forum on Internet of Things, WF-IoT 2014, March 6-8, 2014*. Seoul, South Korea, 2014. p. 89–93. Citado na página 53.

KOMNINOS, N.; TSARCHOPOULOS, P.; KAKDERI, C. New Services Design for Smart Cities: A Planning Roadmap for User-Driven innovation. In: . Philadelphia, Pennsylvania, USA: Proceedings of the 2014 ACM international workshop on Wireless and mobile technologies for smart cities, WiMobCity@MobiHoc 2014, August 11, 2014, 2014. p. 29–38. Citado na página 34.

KOSTELNIK, P.; SARNOVSK, M.; FURDIK, K. The Semantic Middleware for Networked Embedded Systems Applied in the Internet of Things and Services Domain. *Scalable Computing: Practice and Experience*, v. 12, n. 3, p. 307–316, 2011. Citado 2 vezes nas páginas 48 e 51.

KOURTIT, K.; NIJKAMP, P. Smart cities in the innovation age. *Innovation: The European Journal of Social Science Research*, Taylor & Francis, v. 25, n. 2, p. 93–95, 2012. Citado na página 34.

KOVACS, E. et al. Standards-Based Worldwide Semantic Interoperability for IoT. *IEEE Communications Magazine*, IEEE, v. 54, n. 12, p. 40–46, 2016. Citado 4 vezes nas páginas 17, 38, 40 e 60.

KRCO, S.; POKRIC, B.; CARREZ, F. Designing IoT Architecture(s): A European Perspective. In: *IEEE World Forum on Internet of Things (WF-IoT 2014)*. Seoul, South Korea: IEEE Computer Society, 2014. p. 79–84. Citado 3 vezes nas páginas 17, 19 e 42.

KRUCHTEN, P.; OBBINK, J. H.; STAFFORD, J. A. The Past, Present, and Future for Software Architecture. *IEEE Software*, n. 2, p. 22–30, 2006. Citado na página 45.

KRYLOVSKIY, A.; JAHN, M.; PATTI, E. Designing a Smart City Internet of Things Platform with Microservice Architecture. In: IEEE. *3rd International Conference on Future Internet of Things and Cloud, FiCloud 2015, August 24-26, 2015*. Rome, Italy, 2015. p. 25–30. Citado na página 61.

KUIKKANIEMI, K. et al. From Space to Stage: How Interactive Screens will Change Urban Life. *Computer*, IEEE, v. 44, n. 6, p. 40–47, 2011. Citado na página 34.

KYUSAKOV, R. et al. Integration of Wireless Sensor and Actuator Nodes With IT Infrastructure Using Service-Oriented Architecture. *IEEE Trans. Industrial Informatics*, v. 9, n. 1, p. 43–51, 2013. Citado na página 61.

LEA, R.; BLACKSTOCK, M. City-hub: A Cloud-Based IoT Platform for Smart Cities. p. 799–804, 2014. Citado 3 vezes nas páginas 39, 48 e 60.

LI, K. F. Smart Home Technology for Telemedicine and Emergency Management. *J. Ambient Intelligence and Humanized Computing*, v. 4, n. 5, p. 535–546, 2013. Citado na página 61.

LIKERT, R. A Technique for the Measurement of Attitudes. *Archives of psychology*, 1932. Citado 3 vezes nas páginas 84, 85 e 86.

LINKSMART. 2018. Disponível em: <https://www.linksmart.eu/>. Citado na página 51.

LIU, M. et al. Semantic Agent-Based Service Middleware and Simulation for Smart Cities. *Sensors (Switzerland)*, v. 16, n. 12, p. 1–25, 2016. ISSN 14248220. Citado 2 vezes nas páginas 17 e 35.

MACHADO, K. C. de B.; SANTOS, E. M. dos; JUNIOR, A. E. de A. Adoção de Arquiteturas de Interoperabilidade para Governo Eletrônico: Estudo de Casos Múltiplos no Brasil. In: *International Conference on Information Resources Management (CONF-IRM)*. [S.l.: s.n.], 2013. p. 4. Citado 2 vezes nas páginas 19 e 21.

MADNI, A. M.; SIEVERS, M. Systems Integration: Key Perspectives, Experiences, and Challenges. *Systems Engineering*, v. 17, n. 1, p. 37–51, 2014. Citado na página 29.

MADUKWE, K.; EZIKA, I.; ILOANUSI, O. Leveraging Edge Analysis for Internet of Things Based Healthcare Solutions. In: IEEE. Owerri,Nigeria, 2018. v. 2018-January, p. 720–725. Citado na página 51.

MALCHE, T.; MAHESHWARY, P. Harnessing the Internet of Things (iot): A Review. *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)*, v. 5, n. 8, p. 320–323, 2015. Citado na página 51.

MALHOTRA, N. K. *Pesquisa de Marketing: uma Orientação Aplicada*. 3. ed. [S.l.]: Bookman Editora, 2012. Citado na página 23.

MARQUES, G.; GARCIA, N.; POMBO, N. A Survey on IoT: Architectures, Elements, Applications, QoS, Platforms and Security Concepts. In: *Advances in Mobile Cloud Computing and Big Data in the 5G Era*. [S.l.]: Springer, 2017. p. 115–130. Citado na página 43.

MARSAN, J.; PARÉ, G.; BEAUDRY, A. Adoption of Open Source Software in Organizations: A Socio-Cognitive perspective. *J. Strategic Inf. Sys.*, v. 21, n. 4, p. 257–273, 2012. Citado na página 32.

MARTINEZ, A. et al. Generic Module for Collecting Data in Smart Cities. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Copernicus GmbH, v. 42, p. 65–72, 2017. Citado 2 vezes nas páginas 17 e 39.

MATTERN, F.; FLOERKEMEIER, C. From the Internet of Computers to the Internet of Things. In: *From active data management to event-based systems and more - Papers in Honor of Alejandro Buchmann on the Occasion of His 60th Birthday*. [S.l.]: Springer, 2010. p. 242–259. Citado na página 43.

MEIRELLES, P. et al. Brazilian Public Software Portal: An Integrated Platform for Collaborative Development. In: *OpenSym '17 Proceedings of the 13th International Symposium on Open Collaboration*. Galway, Ireland: ACM, 2017. p. 1–10. Citado na página 33.

MELLO, A. P. P.; MESQUITA, H.; VIEIRA, C. E. Introdução à Interoperabilidade (eping). Escola Nacional de Administração Pública (Enap), p. 1–15, 2015. Citado 3 vezes nas páginas 16, 19 e 28.

MERRIAM, S. B. *Qualitative Research and Case Study Applications in Education. Revised and Expanded from"Case Study Research in Education"*. [S.l.]: ERIC, 1998. Citado na página 23.

MICLAUS, A.; RIEDEL, T.; BEIGL, M. End-user Installation of Heterogeneous Home Automation Systems Using Pen and Paper Interfaces and Dynamically Generated Documentation. In: *4th International Conference on the Internet of Things, IOT 2014, October 6-8, 2014*. Cambridge, MA, USA: IEEE, 2014. p. 19–24. Citado na página 53.

MINERAUD, J. et al. A Gap Analysis of Internet-of-Things Platforms. *Computer Communications*, Elsevier B.V., v. 89-90, p. 5–16, 2016. Citado 2 vezes nas páginas 47 e 60.

MINERAUD, J. et al. Middleware Technologies for Cloud of Things - A Survey. *Digital Communications and Networks*, Elsevier B.V., v. 89-90, n. January, p. 1–13, 2017. Citado na página 47.

MIORANDI, D. et al. Internet of Things: Vision, Applications and Research Challenges. *Ad Hoc Networks*, v. 10, n. 7, p. 1497–1516, 2012. Citado 2 vezes nas páginas 36 e 37.

MOHAMED, N.; AL-JAROODI, J. A Survey on Service-Oriented Middleware for Wireless Sensor Networks. *Service Oriented Computing and Applications*, v. 5, n. 2, p. 71–85, 2011. Citado na página 62.

MOHAMMADI, M. et al. Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services. *IEEE Internet of Things Journal*, v. 5, n. 2, p. 624–635, 2018. Citado na página 43.

MOHAMMED, B.; KIRAN, M. Analysis of Cloud Test Beds Using OpenSource Solutions. In: IEEE COMPUTER SOCIETY. *3rd International Conference on Future Internet of Things and Cloud, FiCloud 2015, August 24-26, 2015*. Rome, Italy, 2015. p. 195–203. Citado na página 53.

MOLTCHANOV, B.; ROCHA, O. R. A Context Broker to Enable Future IoT Applications and Services. In: *6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, ICUMT 2014, October 6-8, 2014*. Petersburg, Russia: IEEE, 2014. p. 263–268. Citado 2 vezes nas páginas 37 e 38.

MONDORF, A.; WIMMER, M. Contextual Components of an Enterprise Architecture Framework for Pan-European eGovernment Services. In: *50th Hawaii International Conference on System Sciences, HICSS 2017, Hilton Waikoloa Village, Hawaii, USA, January 4-7, 2017*. [S.l.: s.n.], 2017. p. 2933–2942. Citado na página 16.

MOSTASHARI, A. et al. Citizens as Sensors: The Cognitive City Paradigm. In: IEEE. *8th International Conference & Expo on Emerging Technologies for a Smarter World (CEWIT), 2011*. NY, USA, 2011. p. 1–5. Citado na página 34.

MYNY, K.; STEUDEL, S. Flexible Thin-Film NFC Transponder Chip Exhibiting Data Rates Compatible to ISO NFC Standards Using Self-Aligned Metal-Oxide TFTs. In: *2016 IEEE International Solid-State Circuits Conference, ISSCC 2016, January 31 - February 4, 2016*. San Francisco, CA, USA: IEEE, 2016. p. 298–299. Citado na página 42.

NAGY, M. et al. Challenges of Middleware for the Internet of Things. In: *Automation Control-Theory and Practice*. [S.l.]: InTech, 2009. Citado na página 58.

NAMIOT, D.; SNEPS-SNEPPE, M. On Software Standards for Smart Cities: API or DPI. *Proceedings of the 2014 ITU Kaleidoscope Academic Conference: Living in a Converged World - Impossible Without Standards?, K 2014*, p. 169–174, 2014. Citado na página 38.

NIKITIN, S.; KATASONOV, A.; TERZIYAN, V. Y. Ontonuts: Reusable Semantic Components for Multi-agent Systems. In: IEEE COMPUTER SOCIETY. *Fifth International Conference on Autonomic and Autonomous Systems, ICAS 2009, 20-25 April 2009*. Valencia, Spain, 2009. p. 200–207. Citado na página 58.

NITTI, M. et al. The Virtual Object as a Major Element of the Internet of Things: A Survey. *IEEE Communications Surveys and Tutorials*, v. 18, n. 2, p. 1228–1240, 2016. Citado na página 43.

NOTEPADD++. *Notepad++ v7.6.5 - Current Version*. 2019. Disponível em: <https://notepad-plus-plus.org/download/v7.6.5.html>. Citado na página 86.

OPENHAB. 2018. Disponível em: <https://www.openhab.org/>. Citado 3 vezes nas páginas 52, 53 e 61.

OPENIOT. 2018. Disponível em: <http://www.openiot.eu/>. Citado na página 61.

PALADE, A. et al. Middleware for Internet of Things: A Quantitative Evaluation in Small Scale. *18th IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks, WoWMoM 2017, Macau, China, June 12-15, 2017*, p. 1–6, 2017. Citado 7 vezes nas páginas 43, 51, 52, 58, 59, 61 e 62.

PAN, J.; PAUL, S.; JAIN, R. A Survey of the Research on Future Internet Architectures. *Communications Magazine, IEEE , v.49, n.7*, n. July, p. 26–36, 2011. Citado na página 37.

PARDO, T. A.; NAM, T.; BURKE, G. B. E-government Interoperability: Interaction of Policy, Management, and Technology Dimensions. *Social Science Computer Review*, Sage Publications Sage CA: Los Angeles, CA, v. 30, n. 1, p. 7–23, 2012. Citado na página 16.

PARNAS, D. L. A Technique for Software Module Specification with Examples. *Commun. ACM*, v. 15, n. 5, p. 330–336, 1972. Citado na página 45.

PARNAS, D. L. On the Criteria to be Used in Decomposing Systems into Modules. *Commun. ACM*, Springer Berlin Heidelberg, v. 15, n. 12, p. 1053–1058, 1972. Citado na página 45.

PARNAS, D. L. A Technique for Software Module Specification with Examples (Reprint). *Commun. ACM*, v. 26, n. 1, p. 75–78, 1983. Citado na página 45.

PARNAS, D. L. On the Criteria to be Used in Decomposing Systems into Modules (Reprint). p. 411–427, 2002. Citado na página 45.

PATTI, E. et al. Distributed software infrastructure for general purpose services in smart grid. *IEEE Transactions on Smart Grid*, IEEE, v. 7, n. 2, p. 1156–1163, 2016. Citado na página 52.

PAUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F. Restful Web Services vs. Big'web Services: Making the Right Architectural Decision. In: ACM. *Proceedings of the 17th international conference on World Wide Web*. Beijing, China, 2008. p. 805–814. Citado na página 49.

PERERA, C. et al. Dynamic Configuration of Sensors Using Mobile Sensor Hub in Internet of Things Paradigm. In: IEEE. *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, April 2-5, 2013*. Melbourne, Australia, 2013. p. 473–478. Citado 2 vezes nas páginas 50 e 60.

PERERA, C. et al. Context-aware Dynamic Discovery and Configuration of 'Things' in Smart Environments. *CoRR*, p. 215–241, 2014. Citado na página 50.

PERERA, C. et al. Capturing Sensor Data from Mobile Phones Using Global Sensor Network Middleware. In: *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*. Sydney, NSW, Australia: IEEE, 2012. p. 24–29. Citado na página 59.

PERERA, C. et al. Context Aware Computing for the Internet of Things : A Survey. *IEEE Communications Surveys & Tutorial*, v. 16, n. 1, p. 1–41, 2015. Citado 4 vezes nas páginas 46, 60, 61 e 62.

PERERA, C. et al. Semantic-Driven Configuration of Internet of Things Middleware. In: *2013 Ninth International Conference on Semantics, Knowledge and Grids*. Beijing, China: IEEE, 2013. p. 66–73. Citado na página 50.

PERRY, D. E.; WOLF, A. L. Foundations for the Study of Software Architecture. *SIGSOFT Softw. Eng. Notes*, ACM, New York, NY, USA, v. 17, n. 4, p. 40–52, 10 1992. Citado na página 45.

PETERSEN, K. Measuring and Predicting Software Productivity: A Systematic Map and Review. *Information & Software Technology*, v. 53, n. 4, p. 317–343, 2011. Citado na página 26.

PETERSEN, K. et al. Systematic Mapping Studies in Software Engineering. In: *12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008, 26-27 June 2008*. University of Bari, Italy: BCS, 2008. (Workshops in Computing). Citado na página 25.

PFLANZNER, T.; KERTÉSZ, A. A Survey of IoT Cloud Providers. In: IEEE. *39th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2016, May 30 - June 3, 2016*. Opatija, Croatia, 2016. v. 1, p. 730–735. Citado 4 vezes nas páginas 17, 19, 42 e 51.

PINTO, J. P. V. d. C. *Interoperability in Software Applications for Smart Cities: Towards a Reference Architecture*. 1–98 p. Dissertação (Mestrado) — Faculdade de Engenharia da Universidade do Porto (FEUP), 7 2016. Citado na página 21.

PIRES, P. F. et al. A Platform for Integrating Physical Devices in the Internet of Things. In: *Embedded and Ubiquitous Computing (EUC 2014), 12th IEEE International Conference on*. Milano, Italy: IEEE Computer Society, 2014. p. 234–241. Citado na página 49.

PIRES, P. F. et al. Plataformas para a Internet das Coisas. *Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2015)*, p. 110–169, 2015. Citado 6 vezes nas páginas 46, 48, 49, 50, 53 e 58.

PREVENTIS, A. et al. Iot-a and FIWARE: Bridging the Barriers between the Cloud and Iot Systems Design and Implementation. In: *CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science*. Rome, Italy: SciTePress, 2016. v. 2, p. 146–153. Citado 3 vezes nas páginas 21, 37 e 60.

QIN, W. et al. RestThing: A Restful Web Service Infrastructure for Mash-up Physical and Web Resources. In: IEEE. *2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing (EUC)*. Melbourne, VIC, Australia, 2011. p. 197–204. Citado na página 54.

RABBAH, M. et al. Challenges Facing Middleware for Mobile Robots in Smart Environment. *International Journal of Scientific and Engineering Research*, v. 7, p. 33–40, 2016. Citado na página 43.

RAICU, I.; FOSTER, I. T.; ZHAO, Y. Many-Task Computing for Grids and Supercomputers. In: IEEE. *2008 Workshop on Many-Task Computing on Grids and Supercomputers, MTAGS@SC 2008, November 17, 2008*. Austin, TX, USA, 2008. p. 1–11. Citado 2 vezes nas páginas 57 e 61.

RAMLJAK, M. Security Analysis of Open Home Automation Bus System. In: *40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2017, Opatija, Croatia, May 22-26, 2017*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2017. p. 1245–1250. Citado 2 vezes nas páginas 52 e 61.

RAZZAQUE, M. A. et al. Middleware for Internet of Things: a Survey. *IEEE Internet of Things Journal*, IEEE, v. 3, n. 1, p. 70–95, 2016. Citado 6 vezes nas páginas 17, 19, 35, 37, 42 e 52.

REAIDY, P. J.; GUNASEKARAN, A.; SPALANZANI, A. Bottom-up Approach based on Internet of Things for Order Fulfillment in a Collaborative Warehousing Environmen. *International Journal of Production Economics*, Elsevier, v. 159, p. 29–40, 2015. Citado na página 48.

RONEN, E. et al. Iot Goes Nuclear: Creating a Zigbee Chain Reaction. *IEEE Security & Privacy*, v. 16, n. 1, p. 54–62, 2018. Citado na página 42.

SADOOGHI, I. et al. Achieving Efficient Distributed Scheduling with Message Queues in the Cloud for Many-Task Computing and High-Performance Computing. In: IEEE. *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2014, May 26-29, 2014*. Chicago, IL, USA, 2014. p. 404–413. Citado na página 61.

SAHOO, J.; RATH, M. Study and Analysis of Smart Applications in Smart City Context. In: IEEE. Singapore, Singapore, 2018. p. 225–228. Citado na página 34.

SALHOFER, P. Evaluating the FIWARE Platform A Case-Study on Implementing Smart Application with FIWARE. In: *51st Hawaii International Conference on System Sciences, HICSS 2018, January 3-6, 2018*. Hawaii, USA: HICSS, 2018. p. 1–9. Citado na página 41.

SALIHBEGOVIC, A. et al. Design of a Domain Specific Language and IDE for Internet of Things Applications. In: IEEE. *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*. [S.l.], 2015. p. 996–1001. Citado na página 52.

SANIAT, M. R. et al. Autonomic Frameworks Deployment Using Configuration and Service Delivery Models for the Internet of Things. In: *Interoperability and Open-Source Solutions for the Internet of Things - International Workshop, FP7 OpenIoT Project, Held in Conjunction with SoftCOM 2014, September 18, 2014. Invited Papers*. Split, Croatia: Springer, 2015. v. 9001, p. 89–102. Citado na página 61.

SANTANA, E. M. G. de; ALENCAR, R. L. S. de; CORREIA, C. A. de C. An Analysis of Performance Anti-Patterns in Systems Stored on the Brazilian Public Software Portal. *IEEE Latin America Transactions*, IEEE, v. 15, n. 4, p. 705–710, 2017. Citado 2 vezes nas páginas 32 e 33.

SANTANA, J. R. et al. On the Use of Information and Infrastructure Technologies for the Smart City Research in Europe: A Survey. *IEICE Transactions on Communications*, v. 101-B, n. 1, p. 2–15, 2018. ISSN 17451345. Citado na página 60.

SANTOS, D. F. S.; ALMEIDA, H. O.; PERKUSICH, A. A Personal Connected Health System for the Internet of Things Based on the Constrained Application Protocol. *Computers & Electrical Engineering*, v. 44, p. 122–136, 2015. Citado na página 55.

SANTOS, E.; REINHARD, N. Barriers to Government Interoperability Frameworks Adoption. In: *16th Americas Conference on Information Systems 2010, AMCIS 2010*. [S.l.: s.n.], 2010. v. 2, p. 472. Citado na página 44.

SANTOS, E. M. d. *Desenvolvimento e Implementação de Padrões de Interoperabilidade em Governo Eletrônico no Brasil*. 1–184 p. Tese (Doutorado) — Universidade de São Paulo, 3 2008. Citado 2 vezes nas páginas 20 e 21.

SANTOS, E. M. d.; REINHARD, N. Electronic Government Interoperability: Identifying the Barriers for Frameworks Adoption. *Social Science Computer Review*, Sage Publications Sage CA: Los Angeles, CA, v. 30, n. 1, p. 71–82, 2012. Citado 3 vezes nas páginas 20, 21 e 29.

SANTOS, E. M. dos; REINHARD, N. Barriers to Government Interoperability Frameworks Adoption. In: *16th Americas Conference on Information Systems (AMCIS 2010)*. Lima, Peru: Association for Information Systems, 2010. v. 2, p. 1–10. Citado 3 vezes nas páginas 16, 17 e 28.

SANTOS, S. E. G. S. dos; ROCHA, T. da; CARVALHO, F. O. Highframe: An Integrated Solution for Developing Component-Based Distributed Systems. In: *Proceedings of the 7th Euro American Conference on Telematics and Information Systems (EATIS 2014)*. Valparaíso, Chile: ACM, 2014. p. 1–6. Citado na página 28.

SCHAFFERS, H. et al. Smart Cities and the Future Internet: Towards Cooperation Frameworks for Open Innovation. In: SPRINGER. *The Future Internet - Future Internet Assembly 2011 (FIA): Achievements and Technological Promises*. Berlin, Heidelberg, 2011. v. 6656, p. 431–446. Citado 2 vezes nas páginas 18 e 41.

SCHLEICHER, J. M. et al. Enabling a Smart City Application Ecosystem: Requirements and Architectural Aspects. *IEEE Internet Computing*, IEEE, v. 20, n. 2, p. 58–65, 2016. Citado na página 17.

SCHOLL, H. J.; KLISCHEWSKI, R. E-government Integration and Interoperability: Framing the Research Agenda. *International Journal of Public Administration*, Taylor & Francis, v. 30, n. 8-9, p. 889–920, 2007. Citado na página 29.

SCOPUS. 2018. Disponível em: <https://www.elsevier.com/solutions/scopus>. Citado na página 25.

SCUTURICI, V.-M. et al. UbiWare: Web-Based Dynamic Data & Service Management Platform for AmI. In: *Proceedings of the Posters and Demo Track, Middleware 2012*. [S.l.: s.n.], 2012. p. 1–3. Citado na página 62.

SEIGER, R.; HUBER, S.; SCHLEGEL, T. Toward an Execution System for Self-Healing Workflows in Cyber-Physical Systems. *Software and Systems Modeling*, Springer, v. 17, n. 2, p. 1–22, 2018. Citado na página 52.

SERRANO, M. et al. Defining the Stack for Service Delivery Models and Interoperability in the Internet of Things: A Practical Case With OpenIoT-VDK. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 33, n. 4, p. 676–689, 2015. Citado 2 vezes nas páginas 28 e 53.

SHAPIRO, J. M. Smart Cities: Quality of Life, Productivity, and the Growth Effects of Human Capital. *The Review of Economics and Statistics*, v. 88, n. May, p. 324–335, 2006. Citado na página 34.

SHELBY, Z. Embedded Web Services. *IEEE Wireless Commun.*, v. 17, n. 6, p. 52–57, 2010. Citado na página 47.

SILVEIRA, D. T.; CÓRDOVA, F. P. A Pesquisa Científica. *Métodos de pesquisa. Porto Alegre: Universidade Federal do Rio Grande do Sul*, p. 31–42, 2009. Citado na página 23.

SILVEIRA, L.; WAZLAWICK, R. S.; ROVER, A. J. Assessing the Brazilian e-Justice Interoperability Model. *IEEE Latin America Transactions*, IEEE, v. 13, n. 5, p. 1504–1510, 2015. Citado 2 vezes nas páginas 16 e 21.

SINGH, K. J.; KAPOOR, D. S. Create Your Own Internet of Things: A Survey of IoT Platforms. *IEEE Consumer Electronics Magazine*, IEEE, v. 6, n. 2, p. 57–68, 2017. Citado na página 47.

SMIREK, L.; ZIMMERMANN, G.; ZIEGLER, D. Towards Universally usable smart homes-how can myui, urc and openhab contribute to an adaptive user interface platform. *IARIA, Nice, France*, CENTRIC 2014 : The Seventh International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, p. 29–38, 2014. Citado na página 52.

SOLDATOS, J.; SERRANO, M.; HAUSWIRTH, M. Convergence of Utility Computing with the Internet-of-Things. In: IEEE COMPUTER SOCIETY. *Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012)*. Sanpaolo Palace Hotel, Palermo, Italy, 2012. p. 874–879. Citado 2 vezes nas páginas 42 e 43.

SOUZA, L. M. S. de et al. SOCRADES: A Web Service Based Shop Floor Integration Infrastructure. In: *The Internet of Things, First International Conference, IOT 2008, March 26-28, 2008. Proceedings*. Zurich, Switzerland: Springer, 2008. p. 50–67. Citado na página 56.

STRAVOSKOUFOS, K. et al. Motion Sensor Driven Gesture Recognition for Future Internet Application Development. In: CITESEER. *5th International Conference on Information, Intelligence, Systems and Applications, IISA 2014,July 7-9, 2014*. Chania, Crete, Greece, 2014. p. 372–377. Citado na página 60.

SUN, X.; BLATECKY, A. R. Middleware: the Key to Next Generation Computing. *Journal of Parallel and Distributed Computing*, v. 64, n. 6, p. 689–691, 2004. Citado na página 35.

SUN, Y. et al. Internet of Things and Big Data Analytics for Smart and Connected Communities. *IEEE Access*, IEEE, v. 4, p. 766–773, 2016. Citado na página 40.

TANENBAUM, A. S.; STEEN, M. van. *Distributed Systems - Principles and Paradigms, 2nd Edition*. [S.l.]: Pearson Education, 2007. 1–999 p. Citado 2 vezes nas páginas 18 e 35.

TELES, N. P. J. *Êxodo Framework: Uma Solução para Modelagem do Comportamento de Ambientes para Internet das Coisas*. Tese (Doutorado) — Universidade Federal do Amazonas, 4 2018. Citado 10 vezes nas páginas 44, 46, 47, 49, 52, 57, 58, 59, 60 e 61.

TU'N, A. L. et al. Global Sensor Modeling and Constrained Application Methods Enabling Cloud-Based Open Space Smart Services. In: *2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*. [S.l.]: IEEE, 2012. p. 196–203. Citado na página 50.

VAJDA, V. et al. The EBBITS Project: An Interoperability Platform for a Real-world Populated Internet of Things Domain. *Proceedings of the International Conference Znalosti (Knowledge)*, p. 309–312, 2011. Citado 2 vezes nas páginas 48 e 60.

VALERO-LARA, P. et al. Many-Task Computing on Many-Core Architectures. *Scalable Computing: Practice and Experience*, v. 17, n. 1, p. 32–46, 2016. Citado na página 61.

VEGA-BARBAS, M. et al. Smart Spaces and Smart Objects Interoperability Architecture (S3OiA). In: IEEE. *Proceedings - 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012*. Sanpaolo Palace Hotel, Palermo, Italy, 2012. p. 725–730. Citado 2 vezes nas páginas 54 e 61.

VEGA-BARBAS, M. et al. Adaptive Software Architecture Based on Confident HCI for the Deployment of Sensitive Services in Smart Homes. *Sensors*, v. 15, n. 4, p. 7294–7322, 2015. Citado na página 61.

VENTURA, D. et al. ARIIMA: A Real IoT Implementation of a Machine-Learning Architecture for Reducing Energy Consumption. In: SPRINGER. *Ubiquitous Computing and Ambient Intelligence. Personalisation and User Adapted Services - 8th International Conference, UCAmI 2014, December 2-5, 2014. Proceedings*. Belfast, UK, 2014. p. 444–451. Citado na página 61.

VIJAYAKUMAR, P. et al. Computationally Efficient Privacy Preserving Anonymous Mutual and Batch Authentication Schemes for Vehicular ad hoc Networks. *Future Generation Comp. Syst.*, v. 78, p. 943–955, 2018. Citado na página 42.

VLIET, H. van. Software Architecture Knowledge Management. In: IEEE. *19th Australian Software Engineering Conference (ASWEC 2008), March 25-28, 2008*. Perth, Australia, 2008. p. 24–31. Citado na página 45.

WARRIACH, E. U. et al. Heterogeneous Device Discovery Framework for the Smart Homes. In: IEEE. *GCC Conference and Exhibition (GCC), 2011 IEEE*. Dubai, United Arab Emirates, 2011. p. 637–640. Citado na página 56.

WAZLAWICK, R. *Metodologia de Pesquisa para Ciência da Computação*. Brasil: Elsevier, 2015. v. 2. Citado 2 vezes nas páginas 22 e 23.

WEISS, M. C.; BERNARDES, R. C.; CONSONI, F. L. Cidades Inteligentes: Casos e Perspectivas para as Cidades Brasileiras. *Revista Tecnológica da Fatec Americana*, v. 5, n. 1, p. 01–13, 2017. Citado 2 vezes nas páginas 17 e 18.

WERNER, S.; PALLAS, F.; BERMBACH, D. Designing Suitable Access Control for Web-Connected Smart Home Platforms. In: *Service-Oriented Computing - ICSOC 2017 Workshops - ASOCA, ISyCC, WESOACS, and Satellite Events, November 13-16, 2017, Revised Selected Papers*. Málaga, Spain: Springer, 2018. (Lecture Notes in Computer Science, v. 10797), p. 240–251. Citado na página 61.

WINTERS, J. V. Why are Smart Cities Growing? Who Moves and Who Stays. *Journal of Regional Science*, v. 51, n. 2, p. 253–270, 2011. Citado na página 34.

WIRESHARK. *Wireshark · Go Deep.* 2019. Disponível em: <https://www.wireshark.org/>. Citado na página 87.

WOHLIN, C. et al. *Experimentation in software engineering: an introduction*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 98.

XAMPP. *XAMPP*. 2019. Disponível em: <https://www.apachefriends.org/pt_br/download.html>. Citado na página 67.

YIN, R. K. *Estudo de Caso: Planejamento e Métodos*. [S.l.]: Bookman editora, 2015. Citado na página 83.

ZAHARIADIS, T. B. et al. FIWARE Lab: Managing Resources and Services in a Cloud Federation Supporting Future Internet Applications. In: IEEE. *Utility and Cloud Computing (UCC 2014), IEEE/ACM 7th International Conference on*. [S.l.]: IEEE Computer Society, 2014. p. 792–799. Citado na página 41.

ZANELLA, A. et al. Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, v. 1, n. 1, p. 22–32, 2014. Citado na página 34.

ZDRAVKOVIĆ, M. et al. Survey of Internet-of-Things Platforms. In: *6th International Conference on Information Society and Techology, ICIST 2016*. Barcelona, Spain: ICIST, 2016. v. 1, p. 216–220. Citado na página 47.

ZHILIANG, W. et al. A SOA Based IOT Communication Middleware. In: IEEE. *International conference on Mechatronic Science, Electric Engineering and Computer (MEC) 2011*. United States, 2011. p. 2555–2558. Citado na página 43.

ZHONG, L. J.; ZHAO, J. Z.; DAI, W. H. System Mechanism of Small IOT System – Taking the Openhab Project as an Example. *Applied Mechanics and Materials*, Trans Tech Publications Ltd, v. 599-601, p. 1906–1909, 2014. Citado na página 52.

ZIMMERMANN, A.; LORENZ, A.; OPPERMANN, R. An Operational Definition of Context. In: *Modeling and Using Context, 6th International and Interdisciplinary Conference, CONTEXT 2007, August 20-24, 2007, Proceedings*. Roskilde, Denmark: Springer, 2007. (Lecture Notes in Computer Science, v. 4635), p. 558–571. Citado na página 42.

Apêndices

APÊNDICE A – Instalando e Configurando o *Framework Smart* e-PING

A.1 Introdução

Para realizar a configuração do *Framework Smart* e-PING, primeiro será necessário que o usuário tenha previamente instalados na máquina o PHP 5, o MySQL 5 e o Servidor Apache 2 ou equivalente. Caso o usuário não tenha os aplicativos, elencados acima, já instalados é necessário realizar o *download* do XAMPP. O XAMPP irá instalar automaticamente o PHP 5, o MySQL 5 e o Servidor Apache 2.

Os requisitos que se aplicam a instalação são:

Requisitos de *Hardware*	
Hardware	Requisitos
Processador mínimo Processador recomendado	Intel Core i3-5000U 2.00 GHz - 5ª Geração, 64 *bits* Intel Core i7-3770 3.40 GHz - 7ª Geração, 64 *bits*
Memória mínima Memória recomendada	4 Gb 8 Gb
Espaço em disco mínimo Espaço em disco recomendado	20 Mb 50 Mb
Outros *Hardwares*	Mouse e Teclado
Requisitos de *Software*	
Sistema Operacional	*Linux* (diversos), *Windows* (diversos) Todos que possibilitem a instalação do XAMPP
Banco de Dados	MySQL 5.5 (ou versão superior)
Servidor de Aplicação	Apache 2.2 PHP 5.3
Navegador *Web*	Firefox, Safari, Chrome, Opera, *Internet Explorer* Atualizados na última versão disponível

Fonte – Autora (2019).

A.2 Como Instalar

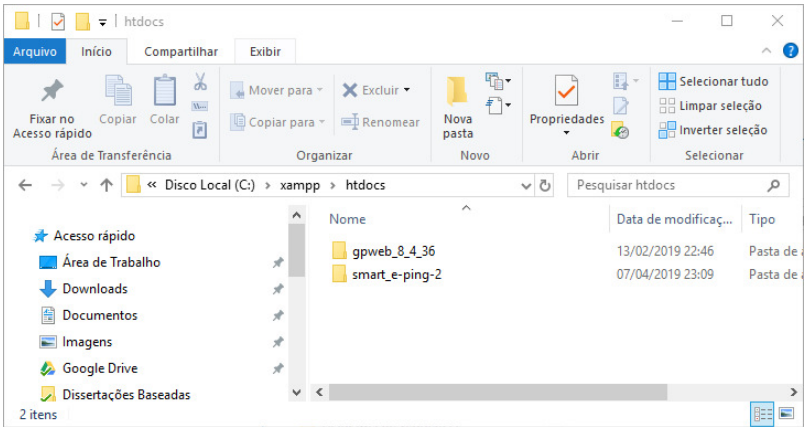
A seguir, é ilustrado como instalar o *Framework Smart* e-PING utilizando o servidor XAMPP no *Windows*. No entanto, os passos em outros servidores são semelhantes.

a) **Passo 1**: realize o *download* dos arquivos do *Framework Smart* e-PING, disponível no endereço eletrônico <https://github.com/ThauaneDcomp/e-smart>.

b) **Passo 2**: efetue o *download* do XAMPP, no endereço eletrônico <https://bit.ly/1JqFvzq>. Em seguida, instale o servidor XAMPP. Caso, já tenha instalado o XAMPP, pule para o passo 3.

c) **Passo 3**: após instalar o XAMPP, copie os arquivos, baixados no **Passo 1**, na pasta "htdocs" do servidor XAMPP. Se estiver utilizando o *Windows*, a pasta encontra-se em "X:\xampp\htdocs", conforme apresentado na Figura 28. Caso esteja utilizando o *Linux*, em geral, a pasta encontra-se em "/opt/lampp/htdocs".

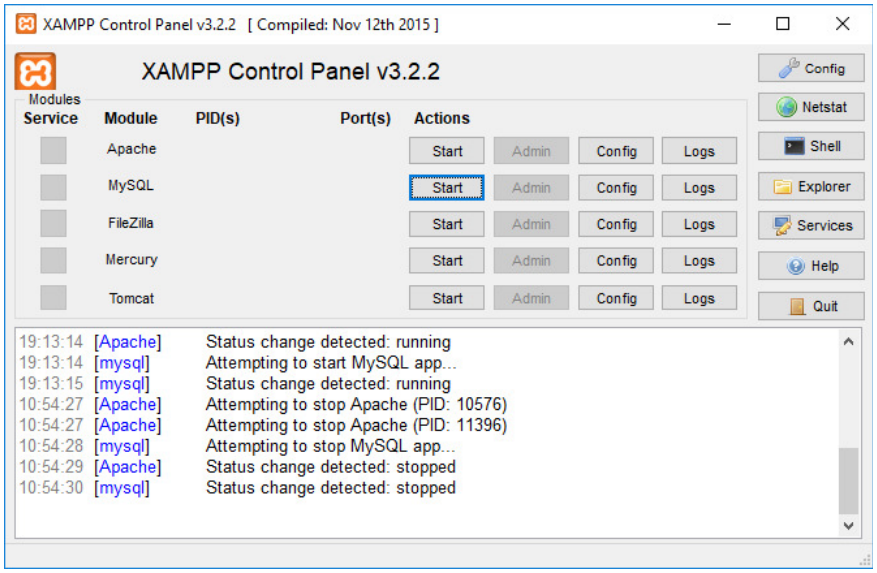
Figura 28 – Copiando os arquivos para o servidor XAMPP.



Fonte – Autora (2019).

d) **Passo 4**: inicie o servidor XAMPP, clicando duas vezes em seu arquivo ".exe" ou ".sh". Em seguida, inicie os serviços Apache e MySQL clicando nos botões "*start*" contido no painel principal do XAMPP, conforme apresentado na Figura 29.

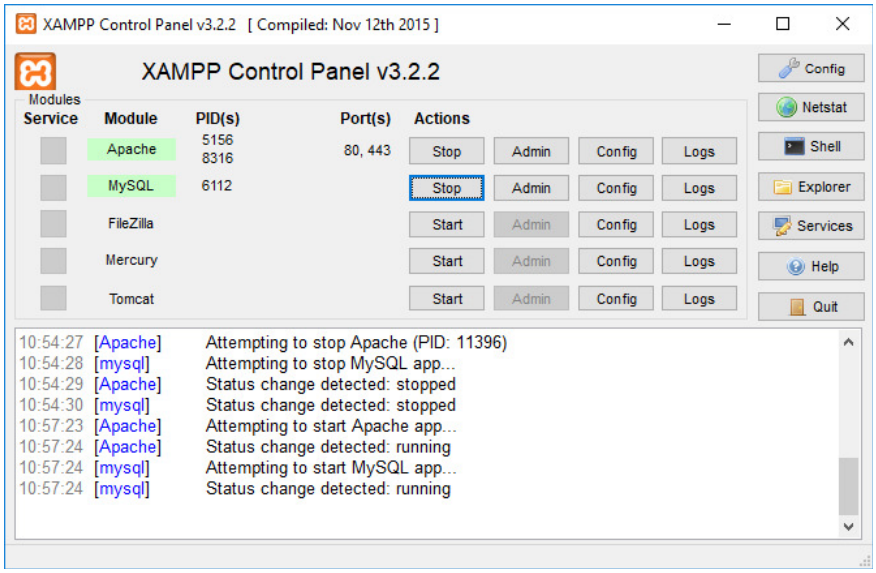
Figura 29 – Iniciando o servidor XAMPP.



Fonte – Autora (2019).

Note que o XAMPP terá como padrão as portas 80, para o serviço apache e porta 3306 para o serviço MySQL, conforme apresentado na Figura 30.

Figura 30 – Iniciando o servidor XAMPP.



Fonte – Autora (2019).

e) **Passo 5**: Caso o XAMPP tenha realizado mudanças nas portas padrões dos seus serviços, basta mudar a porta, no **Passo 3**, e funcionará normalmente. Após o início dos serviços, utilizando qualquer navegador *Web* acesse a URL: "<http://localhost/smart_e-ping-2/public/>".

Seguindo estes passos, ao acessar a URL descrita no **Passo 5** será exibida uma tela inicial do *Framework Smart* e-PING (conforme apresentado na Figura 10 na Seção 4.1 do Capítulo 4). Portanto, a ferramenta está configurada corretamente e pronta para realizar interoperabilidade entre o *Software* Público e a plataforma FIWARE.

Anexos

ANEXO A – Premiação