UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Estudo de Componentes de FIWARE Para IoT e Cidades Inteligentes

Trabalho de Conclusão de Curso

Felipe Matheus Conceição da Silva

São Cristóvão – Sergipe

2019

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Felipe Matheus Conceição da Silva

Estudo de Componentes de FIWARE Para IoT e Cidades Inteligentes

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a): Giovanny Fernando Lucero Palma

São Cristóvão – Sergipe

2019

Dedico este trabalho à minha família e aos amigos que me deram o apoio necessário para realizá-lo.

Agradecimentos

Agradeço a minha mãe por ter sido a força que me foi necessária para concluir essa graduação e este trabalho, por ter me dado o apoio sempre que eu precisava e por ter me encorajado quando eu não consegui sozinho. Agradeço ao meu pai por ter sido sempre um exemplo de dedicação e trabalho árduo. Agradeço aos meus irmãos por terem assumido várias responsbilidades no meu lugar e me apoiado. Agradeço a minha avó pelo carinho e apoio que só uma avó é capaz de dar.

Agradeço aos meus amigos por terem me ajudado a equilibrar tão bem procrastinação e produtividade e por sempre estarem presentes quando eu precisei de um ombro para chorar.

Por fim, agradeço ao meu orientador por ter sido tão dedicado, presente e paciente durante o processo de escrita desse trabalho.

Resumo

A Internet das Coisas (IoT) é um conceito tecnológico que vem se estabelecendo aos poucos no dia-a-dia das pessoas, criando benefícios que passam a não ser notados, uma vez que se incorporam perfeitamente como soluções. Uma dessas soluções são as Cidades Inteligentes, que utilizam os dados e tecnologias fornecidos pela IoT para melhorar diversos aspectos da vida urbana. Um ambiente que pode ser considerado como uma representação em pequena escala das necessidades percebidas em uma cidade é o de uma universidade. Para entender como implementar as soluções oferecidas pela IoT e pelas Cidades Inteligentes, este trabalho tem o objetivo de realizar um estudo bibliográfico que deixa explícitos os desafios impostos por essas tecnologias. Como forma de entender como superar alguns desses desafios, é realizado também o estudo da ferramenta FIWARE, que apresenta componentes projetados especificamente para facilitar as implementações de aplicações inteligentes. Por fim, propõe-se a implementação de uma aplicação de teste que simula uma sala de aula inteligente utilizando os conceitos pesquisados e os componentes Orion Context Broker e IDAS (IoT Agents) do FIWARE.

Palavras-chave: Internet das coisas, cidades inteligentes, FIWARE.

Abstract

The Internet of Things (IoT) is a technological concept that is gradually becoming established in people's daily life, creating benefits that are not noticed, once they are integrated seamlessly as solutions. One example of these solutions are Smart Cities, which use data and technology provided by IoT to improve many aspects of urban life. A university is an environment that could be perceived as a small-scale representation of the needs of a city. In order to understand how to implement the solutions offered by the IoT and Smart Cities, this work intends to carry out a bibliographic study to list the challenges imposed by these technologies. As a way to understand how to overcome some of these challenges, a study on the FIWARE platform is also carried out, as this platform contains components designed specifically to facilitate the implementation of smart applications. Finally, this work proposes the implementation of a test application which simulates a smart classroom using the researched concepts and FIWARE components Orion Context Broker and IDAS (IoT Agents).

Keywords: Internet of things, smart cities, FIWARE.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IoT	Internet of Things
MIT	Massachusetts Institute of Technology)
RFID	Radio-Frequency IDentification
uID	Unique/Universal/Ubiquitous IDentifier
NFC	Unique/Universal/Near Field Communications
WSAN	Wireless Sensorand Actuator Networks
IP	Internet Protocol
IEEE	Institute of Electrical and Electronic Engineers
WSN	Wireless sensor network
SoA	Service Oriented Architecture
REST	REpresentational State Transfer
IETF	Internet Engineering Task Force
OMA	Open Mobile Alliance
HTTP	Hypertext Transfer Protocol
CoAP	Constrained Application Protocol
MQTT	Message Queuing Telemetry Transport
LWM2M	Lightweight Machine to Machine
TDM	Technology Driven Method
HDM	Human Driven Method
CIoT	Cognitive Internet of Things
FI-PPP	Future Internet Public Private Partnership
GE	Generic Enabler
API	Application Programming Interface
NGSI	Next Generation Service Interfaces

JSON	JavaScript Object Notation
URL	Uniform Resource Locator
ASCII	American Standard Code for Information Interchange
SGBD	Sistema de Gerenciamento de Banco de Dados
HDFS	Hadoop Distributed File System
IoTEP	IoT Energy Plataform

Sumário

1

Introdução

A Internet das Coisas (IoT) surgiu como um termo usado por Kevin Ashton em 1999 para explicar a utilização de *tags* de identificação RFID em conjunto com a Internet. Quase 20 anos após essa definição inicial, essa tecnologia é conhecida, entre outras coisas, por ser uma rede de dispositivos interconectados pela Internet com o objetivo de torná-los mais inteligentes em termos de entradas e dados produzidos (ALDRIDGE, 2016). Além disso, a IoT tem como um de seus trunfos a possibilidade de tornar-se vetor para alcançar o desenvolvimento tecnológico em áreas tão diversas como construções, robótica, transporte, planejamento urbano, educação, etc. (INFSO. . . , 2008). Para que isso se torne realidade presente no dia-a-dia das pessoas, é necessário que se entenda e supere os principais desafios apresentados pela IoT, que incluem segurança e privacidade, heterogeneidade de hardware, eficiência energética e gerenciamento dos dados (MIORANDI et al., 2012). Alguns desses desafios já estão sendo superados com o auxílio de ferramentas, como *frameworks* de programação, embora sejam esperados ainda grandes avanços nos próximos anos (GARTNER, 2017).

Entre as áreas que recebem contribuição direta e apresentam potencial de serem ainda mais desenvolvidas pela Internet das Coisas, o planejamento urbano é bem representado pelas Cidades Inteligentes. As Cidades Inteligentes são o resultado da utilização dos dados e tecnologias fornecidos pela IoT para melhorar os aspectos de sustentabilidade, bem-estar populacional e eficiência energética do ambiente urbano. Embora seja uma área muito bem estudada, as Cidades Inteligentes apresentam desafios que incorporam e expandem aqueles impostos pela IoT. Uma vez que uma cidade moderna é formada por partes tão distintas (moradores, indústrias, planejadores, etc.), torna-se ainda mais difícil a interação entre componentes inteligentes de fornecedores diferentes (Ahlgren; Hidell; Ngai, 2016).

Para superar efetivamente os desafios impostos pela heterogeneidade e escala necessários para o desenvolvimento das Cidades Inteligentes e de outras aplicações de IoT, a plataforma FIWARE surgiu como uma iniciativa para facilitar essas implementações. O FIWARE tem

como funcionalidade essencial o gerenciamento de informações de contexto de entidades a partir da utilização do componente Orion Context Broker através de operações NGSI (FIWARE FOUNDATION, 2019c). Além deste, existem diversos outros componentes que fazem parte do ecossistema dessa plataforma, que fornecem às aplicações inteligentes funcionalidades de interface com IoT, gerenciamento e publicação de dados, processamento, análise e visualização de informações de contexto. Em especial o IDAS, responsável por gerenciar medições e comandos de dispositivos inteligentes que utilizam protocolos diferentes com os chamados IoT Agents e conectá-los através de seus contextos registrados no Orion (FIWARE FOUNDATION, 2019d).

Devido à necessidade de acompanhar avanços tecnológicos e científicos recentes no que diz respeito a IoT e integração de tecnologias, assim como também com o intuito de entender quais são as limitações existentes e quais são as alternativas tecnológicas para implementar aplicações voltadas para Cidades Inteligentes, neste trabalho foram realizados estudos sobre Internet das Coisas, Cidades Inteligentes, FIWARE e seus componentes principais. A partir destes estudos, espera-se entender os principais desafios e problemas enfrentados no desenvolvimento dessas aplicações. Conhecendo as definições e os desafios, pretende-se entender como utilizar o FIWARE para lidar com esses problemas. Por fim, propõe-se a implementação de uma aplicação que tem como objetivo demonstrar a utilização de sensores e atuadores como fornecedores de dados para um componente responsável por gerenciar as informações de contexto, sem a necessidade de lidar explicitamente com diversos protocolos envolvendo diversos equipamentos.

O restante desse trabalho é dividido de forma a compreender os assuntos justificados anteriormente. No capítulo 2, é realizado um estudo bibliográfico sobre Internet das Coisas e Cidades Inteligentes, abordando suas origens, aprofundando seus benefícios, investigando os problemas e desafios impostos por essas tecnologias e tocando em algumas das soluções possíveis para eles. No capítulo 3, é realizada uma pesquisa sobre a plataforma FIWARE, sua origem, principais funcionalidades, passando pela especificação da interface NGSI e analisando o funcionamento dos componentes principais da plataforma. No capítulo 4, é feita a descrição do desenvolvimento e execução da aplicação de teste que coloca em prática os resultados dos estudos realizados nos capítulos anteriores utilizando os componentes Orion Context Broker e IDAS do FIWARE. Por fim, no capítulo 5 são analisadas as conclusões do trabalho elaborado, listando as limitações existentes durante sua concepção e sugerindo contribuições possíveis de trabalhos futuros.

2

Referencial Teórico

2.1 Internet das Coisas

Um dos artigos mais citados na literatura sobre Internet das Coisas é (WEISER, 1991). Nele, o autor não chega a definir o conceito conhecido atualmente como Internet das Coisas (IoT), mas discorre sobre o ponto que divide as tecnologias comuns daquelas verdadeiramente profundas. Para ele, essa divisão está relacionada à ubiquidade da tecnologia em questão. Por exemplo, a forma como capturamos uma representação simbólica da linguagem falada atualmente é considerada uma tecnologia ubíqua, uma vez que não é necessário pensar em como isso é feito. Em contrapartida, embora computadores estejam muito presentes, eles ainda não podem ser considerados uma "parte integral e invisível" do modo de viver das pessoas. No artigo, são ainda discutidas as diversas abordagens de construção de computadores ubíquos, e é previsto que a tecnologia necessária para chegar à computação ubíqua é resumida a computadores de baixo custo e baixo consumo de energia, todos conectados por uma rede, com softwares que implementem aplicações ubíquas. O autor ainda faz uma narrativa de como seria um futuro em que a computação ubíqua estivesse presente e já demonstra preocupação com fatores como segurança e privacidade.

Acredita-se que o pesquisador Kevin Ashton, do MIT (Massachusetts Institute of Technology), tenha sido a primeira pessoa a utilizar a expressão *Internet of Things* durante uma apresentação realizada na Procter & Gamble (P&G), em 1999, ao explicar a relação da utilização de *tags* RFID pela empresa e a Internet. Embora tenha cunhado a expressão naquele ano, o pesquisador acredita que computadores ainda são completamente dependentes dos seres humanos para conseguirem informações (ASHTON, 2009). Ashton afirma que "hoje a tecnologia da informação é tão dependente de dados originados por pessoas que nossos computadores sabem mais sobre ideias que sobre coisas". A utilização de RFID e sensores serve para que os computadores sejam os responsáveis por adquirir e processar as informações sobre o mundo ao

seu redor sem dependerem de seres humanos.

Em (ATZORI; IERA; MORABITO, 2010), os autores tentam descrever o paradigma da Internet das Coisas de acordo com as diferentes visões existentes. Segundo os autores, a compreensão do significado de Internet das Coisas pode ser confusa, uma vez que, devido ao grande interesse científico no assunto, existem diversas definições. Para eles, essa confusão é em parte causada pela utilização do termo "Internet das Coisas" quando utilizado pelos *stakeholders* (corpos de padronização e pesquisa, negociadores, etc.) da forma que mais se aproxima ao seu interesse específico, o qual pode se referir ao mesmo tempo a uma visão orientada à rede e a uma visão orientada a "objetos" genéricos a serem integrados em um *framework* comum. Mesmo com essa confusão, eles enfatizam que o significado apropriado do termo Internet das Coisas é "uma rede global de objetos interconectados unicamente endereçáveis, baseada em protocolos de comunicação padrão" (INFSO. . . , 2008). Adicionalmente, uma outra definição de Internet das Coisas é orientada à representação semântica, que se relaciona com os desafios de apresentação e armazenamento das informações. Uma possível intersecção dessas três visões resultaria na definição mais substancial, uma vez que cada uma delas separadamente é muito limitada.

Ainda em (ATZORI; IERA; MORABITO, 2010), a estreita relação que existe entre a definição inicial de Internet das Coisas e *tags* RFID é novamente abordada. A arquitetura uID (*Unique/Universal/Ubiquitous IDentifier*) também é citada nesse contexto. No entanto, para os autores, essas tecnologias referem-se apenas ao aspecto de identificação de objetos, e destacam que a Internet das Coisas é muito mais abrangente. Eles mencionam outros autores que acreditam que tecnologias como NFC (*Near Field Communications*) e WSAN (*Wireless Sensor and Actuator Networks*) contribuirão para o aumento da "inteligência" das Coisas e levarão ao real desenvolvimento da IoT (PRESSER; GLUHAK, 2009). Além disso, citam (DUNKELS; VASSEUR, 2009) no que se refere a uma visão orientada à Internet, mais especificamente em relação aos protocolos que serão utilizados pela IoT, sendo o IP (*Internet Protocol*) o mais apropriado. Em (HUI; CULLER; CHAKRABARTI, 2009) é proposto incorporar o IP nas redes IEEE 802.15.4 para utilização na IoT. No contexto da visão de IoT orientado à semântica, citam (TOMA; SIMPERL; HENCH, 2009), que propõem soluções de modelagem utilizadas para enfrentar os problemas de representação, armazenamento, conexão, pesquisa e organização de informações geradas pela IoT.

Em (MIORANDI et al., 2012) fala-se sobre a evolução que acontece atualmente na Internet, acarretada pelo crescente número de usuários dessa rede e das inovações tecnológicas possibilitadas pela utilização de circuitos eletrônicos em objetos comuns. Os autores escrevem sobre a definição de IoT do ponto de vista conceitual, sob as perspectivas de sistema e serviço. Conceitualmente, a IoT baseia-se na habilidade de objetos de serem identificáveis, comunicáveis e interativos. Do ponto de vista do sistema, os autores dizem que a IoT é basicamente um sistema que apresenta dinamismo e alta distributividade, tendo como componentes objetos inteligentes que têm a habilidade de produzir e consumir informação. A perspectiva de serviço refere-se à

forma como os dados gerados pelos objetos são traduzidos em serviços. Os autores analisam as oportunidades que surgirão com a IoT e afirmam que ela permitirá monitoramento ambiental, de saúde, inventário, produção, segurança e vigilância. Eles concordam que a IoT vai evoluir a partir de tecnologias de identificação (como RFID), e em seguida expandirá serviços Web a partir de arquiteturas de redes de sensores sem fio (WSN) e orientadas a serviços (SoA). Adicionalmente, o artigo lista as características necessárias e os desafios impostos pela Internet das Coisas: heterogeneidade de dispositivos, escalabilidade, troca de dados, energia, organização, interoperabilidade, gerenciamento de dados, segurança e privacidade.

Em (MIORANDI et al., 2012) há uma discussão sobre duas características essenciais da IoT que representam os principais desafios de pesquisa referentes ao assunto: identificação e sensoriamento/atuação. No que se refere à identificação dos objetos, é possível utilizar tecnologias de baixo consumo de energia, que, em sua maioria, fazem uso de baterias. No entanto, essa solução pode gerar problemas relacionados à reposição das baterias. Vários estudos que buscam implementar tecnologias que resolvam esse problema são citados, embora tal preocupação não seja tão urgente, uma vez que a escala de implantação de dispositivos de IoT ainda não é tão grande. Outro ponto tocado pelos autores é a implementação da IoT como um sistema distribuído, uma vez que será necessário ter muitos dados fluindo dos objetos. Felizmente, os autores alegam que o controle de fluxo distribuído é um assunto bem estudado, embora não se tenha demonstrado grande preocupação quando se trata de IoT.

No que se refere a segurança no contexto da Internet das Coisas, os autores citam que, nas implantações iniciais, soluções relacionadas a segurança eram tratadas de maneira *ad hoc*. Eles investigam três principais problemas de segurança que surgem no contexto atual de IoT: confidencialidade de dados, privacidade e confiança. Para atingir um bom nível de confidencialidade de dados, é necessário definir um mecanismo de controle de acesso, uma linguagem de consulta para buscar as informações desejadas, e um processo de autenticação de objetos. A privacidade tem um papel muito importante ao se considerar os domínios e as tecnologias usadas para implementar a Internet das Coisas. Dessa forma, deve-se definir um modelo geral, técnicas e soluções para privacidade em IoT. Para ter serviços de IoT com confiança, uma linguagem de negociação de confiança deve ser introduzida e devem ser definidos um mecanismo de negociação de confiança, um sistema de gerenciamento de identidade e um *framework* de gerenciamento de confiança (MIORANDI et al., 2012).

Em (RAHMAN; OZCELEBI; LUKKIEN, 2016) configuração, programação e ciclos de vida são listados como desafios a serem superados no desenvolvimento de aplicações de IoT. Esses problemas podem facilmente ser vistos como ramificações dos apresentados anteriormente e resumidos em dificuldades de incluir cada dispositivo como membros de um sistema, programados de forma correta e eficiente considerando arquiteturas e protocolos heterogêneos em estágios de desenvolvimento diversos. Os autores apresentam a utilização de *frameworks* de programação como uma solução para esses problemas. Em (Derhamy et al., 2015) um *framework* IoT é

descrito como sendo uma ferramenta que permite implementação rápida, interoperabilidade, manutenibilidade, segurança e flexibilidade de tecnologia para uma aplicação de Internet das Coisas. Em (RAHMAN; OZCELEBI; LUKKIEN, 2016) é realizado um estudo dos *frameworks Works with Nest*, *ARM mbed IoT Device Platform* e *Alljoyn* e suas arquiteturas, projetando uma taxonomia de aspectos arquiteturais, uma vez que a abordagem do *framework* depende de sua arquitetura.

A taxonomia apresentada em (RAHMAN; OZCELEBI; LUKKIEN, 2016) busca classificar os *frameworks* estudados a partir das diferenças e semelhanças entre as arquiteturas. O trabalho separa as ferramentas de acordo com quatro aspectos: implantação de controle, implantação de gerenciamento de dispositivo, implantação de armazenamento, e protocolo de aplicação. Os três primeiros aspectos estão relacionados aos três paradigmas de computação existentes (*cloud*, *fog* e *mist*), isto é, ao local no qual as tarefas de computação serão realizadas (na nuvem, em dispositivos intermediários na borda da rede local, ou nos nós finais na rede local, respectivamente). O quarto aspecto refere-se ao protocolo de aplicação escolhido pelo *framework*, que pode ou não ser um protocolo padrão e segue um estilo de arquitetura entre os estilos SoA (*Service Oriented Architecture*, em que há pouco espaço para lógica de aplicação e separa-se funcionalidade de coordenação), *publish/subscribe* (em que há um *broker* que recebe os eventos dos *publishers*, aos quais os *subscribers* demonstram interesse ao se subscreverem) e REST (*REpresentational State Transfer*, que simplifica protocolos, interoperabilidade e desenvolvimento), dependendo das restrições apresentadas pelos dispositivos utilizados na aplicação IoT. Cada um dos *frameworks* é classificado analisando esses quatro aspectos em termos dos efeitos causados nos dispositivos e na aplicação, como aumento de latência, adição de pontos de falha, segurança, escalabilidade, etc. Os autores chegam à conclusão de que, a partir de uma tabela de pontos construída utilizando os aspectos escolhidos na criação da taxonomia, é possível classificar qualquer *framework* de interesse levando em conta as características mais importantes para a aplicação inteligente em que ele será utilizado.

Uma das características importantes na implementação de uma aplicação de IoT é o protocolo utilizado. Com o interesse de aumentar a interoperabilidade e segurança da aplicação, é preferível utilizar protocolos padronizados, isto é, protocolos que tenham sido aceitos e adotados por algum órgão de padronização (IETF, IEEE, OMA, etc.). Entre os protocolos padronizados mais utilizados em aplicações de IoT estão o *Hypertext Transfer Protocol* (HTTP, o protocolo fundamental para qualquer troca de informação na Internet, no qual clientes e servidores trocam mensagens individuais (MDN WEB DOCS, 2019)), o *Constrained Application Protocol* (CoAP, um protocolo projetado para a utilização em redes e dispositivos com limitações (BORMANN, 2016)), que seguem o estilo de arquitetura REST; e o *Message Queuing Telemetry Transport* (MQTT, protocolo projetado para a troca de mensagens extremamente leves, permitindo a utilização em redes muito limitadas (MQTT, 2019)), que segue o padrão *publish/subscribe*. Além disso, o *Lightweight Machine to Machine* (LWM2M) é um protocolo que utiliza o CoAP para realizar comunicação entre clientes e servidores, e o OneM2M é um protocolo que incorpora

os protocolos LWM2M, HTTP, CoAP e MQTT (RAHMAN; OZCELEBI; LUKKIEN, 2016).

Em (ROBERT et al., 2017) são citados (Ahlgren; Hidell; Ngai, 2016), que ao escrevem que mais um problema da IoT é a formação de "silos verticais", isto é, "sistemas proprietários fechados dedicados a uma tarefa em particular", causados pela dificuldade ou impossibilidade de interação entre dispositivos e componentes de diferentes fornecedores. Para os autores, o maior exemplo desse problema é representado pelas Cidades Inteligentes, uma vez que elas reúnem em uma mesma implementação diferentes planejadores, organizações financeiras, fornecedores de serviços e de telecomunicação, indústrias, moradores, etc., e cada um desses *stakeholders* pode incluir objetos inteligentes diversos que contribuem ainda mais para esse problema.

2.2 Cidades Inteligentes

Em (Ahlgren; Hidell; Ngai, 2016) Cidades Inteligentes são definidas como o resultado da integração de várias soluções tecnológicas da perspectiva de desenvolvimento urbano com a intenção de produzir um ambiente sustentável, melhorar o bem-estar da população e aumentar eficiência e o potencial financeiro a partir dos recursos de uma cidade. Em (BRAUN et al., 2018) são citados (ELMAGHRABY; LOSAVIO, 2014) ao utilizarem a definição da IBM de que cidadãos e componentes de uma Cidade Inteligente estão/são "Instrumentados, Interconectados e Inteligentes" (IN3), além de adicionarem a preocupação com eficiência, segurança e conveniência que as cidades inteligentes supostamente devem oferecer. Em (ALAVI et al., 2018) é demonstrada a crença que Cidades inteligentes proporcionam as tecnologias necessárias para ajudar a resolver problemas de gerenciamento, qualidade de vida e sustentabilidade das cidades modernas, utilizando dados baseados em Internet das Coisas.

Em (KUMMITHA; CRUTZEN, 2017) são apresentadas duas formas de se falar sobre Cidades Inteligentes que vêm sendo utilizadas pela comunidade científica. A primeira forma é conhecida como método orientado a tecnologia (*technology driven method* - TDM) e representa a ideia de que Cidades Inteligentes são formadas por locais conectados nos quais são aplicados recursos tecnológicos para melhorar a qualidade de vida da população. A segunda forma é conhecida como método orientado a seres humanos (*human driven method* - HDM) e surge do fato de que se acredita que as tecnologias da informação por si só não seriam suficientes para melhorar os aspectos de qualidade de vida de uma cidade, tornando necessário melhorar o capital humano e outras habilidades de desenvolvimento dos cidadãos. Para corroborar com esse último ponto de vista, os autores citam exemplos de Cidades Inteligentes cujos aspectos inteligentes demonstram pouca preocupação com os problemas de desigualdade social existentes nas cidades. De posse dessas informações e a partir do estudo da literatura sobre Cidades Inteligentes, o artigo identificou quatro escolas de pensamento (*restrictive*, *reflective*, *rationalistic* e *critical*) existentes na comunidade científica e propõe a definição de um *framework* chamado 3RC de acordo com essas escolas. A escola *restrictive* está mais ligada ao método orientado a tecnologia, enquanto

as escolas *reflective* e *rationalistic* vão gradualmente aproximando-se do método orientado a seres humanos. Em contrapartida, a escola *critical* acredita que Cidades Inteligentes não estão ligadas a nenhum desses dois métodos e que são resultado de *lobby* neoliberal.

Em (ALAVI et al., 2018) é citada a empresa de pesquisa Gartner, que previu em sua pesquisa anual de tendências (GARTNER, 2017) que a Internet das Coisas e as Cidades Inteligentes levarão em torno de 5 a 10 anos para se consolidarem. Os autores listam as cinco camadas de serviços básicos de IoT para o desenvolvimento de Cidades Inteligentes (dispositivo, rede, *middleware*, aplicação, negócios). Para eles, a implantação de componentes inteligentes nas cidades está relacionada diretamente ao custo e à disponibilidade tecnológica. Eles ainda citam diversos projetos que aconteceram no Sétimo Programa de *Framework* para Pesquisa e Desenvolvimento Tecnológico (FP7) na Europa que focam na implantação da IoT em diversas aplicações. Eles elencam os principais domínios em que os princípios de Cidades Inteligentes têm sido implementados, que são crescimento populacional, lotação e tráfego.

Os autores fazem uma análise mais aprofundada dos avanços em IoT nos domínios de mobilidade e transporte, casas e infraestrutura civil, vendas e cuidados com a saúde, e energia. Para chegar ao transporte inteligente, é necessário que existam diferentes redes de IoT que lidem com análise dos dados. Neste contexto, eles citam iniciativas que já foram realizadas para atingir uma melhora desse aspecto em aplicações como estacionamentos para carros elétricos, e apresenta o conceito de um sistema de monitoramento de bicicletas elétricas. Em relação a casas e infraestrutura civil inteligente, os avanços podem ser alcançados através da análise de dados obtidos de sensores sem fio. Eles citam o surgimento de novos paradigmas resultantes dos grandes esforços nesse setor, como a IoT cognitiva (CIoT) e o paradigma de "sensoriar agora, recuperar depois", utilizando técnicas de obtenção de dados baseados em RFID e os enviando para a nuvem através tecnologias IoT. Para vendas e cuidados com a saúde, o artigo menciona que esse setor pode tirar proveito dos dados obtidos através de dispositivos inteligentes usados pelas pessoas da cidade, como dispositivos vestíveis. Segundos os autores, grandes esforços vêm sendo feitos para alcançar avanços em tecnologias utilizadas por pessoas com necessidades especiais, incluindo ambulâncias monitoradas de forma inteligente e cadeiras de rodas conectadas à internet. No que se refere a energia, residências poderiam ser monitoradas e gerenciadas para diminuir o consumo. A eficiência energética é atualmente foco de diversas pesquisas que envolvem IoT, resultando, por exemplo, em aplicativos desenvolvidos e redes IoT utilizadas em projetos de *middlewares* para eficiência energética em espaços públicos utilizando arquiteturas orientadas a serviços (SoA) (ALAVI et al., 2018).

A utilização de tecnologias de Internet das Coisas e Cidades Inteligentes gera esforços que vão desde a obtenção/sensoriamento de dados até o armazenamento e utilização dos dados obtidos (TERROSO-SAENZ et al., 2019), passando por diversas fases intermediárias que podem incluir o intercâmbio desses dados entre dispositivos que utilizem tecnologias e protocolos completamente distintos. Como foi citado nas Seções 2.1 e 2.2, uma forma de lidar com essas dificuldades é a

partir da utilização de *frameworks* de programação que abstraem alguns dos detalhes. Uma das alternativas mais bem estabelecidas atualmente é a utilização da plataforma FIWARE, que conta com componentes criados especificamente para a implementação de aplicações inteligentes, considerando o uso de diversos tipos de dispositivos com diferentes protocolos, de maneira simples e bem documentada.

3

FIWARE e seus componentes

Em 2011, a Comissão Europeia criou o programa *Future Internet Public Private Partnership* (FI-PPP) com o objetivo de aproveitar os benefícios da internet do futuro e contribuir para o aumento da inteligência de diversos processos na Europa. A partir de um investimento inicial de 300 milhões de euros, a iniciativa levou ao surgimento do FIWARE, trazendo avanços em diversos campos da tecnologia da informação (EUROPEAN COMMISSION, 2018)(FIWARE FOUNDATION, 2017).

O FIWARE é uma plataforma (ou um *framework*) que contém diversos componentes de código aberto que podem ser usados em conjunto ou individualmente, e com componentes de terceiros para implementar projetos inteligentes (FIWARE FOUNDATION, 2019c). As tecnologias do FIWARE visam à combinação da Internet das Coisas com Gerenciamento de Informações de Contexto e serviços de *Big Data* na nuvem para facilitar o processamento, análise e visualização dessas informações. Isso permite que a plataforma seja usada na implementação de soluções bem elaboradas de Cidades Inteligentes, agricultura inteligente e indústria inteligente (FIWARE FOUNDATION, 2019j).

Uma das formas de utilizar todas as capacidades disponibilizadas pelo FIWARE é através do FIWARE *Lab*. O FIWARE *Lab* é descrito como "um ambiente não-comercial para inovação e experimentação" (FIWARE FOUNDATION, 2019a). Essa plataforma possibilita a criação de contas de usuário em que máquinas virtuais e endereços IP flutuantes podem ser configurados para a utilização dos componentes do FIWARE. Além disso, são disponibilizadas informações de contexto referentes a dispositivos e sensores reais espalhados pelo mundo inteiro, que podem ser utilizadas em aplicações por qualquer desenvolvedor com acesso. A plataforma é desenvolvida sobre uma arquitetura em camadas, permitindo alta disponibilidade, usando o *software* OpenStack para a criação das máquinas virtuais.

Os componentes do FIWARE são conhecidos como *Generic Enablers* (GEs) e possuem APIs bem definidas que implementam diversas funcionalidades definidas pela plataforma. Existe

uma gama de GEs já implementados para o uso de interface com IoT, robôs e sistemas de terceiros, gerenciamento, publicação e monetização de dados de contexto, e processamento, análise e visualização de informações de contexto. As implementações de GEs mais bem estabelecidas (também definidas como não estando em incubação) incluem o Orion Context Broker (para o gerenciamento de informações de contexto), Cygnus (para a criação de históricos das informações de contexto), IDAS (para comunicação entre dispositivos inteligentes com diferentes protocolos), Wirecloud (para a criação de aplicações *web* por usuários com poucas habilidades de programação), Keyrock (para autenticação de usuários e dispositivos) e CKAN extensions (para publicação de conjuntos de dados a partir de informações de contexto) (FIWARE FOUNDATION, 2019d). A **??** mostra a localização conceitual de cada um desses componentes. Além desses, também é possível construir novas implementações de GEs a partir da descrição de arquitetura do FIWARE. Para que uma aplicação criada por um desenvolvedor seja considerada "*Powered by FIWARE*", é obrigatório o uso do Orion Context Broker, uma vez que ele traz a funcionalidade que é oferecida como pilar do FIWARE: o gerenciamento de informações de contexto (FIWARE FOUNDATION, 2019d). Para entender o Orion Context Broker, é necessário entender a API NGSI (*Next Generation Service Interface*) do FIWARE.

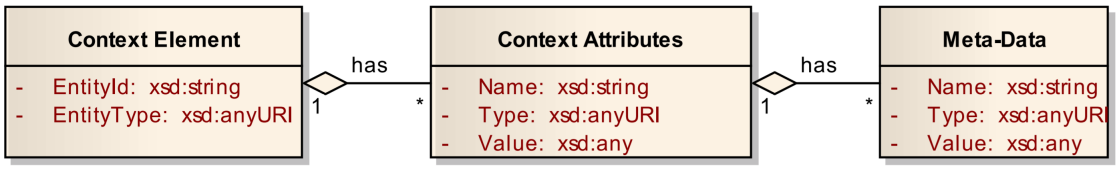
Figura 1 – Componentes do FIWARE

Adaptado de (FIWARE FOUNDATION, 2019d)

3.1 A Interface NGSI

A NGSI para o FIWARE é baseada nas especificações de gerenciamento de contexto NGSI definidas pela OMA (Open Mobile Alliance), que fornecem interfaces para gerenciar informações de contexto e para permitir acesso (consulta, subscrição e notificação) acerca de entidades registradas. A interface da OMA oferece as especificações NGSI-9 e NGSI-10. A NGSI- 9 é uma API RESTful via HTTP que possui como principais operações o registro e a descoberta de entidades de contexto, e a subscrição e notificação baseadas na descoberta de entidades de contexto. A NGSI-10 também é uma API RESTful via HTTP com operações de atualização de contexto, consulta, subscrição e notificação. Essa interface define um modelo de informação de contexto, que descreve como essas informações são estruturadas e associadas a entidades de contexto (pessoa, grupo, lugar, evento, coisa), que possuem um id de entidade (`EntityId`) e um tipo de entidade (`EntityType`). Para cada entidade, existem os atributos de contexto, que possuem nome e metadados opcionais (ambos com nome, tipo e valor). Um diagrama desse modelo pode ser visto na Figura 2.

Figura 2 – Modelo de informações de contexto



Fonte: OMA (2012, p. 13)

A implementação de referência da API NGSI do FIWARE é o Orion Context Broker, e a API recebe as denominações NGSIv1 e NGSIv2. A NGSIv1 fornece as interfaces NGSI-9 e NGSI-10, com as operações padrão descritas anteriormente, além das chamadas operações de conveniência, que foram definidas na implementação do FIWARE para facilitar o uso da interface, permitindo o registro de aplicações produtoras de contexto, atualização das informações de contexto, recebimento de atualizações das informações de contexto e consulta às informações de contexto (TELEFÓNICA I+D, 2018). As operações padrão definidas pela NGSI-10 podem ser vistas na Tabela 1 enquanto que as operações de conveniência, na Tabela 2.

A NGSIv2 é uma versão renovada da API, que simplifica as operações que estavam disponíveis na v1 e torna possível a realização de operações mais complexas de forma ágil e orientada a implementação, focando em ser amigável para o desenvolvedor (FIWARE FOUNDATION, 2018b). Entidades são representadas nos corpos das requisições e respostas da API são retornadas no formato JSON que segue o modelo:

```
{
    "id": "entityID",
```

Tabela 1 – Operações padrão da NGSI-10.

Operação	**Função**
`/v1/updateContext`	Criar ou deletar entidades. O corpo da requisição deve conter uma lista com um elemento do tipo contextElement e um elemento do tipo `updateAction` (`APPEND` ou `DELETE`).
`/v1/queryContext`	Acessar as informações de contexto. O corpo da requisição deve conter uma lista com as informações de contexto desejadas (`entities`, `attributes`, `restriction`) e o corpo da resposta conterá `contextResponse`, `contextElement` e `statusCode`.
`/v1/subscribeContext`	Subscrever-se para receber notificações assíncronas quando ocorrerem mudanças a certas informações de contexto. Pode ser do tipo `ONTIMEINTERVAL` (para ser notificado a cada intervalo de tempo definido), com corpo contendo `entities`, `attributes`, `reference` (URL para onde as notificações serão enviadas), `duration` e `notifyConditions`, e o corpo da resposta conterá `subscribeResponse`. Ou do tipo `ONCHANGE` (para ser notificado quando um atributo muda), com corpo contendo os mesmos campos do tipo `ONTIMEINTERVAL`, além do campo `throttling` (para especificar um tempo mínimo de chegada entre notificações), e a resposta com o mesmo campo do tipo anterior.
`/v1/updateContextSubscription`	Atualizar o valor de atributos de entidades. O corpo de requisição deve conter `contextElements` e `updateAction` (`UPDATE` ou `APPEND`), e o corpo da resposta contém `contextResponse`, `contextElement` e `statusCode`.
`/v1/unsubscribeContext`	Cancelar uma subscrição. O corpo da requisição deve conter o `subscriptionId`, e o corpo da resposta contém `statusCode` e `subscriptionId`.

Fonte: (TELEFÓNICA I+D, 2018)

```
    "type": "entityType",
    "attr_1": <val_1>,
    "attr_2": <val_2>,
     ...
    "attr_N": <val_N>
}
```

E atributos seguem o padrão:

```
{
    "value": <...>,
    "type": <...>,
    "metadata": <...>
}
```

Tabela 2 – Operações de conveniência da NGSI-10.

Operação	**Função**

`/v1/contextEntities`
`/v1/contextEntities/{EntityID*}`
`/v1/contextEntities/{EntityID*}/attributes/{attributeName}`
`/v1/contextEntities/{EntityID*}/attributes/attributeName/{valueID}`
	Criar, verificar existência, atualizar, ou eliminar entidades de contexto (ou atributos específicos de uma entidade, e até metadados de atributos), dependendo do verbo utilizado (POST, GET, PUT, DELETE).

`/v1/contextTypes` `/v1/contextTypes/{entityType}`	Verificar tipos de todas as entidades de contexto ou informações detalhadas de um tipo de entidade de contexto específico.
`/v1/contextEntityTypes/{typeName}` `/v1/contextEntityTypes/{typeName}/attributes/{attributeName}`	Verificar todas as informações sobre todas as entidades com o tipo de entidade especificado ou todos os valores de um atributo especificado de todas as entidades de um tipo de entidade especificado.
`/v1/contextSubscriptions` `/v1/contextSubscriptions/{subscriptionID}`	Criar, verificar, atualizar ou eliminar subscrições especificadas.

Fonte: (TELEFÓNICA I+D, 2018)

As operações disponíveis na NGSIv2 são praticamente as mesmas da NGSIv1, com a utilização de identificadores um pouco diferentes na URL de requisição, além da adição da operação de registro de fornecedores de contexto externo e de operações em lote. A lista de operações básicas pode ser vista na Tabela 3.

A NGSIv2 permite dois modos de representação simplificada para entidades e atributos. No modo `keyValue`, os atributos da entidade são representados apenas por seus valores, sem informações sobre tipo e metadados, como pode ser visto:

```
{
    "id": "R12345",
    "type": "Room",
    "temperature": 22
}
```

No modo values, a entidade é representada como um vetor de valores de atributos, sem informações de id e tipo:

```
[ 'Ford', 'black', 78.3 ]
```

A API define restrições de tipos especiais de atributos que não podem ser definidos pelo usuário, como `dateTime`, `geo:point`, `geo:line`, `geo:box`, `geo:polygon` e `geo:json`. Também define atributos e metadados embutidos (como `dateCreated` e `dateModified`). Algumas das restrições de caracteres para IDs e tipos de entidade, nomes e tipos de atributos e

Tabela 3 – Operações básicas da NGSIv2.

Operação	**Função**

`/v2/entities`
`/v2/entities/{entityID}`
`/v2/entities/{entityID}/attrs/{attrName}`
`/v2/entities/{entityID}/attrs/{attrName}/{value}`
	Criar, verificar existência, atualizar, ou eliminar entidades de contexto (ou atributos específicos de uma entidade, e até metadados de atributos), dependendo do verbo HTTP utilizado (POST, GET, PUT, DELETE, respectivamente).

`/v2/types` `/v2/types/{typeID}`	Verificar os tipos de todas as entidades de contexto ou informações detalhadas de um tipo de entidade de contexto específico.
`/v2/subscriptions` `/v2/subscriptions/{subscriptionID}`	Criar, verificar, atualizar ou eliminar subscrições, dependendo do verbo HTTP utilizado (POST, GET, PUT, DELETE, respectivamente), podendo verificar, atualizar ou eliminar uma subscrição específica passando seu ID.
`/v2/registrations` `/v2/registrations/{registrationID}`	Criar, verificar, atualizar ou eliminar fornecedores de contexto externos, dependendo do verbo HTTP utilizado (POST, GET, PUT, DELETE, respectivamente), podendo verificar, atualizar ou eliminar um fornecedor de contexto externo passando seu ID.
`/v2/op/update`	Criar, atualizar e/ou eliminar várias entidades em uma única operação. O corpo da requisição é um objeto com as propriedades actionType (append, appendStrict, update, delete ou replace) e entities (especificando as entidades).
`/v2/op/query`	Consulta que retorna todas as entidades ou um array vazio `[]` caso nenhuma seja encontrada. O corpo da requisição pode conter os elementos entities (entidades a procurar), attrs (lista de atributos a serem retornados), expression (composta por `q`, `mq`, `georel`, `geometry` e `coords`) e `metadata` (nomes de metadados a serem incluídos).
`/v2/op/notify`	Consome o corpo de uma notificação do Orion para que os dados da entidade notificada sejam persistidos. Apresenta o mesmo comportamento da operação `/v2/op/update` com `actionType` `append`.

Fonte: (FIWARE FOUNDATION, 2018a)(GALÁN, 2016)

metadados incluem a utilização de caracteres ASCII (exceto `&`, `?`, `/` e), tamanho máximo de 256 caracteres e tamanho mínimo de 1 caractere. Além disso, as entidades podem ter propriedades geoespaciais, isto é, informações relacionadas a localização, que podem ser representadas como atributos de contexto normais (utilizando o formato de representação simples para geometrias básicas, como ponto, linha, caixa e polígono) ou o GeoJSON (formato baseado em JSON que permite representar propriedades geoespaciais mais complexas, como pontos de altitude ou formatos multigeométricos (MACWRIGHT, 2015)) (BUTLER et al., 2015). Dois exemplos de representação em formato de localização simples podem ser vistos abaixo.

```
{
  "location": {
    "value": "41.3763726, 2.186447514",
    "type": "geo:point"
  }
}
```

```
{
  "location": {
    "value": [
      "40.63913831188419, -8.653321266174316",
      "40.63881265804603, -8.653149604797363"
    ],
    "type": "geo:box"
  }
}
```

Enquanto que abaixo pode ser visto um exemplo de representação de um ponto utilizando GeoJSON.

```
{
  "location": {
    "value": {
      "type": "Point",
      "coordinates": [2.186447514, 41.3763726]
    },
    "type": "geo:json"
  }
}
```

A NGSIv2 também define um linguagem de consulta simples (*Simple Query Language*) que permite a realização de consultas através de requisições de forma mais enxuta que na NGSIv1.

Uma requisição pode ser formada por várias partes que são separadas por ponto-e-vírgula (`;`) e retorna todas as entidades que são verdadeiras para todas as condições passadas. A requisição contém o caminho do atributo, um operador e um valor. O caminho do atributo é formado por uma lista de *tokens* que endereçam o nome de uma propriedade JSON separados por ponto (`.`). O primeiro *token* na lista deve ser o nome de um atributo NGSI de uma entidade e os demais *tokens* devem representar o caminho do valor de atributo ou valor de metadados desejado. Caso a consulta seja filtrada por valor de atributo, utiliza-se o `q`; caso seja filtrada por metadados, utiliza-se `mq`. Os operadores incluem os de igualdade (== ou `:`), desigualdade (`!=`), menor (<), menor ou igual (<=), maior (>), maior ou igual (>=) e correspondência de padrão de expressão regular (`~=`). Também é possível utilizar apenas o nome de uma propriedade sem operador e valor para verificar sua existência ou o operador de não existência (`!`) seguido de uma propriedade para verificar sua não existência. Um exemplo de query realizada utilizando essa linguagem de consulta simples é fazer uma requisição `GET localhost:1026/v2/entities?q=temperature>22`. Em comparação, é possível realizar um requisição `POST localhost:1026/v2/op/query` com o corpo no formato JSON a seguir e obter o mesmo resultado.

```
{
    "entities": [
        {
            "idPattern": ".*",
            "type": "Room"
        },
    ],
    "expression": {
        "q": "temperature>22"
    },
}
```

A realização de consultas geográficas é possibilitada na NGSIv2 usando os parâmetros `georel` e `geometry`. O primeiro consiste de uma lista de `tokens`, em que o primeiro é o nome de um relacionamento (`near`, `coveredBy`, `intersects`, `equals`, `disjoint`) e os demais são modificadores que indicam mais informações sobre o relacionamento. O relacionamento `near` pode ser utilizado com os modificadores `maxDistance` e `minDistance`. O segundo parâmetro é `geometry`, que corresponde a uma forma (`point`, `line`, `polygon`, `box`) que é usada como referência no processamento da consulta. O parâmetro `georel` funciona como uma relação espacial entre as entidades encontradas e a geometria definida no parâmetro `georel`. O parâmetro `coords` é utilizado para especificar coordenadas geográficas para a forma geométrica passada no parâmetro `geometry`. Para que o resultado de uma consulta geográfica seja válido, é necessário que alguma entidade possua o atributo de localização correspondente, além de uma propriedade de metadado chamada `defaultLocation`. Um exemplo simples

de consulta geográfica utilizando a linguagem de consulta simples é fazer uma requisição `GET localhost:1026/v2/entities?georel=near;maxDistance:1000&geometry=point&coords=-40.4,-3.5`. Em comparação, o mesmo resultado poderia ser obtido fazendo uma requisição `POST localhost:1026/v2/op/query` com o corpo mostrado a seguir.

```
{
    "entities": [
        {
            "idPattern": ".*",
            "type": "Room"
        },
    ],
    "expression": {
        "georel": "near;maxDistance:1000",
        "geometry": "point",
        "coords": "40,4,-3.5"
    }
}
```

A NGSIv2 permite o registro de fornecedores de contexto que provêm informações de contexto. Para que isso seja possível, é necessário que o fornecedor seja implementado em alguma linguagem de programação ou que seja utilizado algum fornecedor previamente disponibilizado (como uma API, tais como a Twitter Search API ou a Open Weather Map API). Considerando um fornecedor de contexto implementado na URL `http://context-provider:3000/proxy/v1/random/weatherConditions`, o registro pode ser criado ao realizar uma requisição `POST localhost:1026/v2/registrations` com o corpo mostrado a seguir.

```
{
    "description": "Random Weather Conditions",
    "dataProvided": {
        "entities": [
            {
                "id": "urn:ngsi-ld:Store:001",
                "type": "Store"
            }
        ],
        "attrs": [
            "relativeHumidity"
        ]
    },
```

```
    "provider": {
        "http": {
            "url": "http://context-provider:3000/proxy/v1/random/weather
            Conditions"
        },
        "legacyForwarding": true
    }
}
```

Dessa forma, a entidade `urn:ngsi-ld:Store:001` poderá receber informações de contexto relativas ao atributo `relativeHumidity` da URL registrada (FIWARE FOUNDATION, 2019f).

Uma das capacidades mais importantes definidas pelas interfaces NGSI são o recebimento de notificações a partir da mudança em alguma entidade. Essa funcionalidade é implementada pelo Orion Context Broker e também é utilizada por diversos outros componentes que necessitam saber sobre as mudanças nas informações de contexto. A criação de uma subscrição consiste em uma requisição `POST localhost:1026/v2/subscriptions` em que o corpo da requisição contém a descrição da notificação, a(s) entidade(s) de interesse, a(s) condição(ões) para que seja enviada uma notificação e uma URL conhecida para a qual uma mensagem será enviada quando uma mudança ocorrer nas condições passadas. A URL passada é conhecida como acumulador e deve faz o papel de consumidor, e precisa ser capaz de receber as notificações enviadas (FIWARE FOUNDATION, 2018a). Por exemplo, é possível ver a criação de uma subscrição que envia notificações sempre que o atributo `price` é alterado em qualquer entidade do tipo `Product` (FIWARE FOUNDATION, 2019g).

```
{
    "description": "Notify me of all product price changes",
    "subject": {
        "entities": [{"idPattern": ".*", "type": "Product"}],
        "condition": {
            "attrs": [ "price" ]
        }
    },
    "notification": {
        "http": {
            "url": "http://tutorial:3000/subscription/price-change"
        }
    }
}
```

As mensagens de notificação definidas na NGSIv2 incluem os campos `subscriptionId` e `data`. Como os nomes sugerem, o primeiro campo corresponde ao ID da subscrição e o segundo corresponde a um vetor cujos campos se referem a cada entidade de interesse, juntamente com seus atributos. Essas mensagens podem ser exibidas na forma padrão (com o modificador `attrsFormat` definido como `normalized` ou simplesmente omitido, exibindo todas as informações sobre as entidades), na forma de entidade parcial (com o modificador `attrsFormat` definido como `keyValues`, e exibindo apenas os valores chave de cada entidade) ou na forma de atributo parcial (com `attrsFormat` definido como `value` e exibindo apenas os valores dos atributos). É possível ver abaixo exemplos de exibição em forma padrão, entidade parcial e atributo parcial.

```
{
    "subscriptionId": "12345",
    "data": [
        {
            "id": "Room1",
            "type": "Room",
            "temperature": {
                "value": 23,
                "type": "Number",
                "metadata": {}
            },
            "humidity": {
                "value": 70,
                "type": "percentage",
                "metadata": {}
            }
        },
        {
            "id": "Room2",
            "type": "Room",
            "temperature": {
                "value": 24,
                "type": "Number",
                "metadata": {}
            }
        }
    ]
}
```

```
{
    "subscriptionId": "12345",
    "data": [
        {
            "id": "Room1",
            "type": "Room",
            "temperature": 23,
            "humidity": 70
        },
        {
            "id": "Room2",
            "type": "Room",
            "temperature": 24
        }
    ]
}

{
    "subscriptionId": "12345",
    "data": [ [23, 70], [24] ]
}
```

Por fim, é possível customizar as notificações HTTP recebidas a partir da propriedade `notification.httpCustom` de uma subscrição, permitindo personalizar os campos `url`, `headers`, `qs` e `payload`. É possível configurar os campos utilizando a sintaxe `${..}`, como pode ser visto abaixo.

```
"httpCustom": {
    "url": "http://foo.com/entity/${id}",
    "headers": {
        "Content-Type": "text/plain"
    },
    "method": "PUT",
    "qs": {
        "type": "${type}"
    },
    "payload": "The temperature is ${temperature} degrees"
}
```

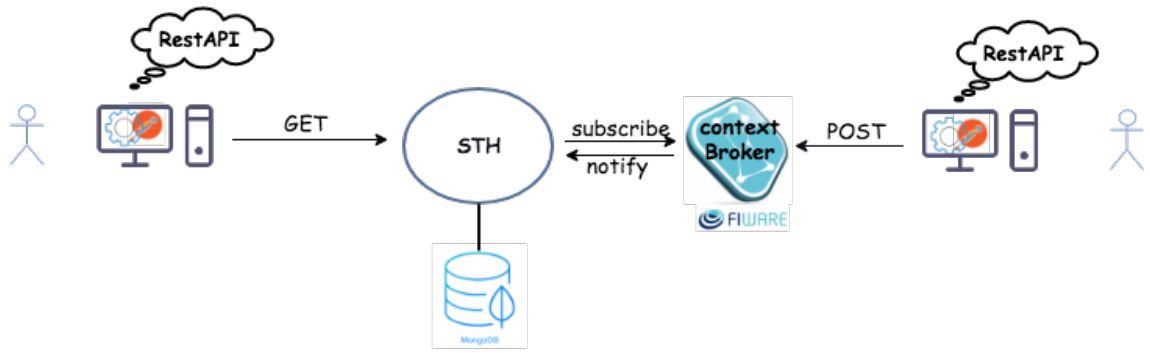
Em que `${id}` e `${type}` são substituídos pelas entidades `id` e `type`, respectivamente, e qualquer outro `${token}` (neste exemplo, `${temperature}` é substituído pelo valor do atributo que tem o nome `temperature`).

Dessa forma, em uma aplicação dependente de contexto, o funcionamento básico do Orion Context Broker é o de consultar no Consumidor de Contexto as informações fornecidas pelo Produtor de Contexto. O Orion é utilizado para criar as entidades de contexto, o Produtor de Contexto atualiza as entidades e atributos da aplicação específica, e o Consumidor de Contexto consulta as informações de contexto de interesse, tendo a possibilidade de subscrever-se às mudanças de contexto para ser notificado assincronamente a cada vez que uma informação de interesse seja modificada. Essas capacidades são permitidas através das requisições descritas anteriormente.

3.2 Persistência das Informações

Para que as informações de contexto armazenadas no Orion Context Broker sejam persistidas, é necessário que elas sejam colocadas em um banco de dados. Para este propósito, podem ser utilizados GEs, como o STH-Comet ou o Cygnus. Segundo (FIWARE FOUNDATION, 2019k), o STH-Comet permite gerar "uma série temporal de informações de contexto não-processadas e agregadas" juntamente com o banco de dados MongoDB. Uma simplificação da arquitetura pode ser vista na Figura 3.

Figura 3 – Arquitetura de uma aplicação que utiliza o STH-Comet

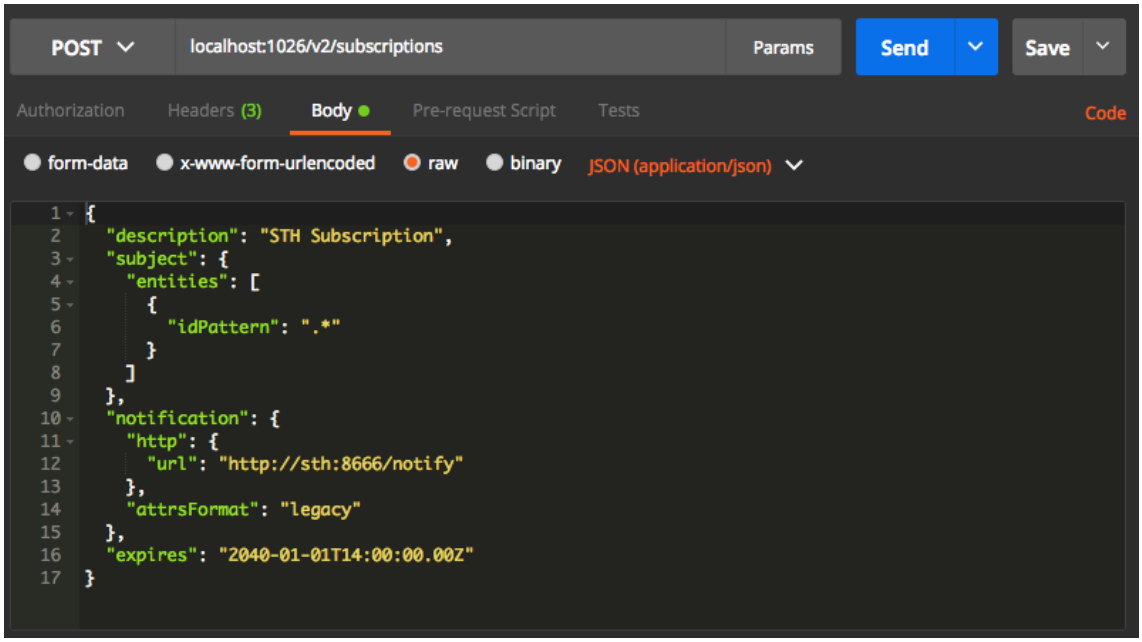


Fonte: (FIWARE FOUNDATION, 2019k)

O funcionamento básico dessa arquitetura resume-se ao usuário fazendo requisições HTTP REST normalmente ao Orion Context Broker, como foi explicado anteriormente, mas agora com a adição do STH como um Consumidor de Contexto a partir da criação de uma subscrição com a operação NGSI `<host-do-orion>:<porta-do-orion>/v2/subscriptions`, passando também o URL no qual o STH estará escutando para receber notificações de alterações

nas informações de contexto, as quais serão repassadas para o banco de dados MongoDB. Um exemplo da criação dessa subscrição pode ser visto na Figura 4.

Figura 4 – Criação de uma subscrição do Orion ao STH-Comet

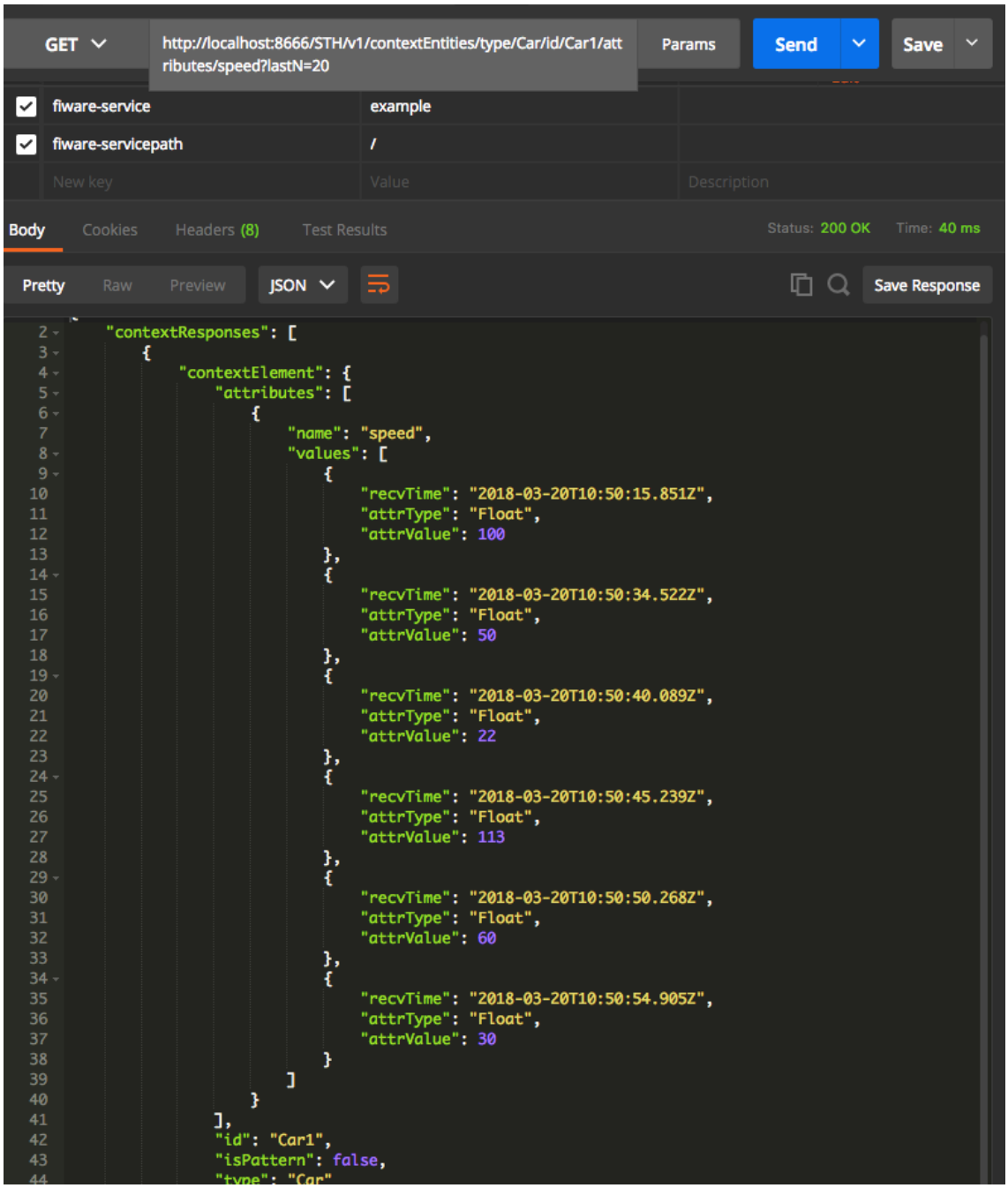


Fonte: (FIWARE FOUNDATION, 2019k)

É possível consultar as informações de contexto históricas não-processadas diretamente do STH-Comet com uma operação NGSI do tipo `<host-do-STH>:<porta-do-STH>/STH/v1/contextEntities/type/<entityType>/id/<entityId>/attributes/<attrName>?<parametros>`, cuja resposta conterá o nome do atributo modificado e os diferentes valores que ele assumiu ao longo do tempo. Os parâmetros possíveis nas consultas são `lastN` (especifica o número de entradas), `hLimit` (em caso de paginação, especifica o número de entradas por página), `hOffset` (em caso de paginação, especifica o *offset* da consulta), `dateFrom` (especifica a data de início), `dateTo` (especifica a data final), `filetype` (especifica um tipo de arquivo no qual retornar os dados) e `count` (se for true, a resposta conterá a quantidade de elementos). Na Figura 5 é possível ver o exemplo de uma requisição que retorna os últimos 20 valores assumidos pelo atributo `speed` da entidade `Car1`.

As informações de contexto históricas agregadas podem ser consultadas com uma requisição `<host-do-STH>:<porta-do-STH>/STH/v1/contextEntities/type/<entityType>/id/<entityId>/attributes/<attrName>?<parametros>`, em que os parâmetros podem incluir os valores `aggrMethod` (`max`, `min`, `sum` ou `sum2` para atributos numéricos e `occur` para valores alfanuméricos), `aggrPeriod` (período ou resolução da agregação), `dateFrom` (data de início da agregação) e `dateTo` (data final da agregação). Esse tipo de consulta diferencia-se da simples verificação das alterações pontuais nas informações de contexto, pois permite a composição de dados estatísticos acerca das entidades ou atributos de interesse.

Figura 5 – Informações de contexto do atributo `speed` da entidade `Car1`



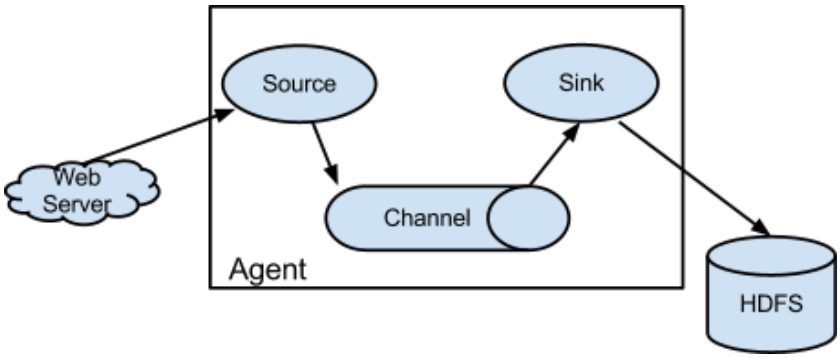
Fonte: (FIWARE FOUNDATION, 2019k)

Também é possível consultar todos os dados históricos das entidades registradas diretamente a partir do MongoDB ao executar o SGBD e verificar as coleções geradas.

Uma alternativa ao STH-Comet para persistência de informações de contexto advindas do Orion Context Broker é o GE Cygnus. Diferentemente do STH-Comet, o Cygnus não permite a consulta de uma série temporal, mas apenas das modificações em um dado momento do tempo. A documentação do FIWARE (APACHE SOFTWARE FOUNDATION, 2019) descreve

o Cygnus como sendo um conector entre o Orion e vários tipos de armazenamentos do FIWARE (como CKAN, Hadoop, com o sistema de arquivos HDFS, e o próprio STH Comet) ou outros tipos de armazenamento externos (como o MySQL), baseado na tecnologia Apache Flume, que lida com a execução e movimentação de grandes quantidades de dados, e com a persistência dos agentes. Um agente é composto de uma fonte que recebe os dados (por exemplo, de um servidor Web), um canal para o qual a fonte envia os dados depois de transformá-los em eventos do tipo Flume e um *sink*, responsável por persistir os dados no armazenamento externo. A arquitetura de um evento Flume pode ser visualizada na Figura 6.

Figura 6 – Arquitura de um evento Flume



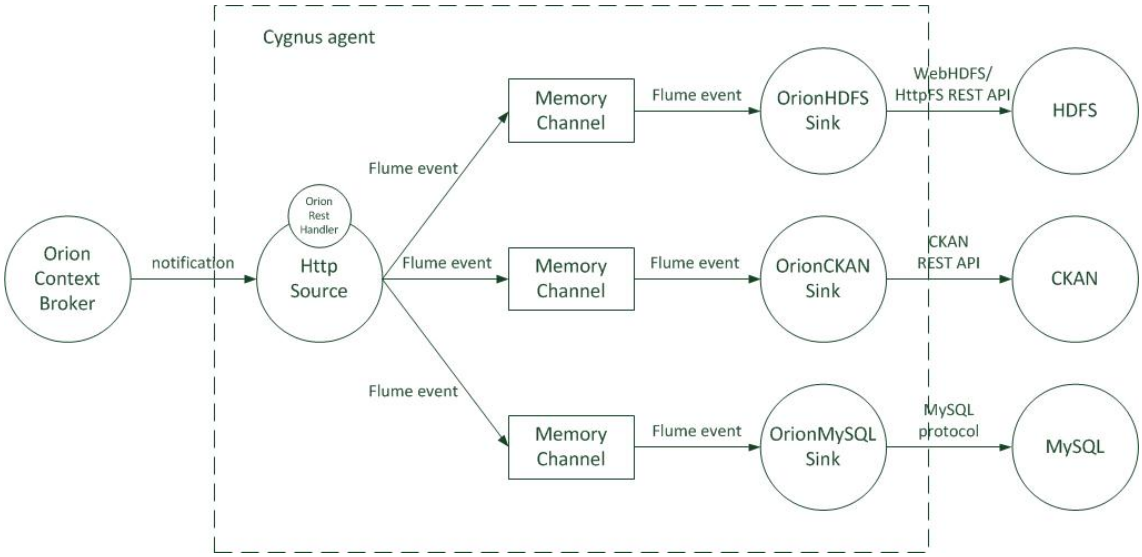
Fonte: (APACHE SOFTWARE FOUNDATION, 2019)

A arquitetura básica de um agente Cygnus adota o esquema de um evento Flume como apresentado acima. Para cada tipo de armazenamento externo (por exemplo, MySQL), deve existir um fluxo específico para que as fontes possam receber as notificações de alterações nas informações de contexto. Assim, deve existir um objeto `HttpSource` que liga o Orion a cada um dos canais, um canal e um *sink* para cada tipo de armazenamento. A Figura 7 mostra o esquema de arquitetura básico para a utilização de armazenamentos HDFS, CKAN e MySQL. Também é possível utilizar os armazenamentos MongoDB, STH-Comet, PostgreSQL, CartoDB, DynamoDB, Kafka e TwitterHDFS.

Também existem arquiteturas avançadas do Cygnus, em que o desempenho do componente é melhorado. As arquiteturas incluem a de múltiplos sinks e um único canal, múltiplos sinks e múltiplos canais e uma arquitetura de *high availability*.

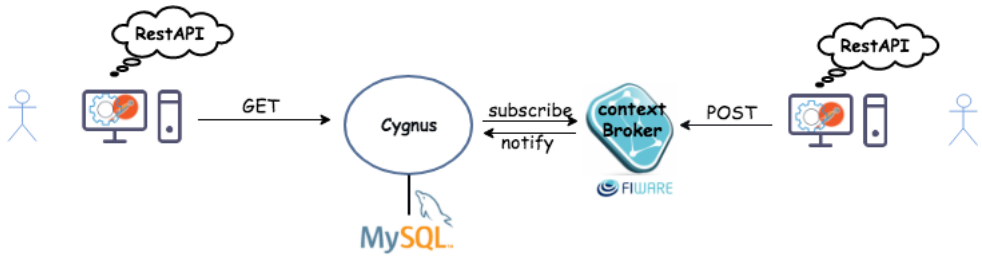
O funcionamento do Cygnus utilizando MySQL resume-se a configurar os hosts e portas referentes ao Orion, Cygnus e MySQL (além das informações do banco de dados e das informações de sink e canal), criar uma subscrição para ouvir as mudanças de informações de contexto no Orion realizando a requisição POST para `<host-do-orion>:<porta-do-orion>/v2/subscriptions`, passando a URL na qual o Cygnus estará escutando, e, por fim, verificar as modificações nas entidades diretamente no banco de dados. A arquitetura dessa configuração pode ser visualizada na Figura 8.

Figura 7 – Arquitetura básica do Cygnus utilizando armazenamentos HDFS, CKAN e MySQL



Fonte: (FIWARE FOUNDATION, 2019b)

Figura 8 – Arquitetura de uma aplicação que utiliza o Cygnus com MySQL



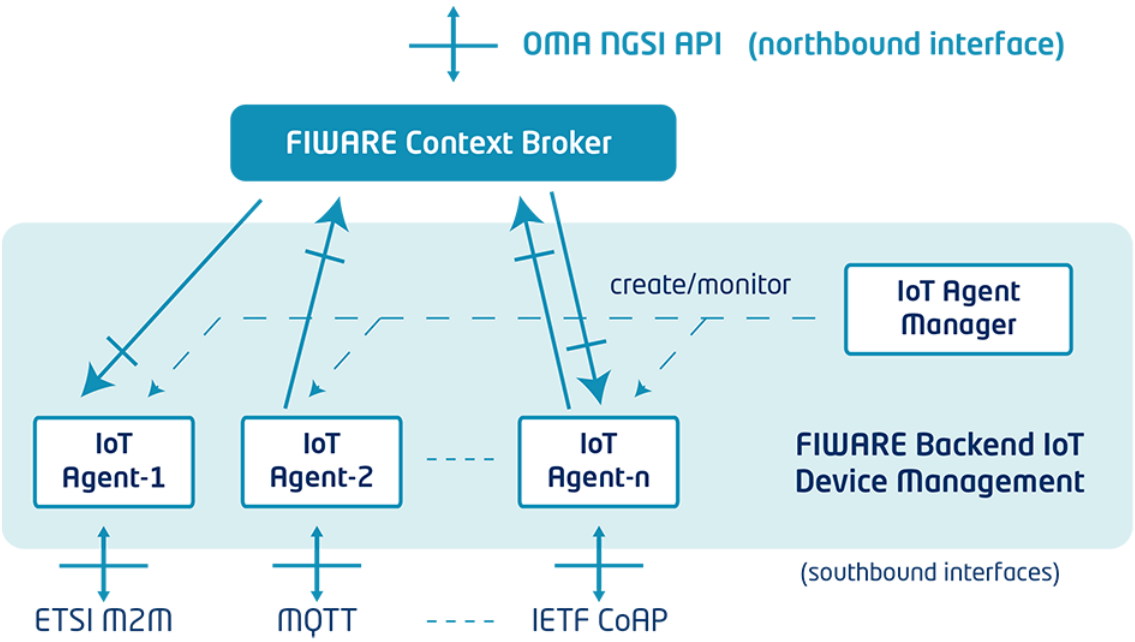
Fonte: (FIWARE FOUNDATION, 2019l)

3.3 IoT com FIWARE

Um dos propósitos do surgimento do FIWARE é facilitar a implementação de aplicações de Internet das Coisas. Esse tipo de aplicação encaixa-se perfeitamente no modelo de entidades de contexto implementado pela API do FIWARE, uma vez que é composta de objetos (coisas) inteligentes e cada um deles possui informações sobre seu estado atual (informações de contexto). No entanto, a heterogeneidade de coisas inteligentes causa uma das maiores dificuldades na IoT: conectar diversos dispositivos (sensores e atuadores) diferentes, com tecnologias e protocolos diferentes a cada camada de comunicação. Para lidar com essa complexidade, o GE IDAS funciona como um canal de comunicação entre os dispositivos inteligentes e o Orion Context Broker. A arquitetura básica de uma aplicação usando o IDAS pode ser vista na Figura 9.

O Orion permanece sendo o local onde as entidades e atributos referentes aos dispositivos inteligentes estarão armazenadas. Além disso, o IDAS possui diferentes implementações depen-

Figura 9 – Arquitetura básica de uma aplicação que utiliza o IDAS para conectar dispositivos inteligentes ao Orion Context Broker



Fonte: (FIWARE FOUNDATION, 2019m)

dendo do protocolo utilizado pelo dispositivo inteligente, conhecidas como IoT Agents. O Orion utiliza requisições NGSI, cada IoT Agent fornece uma interface para as requisições advindas do Orion e uma interface para as requisições utilizando os protocolos nativos dos dispositivos. Os IoT Agents implementados atualmente incluem o suporte aos protocolos HTTP/MQTT (com corpo em JSON), Lightweight M2M, HTTP/MQTT (com corpo em UltraLight2.0) e LoRaWAN.

Para que uma aplicação básica possa tirar proveito dos IoT Agents, é necessário configurar os *hosts* e portas do Orion e do IoT Agent específico para o protocolo do dispositivo. Cada dispositivo será registrado como uma entidade de contexto no Orion com ID `<device-id>` e todas as requisições devem conter os cabeçalhos `fiware-service` e `fiware-servicepath` para diferenciar partes de uma aplicação, uma vez que os IDs dos dispositivos não têm garantia de serem únicos devido à grande quantidade de dispositivos que podem existir. Para iniciar, é necessário definir um grupo de serviços para os dispositivos realizando uma requisição POST para `<host-dos-iot-agents>:<porta-da-interface-entre-iot-agents-e-orion>/iot/services` com corpo contendo a entidade `services` com atributos `apikey` (referente a segurança e autenticação), `cbroker` (com a URL em que o Orion está respondendo), `entity_type` (opcional, contendo o tipo de entidade que está sendo criada) e `resource` (informando o ponto no qual os serviços serão executados no IoT Agent). Em seguida, o IoT Agent deve ser informado dos dispositivos inteligentes a partir de uma requisição POST para `<host-dos-iot-agents>:<porta-da-interface-entre-iot-agents-e-orion>/iot/devices`.

Para um sensor, o corpo da requisição contém a entidade `devices` com atributos `device_id` (ID do sensor no IoT Agent), `entity_name`, `entity_type` (nome e tipo da entidade no Orion referentes ao sensor), `timezone`, `protocol`, `attributes` (atributos da entidade de contexto no Orion referentes ao estado do sensor, ou seja, de suas medições, com id, nome e tipo), `lazy` e `static_attributes` (dados estáticos sobre o sensor, como relacionamentos).

Para um atuador, o corpo da requisição contém a entidade devices com atributos `device_id` (ID do atuador no IoT Agent), `entity_name`, `entity_type` (nome e tipo da entidade no Orion referentes ao atuador), `timezone`, `protocol`, `transport` (protocolo de comunicação), `endpoint` (local para onde o IoT Agent deve enviar os comandos no protocolo nativo do dispositivo), `commands` (atributos da entidade de contexto no Orion referentes aos comandos do atuador, com nome e tipo), `lazy` e `static_attributes` (dados estáticos sobre o atuador, como relacionamentos).

Para um dispositivo que é ao mesmo tempo sensor e atuador, o corpo da requisição pode conter todas a propriedades listadas anteriormente.

Após o fornecimento dos dispositivos inteligentes ao IoT Agent conectado ao Orion Context Broker, este último GE já fica ciente de quais atributos correspondem a informações de medições dos sensores ou comandos dos atuadores.

Para enviar uma medição de um sensor, basta enviar uma requisição POST para `<host-do-iot-agent>:<porta-da-interface-entre-dispositivo-e-iot-agent>/iot/` com o corpo da requisição contendo o id do atributo e o valor que ele deve assumir. Para verificar que a medição foi recebida com sucesso, basta realizar uma requisição GET para `<host-do-orion>:<porta-do-orion>/v2/entities/<nome-da-entidade>` e perceber que a resposta conterá a entidade e o atributo terá o valor passado na requisição POST feita anteriormente.

Para enviar um comando para um atuador, é possível testar que o comando pode ser enviado fazendo uma requisição POST diretamente ao IoT Agent para `<host-dos-iot-agents>:<porta-da-interface-entre-iot-agents-e-orion>/v1/updateContext` (que é o ponto que será chamado eventualmente pelo Orion no evento de um comando ser enviado), com o corpo contendo os `contextElements` (neste caso, com o id do dispositivo no Orion, atributos correspondentes ao comando e outras propriedades opcionais) e `updateAction` (neste caso, `UPDATE`), cuja resposta deve ser uma mensagem de sucesso da atualização do contexto. Para ler o resultado do envio do comando, basta fazer uma requisição GET para `<host-do-orion>:<porta-do-orion>/v2/entities/<nome-da-entidade>` e verificar que as informações de contexto corresponderão ao que se espera do dispositivo ao receber o comando enviado.

Também é possível registrar os comandos dos dispositivos inteligentes no Orion, para que o dispositivo cadastrado no IoT Agent funcione como um fornecedor de contexto externo e assim se possa enviar um desses comandos com uma simples requisição PATCH para o Orion. Para isso, basta enviar uma requisição POST para `<host-do-orion>:<porta-do-orion>/v2/registra`

`tions` com o corpo contendo description (uma descrição do comando), entities (com id e tipo da entidade do Orion correspondente ao dispositivo), attrs (com os valores que o atributo alterado pelo comando pode assumir) e provider (contendo o URL `<host-dos-iot-agents>:<porta-da-interface-entre-iot-agents-e-orion>/v1`, que é o ponto no qual as atualizações de contexto foram cadastradas). Depois disso, é possível enviar uma requisição PATCH para `<host-do-orion>:<porta-do-orion>/v2/entities/<nome-da-entidade>/attrs` com o corpo contendo o nome do comando.

Em (TERROSO-SAENZ et al., 2019) o FIWARE é utilizado para desenvolver uma plataforma de IoT para gerenciar e analisar dados de energia, chamada IoTEP (*IoT Energy Plataform*). O uso do FIWARE se deu devido a sua capacidade de exibir dados de forma homogênea, possibilitado pela compatibilidade com o padrão NGSI. A plataforma é desenvolvida utilizando um modelo de entidades de contexto e sua arquitetura é dividida em quatro camadas: sensoriamento, homogeneização e armazenamento, suporte analítico, e serviço. A camada de sensoriamento utiliza o componente IoT Agents do FIWARE para conectar os dispositivos e mapear os dados. A camada de homogeneização e armazenamento utiliza os componentes Orion Context Broker e STH Comet para receber os dados seguindo o modelo de informações de contexto e armazená-los, criando séries históricas que servem para a análise estatística. Nessa camada, também é utilizado o componente Perseo para definir e executar regras de processamento de eventos complexos para remover *outliers* dos dados recebidos. A camada de suporte analítico possibilita a mineração dos dados com o auxílio de um detector de volatilidade de dados (que verifica o comportamento anormal de algum dos dados e realiza uma verificação a partir das séries históricas disponíveis no STH Comet) e um gerador de entidades virtuais (que diminui a sobrecarga gerada por informações redundantes ainda preservando informações relevantes, utilizando ferramentas disponíveis no STH Comet). A camada de serviço funciona como local de contato entre a plataforma e o usuário, permitindo a visualização dos resultados das tarefas das camadas anteriores. A plataforma implementada foi utilizada para prever o consumo de energia de três prédios na Universidade de Murcia, na Espanha, e mostrou-se um sucesso, confirmando a importância da integração e análise de dados em uma aplicação de Internet das Coisas, demonstrando também a eficiência dos componentes do FIWARE.

Em (ZAMORA-IZQUIERDO et al., 2019) é proposto o desenvolvimento de uma plataforma inteligente para automatização de agricultura de precisão, utilizando sensores e atuadores em uma arquitetura de três camadas. A primeira camada é chamada de camada de corte e é onde os sensores e atuadores são implantados e conectados com nós *Cyber-Physical Systems* (CPS, sistemas usados para receber dados dos sensores e comunicar-se com os atuadores). A segunda camada é a de computação edge, na qual as principais tarefas são monitoradas e gerenciadas de modo que a confiabilidade da plataforma aumente, além de ser responsável por controlar a camada de corte a partir de módulos de controle que se comunicam com os dispositivos usando os protocolos MQTT e CoAP. A terceira camada é a de nuvem de dados e funciona como interface entre os usuários e a plataforma, utilizando-se de componentes do FIWARE e da interface NGSI

para realizar essa comunicação e o gerenciamento dos dados. O Orion Context Broker e o STH Comet são utilizados na camada de nuvem de dados para armazenar as informações referentes ao estado da estufa e as funções de subscrição a dados específicos do Orion são utilizadas para que certas mudanças disparem ações específicas, enquanto que o Comet mantém histórico de dados e fornece informações para criação de rotinas de análises de *Big Data*. Nessa mesma camada, o componente Cygnus é utilizado para salvar registros de dados no sistema de armazenamento Hadoop. O Orion, os IoT Agents do IDAS e o STH Comet são utilizados por cada um dos módulos de controle da camada de computação edge para recebimento e armazenamento dos dados dos sensores sem a necessidade de acesso aos dados da camada de nuvem e para atuação dos atuadores. O sistema foi testado em uma estufa real e o sucesso da plataforma foi nítido, mostrando bons resultados em economia de água e melhoria do cultivo a partir da agregação de tecnologias de Internet das Coisas à agricultura de precisão valendo-se dos componentes do FIWARE.

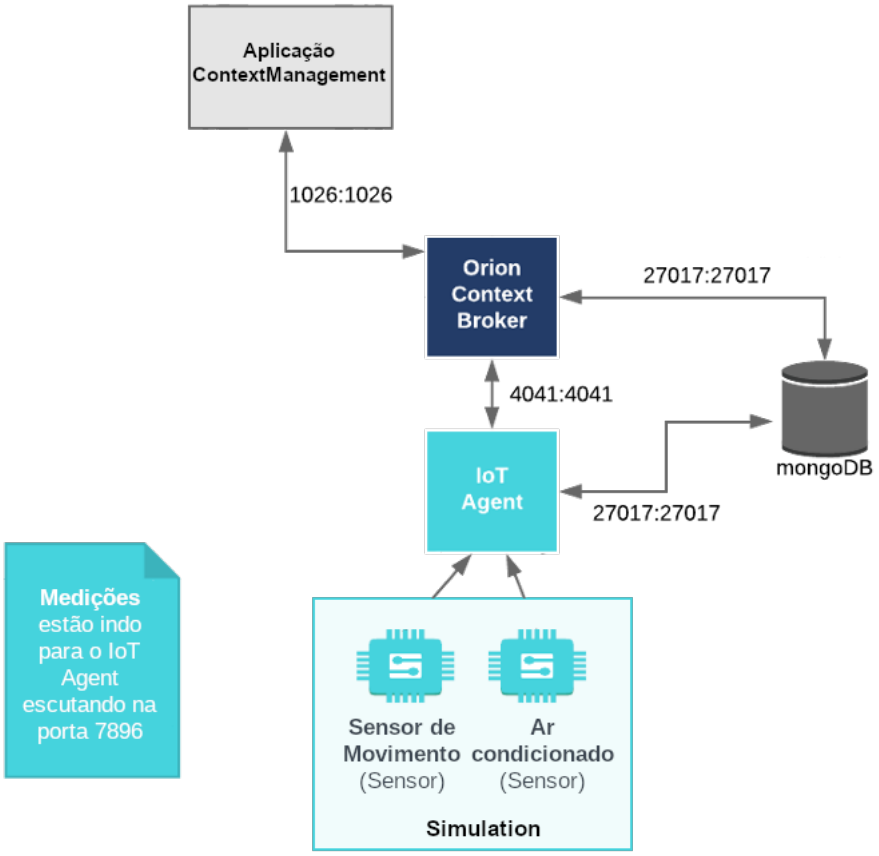
4
Aplicação

A implementação realizada neste trabalho tem como objetivo demonstrar o entendimento dos conteúdos estudados relativos a IoT e FIWARE, colocando-os em prática. Dessa forma, foi simulado o comportamento de uma sala de aula inteligente em um dia com três horários de aula, contando com sensores de movimento e um ar-condicionado inteligente que deve realizar certas ações dependendo do sensoriamento. O funcionamento dos componentes utlizados na aplicação baseia-se nos tutoriais disponibilizados em (FIWARE FOUNDATION, 2019e).

Os componentes do FIWARE utilizados na aplicação foram o Orion Context Broker e o IDAS, com o IoT Agent do protocolo UltraLight. O Orion foi utilizado para o registro das entidades e das informações de contexto relacionadas aos atributos da sala de de aula, do sensor de movimento e do ar-condicionado inteligente, assim como para a subscrição para as mudanças sensoriadas pelo sensor de movimento. O sensor e o ar-condicionado foram registrados no IoT Agent, considerando o uso simulado do protocolo UltraLight 2.0, que é um protocolo leve baseado em texto, apropriado para dispositivos com capacidades limitadas (FIWARE FOUNDATION, 2019h). Além disso, o banco de dados MongoDB foi utilizado para manter persistências das informações de contexto de ambos os componentes (porém, sem manutenção de histórico, apenas as últimas informações atualizadas). A arquitetura da aplicação pode ser visualizada na Figura 10. Ela implementa dois objetos, um, denominado `ContextManagement`, que implementa as regras de controle do ar-condicionado, e outro, denominado `Simulation`, que simula eventos para o sensor de presença e o ar-condicionado inteligente.

A utilização dos componentes do FIWARE foi possível com o auxílio da ferramenta Docker. Ela é uma ferramenta que permite o trabalho de desenvolvimento utilizando contêineres, isto é, unidades de software com os códigos e dependências necessários para a execução de cada um dos componentes, sem precisar virtualizar o hardware nem um sistema operacional separadamente para cada um deles (DOCKER INC., 2019b). Cada componente tem disponibilizada uma imagem de contêiner no Docker Hub (DOCKER INC., 2019a), e com o Docker

Figura 10 – Arquitetura da aplicação desenvolvida



Adaptado de: (FIWARE FOUNDATION, 2019i)

Compose torna-se possível definir e executar aplicações multi-contêiner, de forma que todos os componentes necessários são executados com a definição de um arquivo de configuração YAML[1] (formato de serialização de dados baseado em XML) e um comando no terminal. Os conteúdos do arquivo e o comando podem ser vistos abaixo.

```
version: "3.1"
services:
  mongo-db:
    image: mongo:3.6
    hostname: mongo-db
    container_name: db-mongo
    expose:
      - "27017"
    ports:
      - "27017:27017"
```

[1] http://www.rfc-editor.org/rfc/rfc2822.txt

```
11      networks:
12        - default
13      command: --bind_ip_all --smallfiles
14      volumes:
15        - mongo-db:/data
16
17    orion:
18      image: fiware/orion:2.1.0
19      hostname: orion
20      container_name: fiware-orion
21      depends_on:
22        - mongo-db
23      networks:
24        - default
25      expose:
26        - "1026"
27      ports:
28        - "1026:1026"
29      command: -dbhost mongo-db -logLevel DEBUG
30      healthcheck:
31        test: curl --fail -s http://localhost:1026/version || exit 1
32
33    iot-agent:
34      image: fiware/iotagent-ul:1.8.0
35      hostname: iot-agent
36      container_name: fiware-iot-agent
37      depends_on:
38        - mongo-db
39      networks:
40        - default
41      expose:
42        - "4041"
43        - "7896"
44      ports:
45        - "4041:4041"
46        - "7896:7896"
47      environment:
48        - "IOTA_CB_HOST=orion" # name of the context broker to update
49        context
50        - "IOTA_CB_PORT=1026" # port the context broker listens on to
51        update context
52        - "IOTA_NORTH_PORT=4041"
53        - "IOTA_REGISTRY_TYPE=mongodb" #Whether to hold IoT device info
54        in memory or in a database
55        - "IOTA_LOG_LEVEL=DEBUG" #The log level of the IoT Agent
56        - "IOTA_TIMESTAMP=true"
57        - "IOTA_MONGO_HOST=mongo-db" # The host name of ongoDB
58        - "IOTA_MONGO_PORT=27017" # The port mongoDB is listening on
```

```
      - "IOTA_MONGO_DB=iotagentul" # The name of the database used in
      mongoDB
      - "IOTA_HTTP_PORT=7896" # The port used for device traffic over
      HTTP
      - "IOTA_PROVIDER_URL=http://iot-agent:4041"
    healthcheck:
      test: curl --fail -s http://localhost:4041/iot/about || exit 1

networks:
  default:
    ipam:
      config:
        - subnet: 172.18.1.0/24

volumes:
  mongo-db:
```

Comando:

```
docker-compose --log-level ERROR -p fiware up -d --remove-orphans
```

A linguagem de programação Java foi utilizada para desenvolver o acumulador `ContextManagement`, responsável por receber notificações enviadas pela subscrição realizada às mudanças dos atributo das entidades registradas no Orion Context Broker. Adicionalmente, para cada tipo de entidade registrada no Orion foi implementada uma classe Java respectiva, de modo que fosse possível executar comandos NGSI utilizando objetos bem estruturados. As bibliotecas Google HTTP Client e Gson do Google foram utilizadas com o propósito de tornar transparente a transformação de objetos JSON para Java e vice-versa.

Na inicialização do `ContextManagement`, uma entidade sala com o ID `urn:ngsi-ld:Room:001` e o tipo `Room` é registrada com os atributos `name`, `maxCapacity`, `occupation` e `temperature` no Orion Context Broker. É possível ver abaixo o equivalente em formato JSON do corpo da requisição realizada.

```
{
    "id":"urn:ngsi-ld:Room:001",
    "type":"Room",
    "name":{
        "type":"Text", "value":"Auditório DCOMP"
    },
    "maxCapacity":{
        "type":"Integer", "value":50
```

```
    },
    "occupation":{
        "type":"Integer", "value":0
    },
    "temperature":{
        "type":"Float", "value":30.0
    }
}
```

Para o recebimento das medições do sensor de movimento e do ar-condicionado, é necessária a criação de um grupo de serviço para o IoT Agent realizar a autenticação (através de uma chave de API) e ter conhecimento da URL à qual o Orion está respondendo. Dessa forma, o IoT Agent sabe que os dispositivos enviarão mensagens para a porta definida no arquivo YAML (7896), na qual o IoT Agent estará escutando para o recebimento das medições. A criação do grupo de serviço também faz parte da inicialização do `ContextManagement`. O equivalente em formato JSON do corpo da requisição enviada ao IoT Agent em `localhost:4041/iot/services` pode ser visto abaixo.

```
{
    "services": [
        {
            "apikey":      "APIKEY",
            "cbroker":     "http://localhost:1026",
            "entity_type": "Thing",
            "resource":    "/iot/d"
        }
    ]
}
```

Dessa forma, os dispositivos poderão enviar medições autenticadas realizando requisições HTTP para a URL `iot-agent:7896/iot/d?i=<id>&k=<APIKEY>`, seguindo o padrão do protocolo UltraLight 2.0, em que `id` corresponde ao ID do dispositivo registrado no IoT Agent e `APIKEY` corresponde a uma chave aleatória gerada pela aplicação.

Após registrar no Orion a sala de aula e o grupo de serviço responsável pelo recebimento das medições dos sensores, o `ContextManagement` registra o sensor de movimento no IoT Agent. O corpo da requisição é composto por uma lista dos dispositivos que se deseja registrar, em que cada um deles contém um `device_id`, `entity_name`, `timezone`, `attributes` e `static_attributes`. O sensor de movimento tem como `entity_name` o nome da entidade a que ele será associado no Orion, como atributo um inteiro correspondente às presenças detectadas

na sala, e como atributo estático o relacionamento com a sala de aula. O equivalente ao corpo da requisição em formato JSON feito ao endereço `localhost:4041/iot/devices` pode ser visto abaixo.

```
{
    "devices": [
        {
            "device_id":   "motion001",
            "entity_name": "urn:ngsi-ld:Motion:001",
            "entity_type": "Motion",
            "timezone":    "America/Belem",
            "attributes": [
                { "object_id": "c", "name": "count", "type": "Integer" }
            ],
            "static_attributes": [
                { "name":"refRoom", "type": "Relationship", "value":
                "urn:ngsi-ld:Room:001"}
            ]
        }
    ]
}
```

Após isto, é possível enviar medições correspondentes ao sensoriamento do sensor de movimento diretamente para o IoT Agent, o qual será responsável por alterar as informações de contexto relacionadas ao dispositivo através de conversa com o Orion.

De maneira similar, o ar-condicionado inteligente é registrado no IoT Agent, com a diferença que contém atributos referentes ao estado do dispositivo, modo de funcionamento e temperatura. É possível ver abaixo o corpo da requisição feita ao endereço `localhost:4041/iot/devices`.

```
{
    "devices": [
        {
            "device_id": "ac001",
            "entity_name": "urn:ngsi-ld:AC:001",
            "entity_type": "AirConditioner",
            "timezone":    "America/Belem",
            "attributes": [
                {
```

```
                    "object_id": "t",
                    "name": "temperature",
                    "type": "Float"
                },
                {
                    "object_id": "s",
                    "name": "state",
                    "type": "Text"
                },
                {
                    "object_id": "m",
                    "name": "mode",
                    "type": "Text"
                }
            ],
            "static_attributes": [
                { "name":"refStore", "type": "Relationship","value":
                "urn:ngsi-ld:Room:001"}
            ]
        }
    ]
}
```

Para que o acumulador `ContextManagement` implementado fique ciente das alterações do atributo referente à presença de pessoas na sala, é necessário criar uma subscrição às alterações desse atributo na entidade mapeada ao sensor de movimento. Para isso, basta enviar uma requisição a `localhost:1026/v2/subscriptions` com o corpo abaixo.

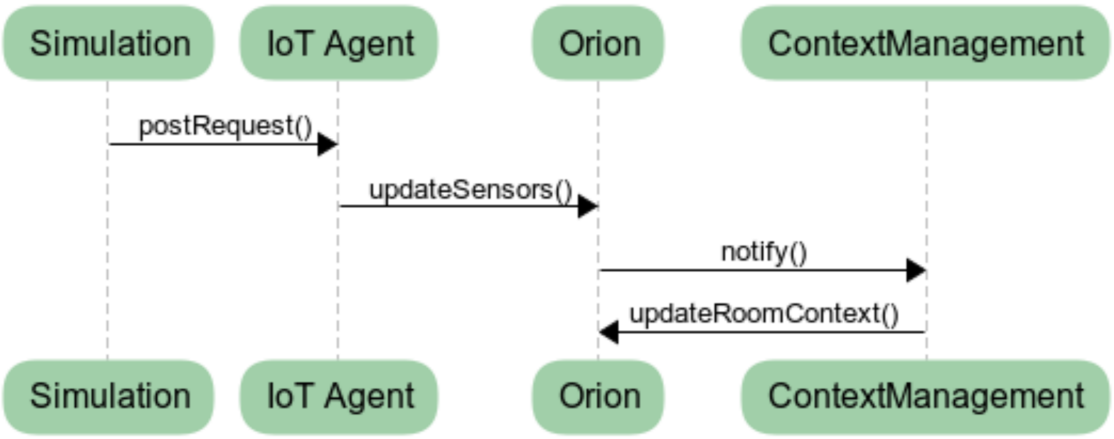
```
{
    "description": "Motion:001 Subscription",
    "subject": {
        "entities": [
            { "id": "urn:ngsi-ld:Motion:001",  "type": "Motion"  }
        ],
        "condition": {  "attrs": [ "count"  ] }
    },
    "notification": { "http": { "url": "<url-do-acumulador>" },
        "attrs": [ "count" ]
    },
    "expires": "2040-01-01T14:00:00.00Z",
```

```
    "throttling": 5
}
```

Dessa forma, o Orion Context Broker estará ciente que deve enviar uma notificação à URL passada em `<url-do-acumulador>` cada vez que o atributo `count` do sensor `urn:ngsi-ld:Motion:001` for alterado.

Todas as requisições citadas anterioremente são realizadas pelo objeto `ContextManagement`. Depois disso, ele fica aguardando por alterações dos sensores notificadas pelo Orion Context Broker. Por outro lado, o objeto `Simulation` é responsável por simular alterações aleatórias nos sensores através de mensagens UltaLight enviadas ao IoT Agent e este, por sua vez, é responsável por atualizá-las no Orion. A Figura 11 mostra a sequência de execução da aplicação implementada.

Figura 11 – Sequência de execução da aplicação



Sumarizando o comportamento da aplicação, quando uma aula é criada, um objeto `Class` define o número de alunos da turma, o tempo de duração real da aula e o ar-condicionado `ac001` é inicializado com o estado `off`, o modo `normal` e a temperatura desejada. Quando a aula é iniciada, o estado de `ac001` muda para `on` e os alunos começam a chegar. Quando os alunos começam a chegar, o modo do ar-condicionado é mudado para `turbo` e as medições de movimento são enviadas ao IoT Agent pelo sensor `motion001` a cada 5 segundos. O objeto `ContextManagement` fica ciente da chegada dos alunos e atualiza os atributos `occupation` e `temperature` da entidade `urn:ngsi-ld:Room:001` com as informações de ocupação e temperatura calculadas a partir das presenças detectadas e da temperatura configurada no ar-condicionado. A aula segue por um período ocioso em que nenhuma presença a mais é detectada pelo sensor, e a temperatura da sala continua diminuindo ou aumentando até que chegue próximo do valor desejado. Quando os alunos começam a sair, o modo do ar-condicionado é configurado como `normal` e os movimentos de saída são enviados ao IoT Agent a cada 5 segundos. O objeto `ContextManagement` atualiza o contexto à medida que as pessoas saem. Quando todos os alunos deixam a sala, o ar-condicionado

é desligado e `ContextManagement` é responsável por atualizar a temperatura da sala, que tende a voltar ao valor inicial caso uma aula não comece logo. Todos os valores (número de alunos, chegadas, partidas, temperatura e tempo) são definidos a partir do sorteio de inteiros e números reais utilizando a biblioteca `Thread` do Java.

Partes do trecho de código que implementa o comportamento dos sensores pode ser visto abaixo.

```
System.out.println(students + " alunos sao esperados");
System.out.println("Tempo real da aula: "+ realTime + " minutos");
runPostRequest("http://localhost:7896/iot/d?k=6NUB3eD0YERJml1btYssPOa1qY&i=ac001",
 ↪  "s|on");
System.out.println("Ar condicionado ligado");

arrivals();

idleTime(realTime);

if(occupation > 0)
    departures();

runPostRequest("http://localhost:7896/iot/d?k=6NUB3eD0YERJml1btYssPOa1qY&i=ac001",
 ↪  "s|off");
System.out.println("Ar condicionado desligado");

idleTime(duration);

System.out.println("Fim de aula\n");
```

E partes do trecho responsável pelo gerenciamento de contexto pode ser visto abaixo.

```
while(true) {
    CountNotification countNotification =
     ↪  gson.fromJson(context.accumulator.listen(), CountNotification.class);
    for(CountNotification.Data notificationData: countNotification.getData())
        context.motionDetected = notificationData.getCount().getValue();

    if(context.airConditioner.getState().equals("on")) {
        if(context.motionDetected != 0 &&
         ↪  context.airConditioner.getMode().equals("turbo")) {
            context.room.setOccupation(context.room.getOccupation().getValue() +
             ↪  context.motionDetected);
            context.room.setTemperature(context.room.getTemperature().getValue() +
             ↪  (double)context.motionDetected * 0.025);
        }
        else if(context.motionDetected != 0 &&
         ↪  context.airConditioner.getMode().equals("normal")) {
```

```
12              context.room.setOccupation(context.room.getOccupation().getValue() -
                ↪  context.motionDetected);
13          }
14
15          if(context.room.getTemperature().getValue() >
            ↪  context.airConditioner.getTemperature()) {
16              context.room.setTemperature(context.room.getTemperature().getValue() -
                ↪  1);
17          }
18          else {
19              context.room.setTemperature(context.room.getTemperature().getValue() +
                ↪  1);
20          }
21      }
22      else {
23          if(context.room.getTemperature().getValue() < initialTemperature) {
24              context.room.setTemperature(context.room.getTemperature().getValue() +
                ↪  1);
25          }
26          else {
27              context.room.setTemperature(context.room.getTemperature().getValue() -
                ↪  1);
28          }
29      }
30      System.out.println("Temperatura " + context.room.getTemperature().getValue());
31      System.out.println("Ocupacao " + context.room.getOccupation().getValue());
32  }
```

Embora a implementação tenho sido limitada devido a não estarem disponíveis sensores reais e de ter um comportamento relativamente simples (simulando sensores ideais e que atendem perfeitamente às necessidades imaginadas), a aplicação se mostrou um passo muito importante na confirmação do entendimento dos conceitos de IoT e da utilização dos componentes do FIWARE.

Os códigos completos da aplicação gerenciadora de contexto e do simulador, assim como o arquivo de configuração YAML, com as devidas instruções para executá-los encontram-se disponíveis para acesso no Gitlab[2].

[2] https://git.dcomp.ufs.br/felipematheuscs/TCC

5
Conclusão

A Internet das Coisas encontra-se em um estágio mais avançado do que já esteve em qualquer outro momento. Espera-se que a tecnologia evolua ainda mais no futuro próximo e que a computação torne-se algo ubíquo e praticamente invisível. A necessidade do estudo da IoT e de Cidades Inteligentes mostrou-se muito justificada ao longo deste trabalho de modo a tornar claro a origem dessas tecnologias e trazer entendimento sobre os desafios enfrentados em suas implantações em ambientes modernos.

O entendimento dos conceitos de IoT e de Cidades Inteligentes também permitiu visualizar esse tipo de aplicação como sendo essencialmente composta de entidades que possuem informações de contexto que necessitam de atualização. Isso, por sua vez, encontra grande conveniência na proposta da plataforma FIWARE. O estudo do FIWARE e de seus componentes demonstrou-se muito útil para a compreensão de como a implementação de aplicações inteligentes com necessidades de gerenciamento e armazenamento de informações de contexto pode tornar-se mais simples. Esse estudo também demonstrou como já existem grandes avanços nos esforços de superar os desafios impostos pela heterogeneidade de hardware e software de dispositivos de aplicações IoT.

Ter disponíveis todas as informações sobre IoT, Cidades Inteligentes e o FIWARE também permitiu visualizar como esboçar uma solução para uma aplicação simples de controle de climatização em uma sala de aula. A implementação dessa aplicação demonstrou e poderá ser útil ao funcionar como modelo para uma possível implantação real. Essa implantação poderia tornar-se ponto de partida para um aumento na "inteligência"do ambiente universitário, permitindo encontrar outros casos de uso para a utilização de sensores, atuadores e dispositivos inteligentes para melhorar a eficiência da universidade. Dessa forma, os esforços tecnológicos e científicos de implementar aplicações de IoT contribuiriam ainda mais para o desenvolvimento dessas áreas.

5.1 Limitações

Embora tenha sido útil para compreender os desafios e chegar à implementação de uma aplicação simples, o estudo bibliográfico de Internet das Coisas e Cidades Inteligentes mostrou-se abrangente demais para este trabalho de conclusão de curso, tornando-se possivelmente superficial.

As implementações de aplicações inteligentes com o auxílio da plataforma FIWARE eram em número muito reduzido durante a maior parte da realização deste trabalho e ainda são muito limitadas, o que se mostrou um desafio para o estudo bibliográfico. Além disso, as informações sobre essa plataforma encontram-se nas documentações e tutoriais disponibilizados pela FIWARE Foundation. A maior parte dessa documentação demorou para ser atualizada e os tutoriais não existiam durante grande parte do tempo em que o trabalho foi realizado. Felizmente, esse problema foi praticamente sanado nos últimos meses.

Para este trabalho, não havia a disponibilidade de utilização de sensores e atuadores físicos para utilização em condições reais. Por conta disso, foi necessário depender de uma simulação. Por conta disso, os dispositivos simulados na aplicação de teste implementada podem não refletir fielmente aqueles que seriam utilizados em uma implantação real, uma vez que foi necessário abstrair muitos dos detalhes dos dispositivos reais, além de os valores das medições e ajustes terem sido gerados de maneira aleatória.

5.2 Trabalhos Futuros

A plataforma FIWARE apresenta diversos componentes, como aqueles voltados à persistência das informações, análise estatística dos dados históricos, etc. Neste trabalho, foram utilizados apenas os componentes de gerenciamento de informações de contexto, Orion Context Broker, e de conexão de dispositivos inteligente, IDAS (IoT Agents). Um trabalho futuro poderia incrementar as funcionalidades da aplicação implementada para tirar proveito de componentes como o STH Comet e Cygnus. Além disso, um estudo aprofundado de outros componentes deixados de fora deste trabalho, como o Cosmos, possibilitaria uma análise de Big Data.

A implementação da aplicação foi realizada utilizando bibliotecas externas para tentar tornar a programação o mais simples possível, porém, apenas para as poucas funcionalidades utilizadas (atualização de contexto, notificações, criação e atualização de dispositivos). Um trabalho futuro poderia abstrair todas as operações necessárias para a utilização de cada um dos componentes do FIWARE a partir da criação de um *framework* de programação para facilitar a tarefa de programar uma aplicação inteligente. Adicionalmente, poderia ser realizado o incremento de funcionalidades e a validação da aplicação implementada utilizando dispositivos reais.

Referências

Ahlgren, B.; Hidell, M.; Ngai, E. C. . Internet of things for smart cities: Interoperability and open data. *IEEE Internet Computing*, v. 20, n. 6, p. 52–56, Nov 2016. ISSN 1089-7801. Citado 2 vezes nas páginas 11 e 17.

ALAVI, A. H. et al. Internet of things-enabled smart cities: State-of-the-art and future trends. *Measurement*, v. 129, p. 589 – 606, 2018. ISSN 0263-2241. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0263224118306912>. Citado 2 vezes nas páginas 17 e 18.

ALDRIDGE. *IoT in the Classroom: How Traditional Education is Changing*. 2016. Disponível em: <https://aldridge.com/future-iot-in-the-classroom-education/>. Citado na página 11.

APACHE SOFTWARE FOUNDATION. *Apache Flume*. 2019. Disponível em: <http://flume.apache.org/>. Citado 2 vezes nas páginas 34 e 35.

ASHTON, K. That 'internet of things' thing. *RFiD Journal*, jun. 2009. Citado na página 13.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer Networks*, v. 54, n. 15, p. 2787 – 2805, 2010. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128610001568>. Citado na página 14.

BORMANN, C. *Constrained Application Protocol*. 2016. Disponível em: <http://coap.technology/>. Citado na página 16.

BRAUN, T. et al. Security and privacy challenges in smart cities. *Sustainable Cities and Society*, v. 39, p. 499 – 507, 2018. ISSN 2210-6707. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2210670717310272>. Citado na página 17.

BUTLER, H. et al. *The GeoJSON Format*. 2015. Disponível em: <https://tools.ietf.org/html/draft-butler-geojson-06>. Citado na página 26.

Derhamy, H. et al. A survey of commercial frameworks for the internet of things. In: *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*. [S.l.: s.n.], 2015. p. 1–8. ISSN 1946-0740. Citado na página 15.

DOCKER INC. *Docker Hub*. 2019. Disponível em: <https://hub.docker.com/u/fiware>. Citado na página 41.

DOCKER INC. *Enterprise Application Container Platform | Docker*. 2019. Disponível em: <https://www.docker.com/>. Citado na página 41.

DUNKELS, A.; VASSEUR, J. *IP for Smart Objects, Internet Protocol for Smart Objects (IPSO) Alliance*. 2009. Citado na página 14.

ELMAGHRABY, A. S.; LOSAVIO, M. M. Cyber security challenges in smart cities: Safety, security and privacy. *Journal of Advanced Research*, v. 5, n. 4, p. 491 – 497, 2014. ISSN 2090-1232. Cyber Security. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2090123214000290>. Citado na página 17.

EUROPEAN COMMISSION. *The Future Internet platform FIWARE*. 2018. Disponível em: <https://ec.europa.eu/digital-single-market/en/future-internet-public-private-partnership>. Citado na página 20.

FIWARE FOUNDATION. *After the Open Day: From the FI-PPP to the FIWARE Foundation*. 2017. Disponível em: <https://www.fiware.org/2017/03/09/after-the-open-day-from-the-fi-ppp-to-the-fiware-foundation/>. Citado na página 20.

FIWARE FOUNDATION. *FIWARE NGSI APIv2 Walkthrough*. 2018. Disponível em: <https://fiware-orion.readthedocs.io/en/master/user/walkthrough_apiv2/index.html>. Citado 2 vezes nas páginas 25 e 29.

FIWARE FOUNDATION. *FIWARE-NGSI v2 Specification*. 2018. Disponível em: <http://fiware.github.io/specifications/ngsiv2/stable/>. Citado na página 22.

FIWARE FOUNDATION. *About Us*. 2019. Disponível em: <https://www.fiware.org/about-us/>. Citado na página 20.

FIWARE FOUNDATION. *Cygnus*. 2019. Disponível em: <https://fiware-cygnus.readthedocs.io/en/latest/>. Citado na página 36.

FIWARE FOUNDATION. *Developers*. 2019. Disponível em: <https://www.fiware.org/developers/>. Citado 2 vezes nas páginas 12 e 20.

FIWARE FOUNDATION. *Developers Catalogue*. 2019. Disponível em: <https://www.fiware.org/developers/catalogue/>. Citado 2 vezes nas páginas 12 e 21.

FIWARE FOUNDATION. *FIWARE - Github*. 2019. Disponível em: <https://github.com/FIWARE>. Citado na página 41.

FIWARE FOUNDATION. *FIWARE 104: Registering Context Providers*. 2019. Disponível em: <https://github.com/Fiware/tutorials.Context-Providers>. Citado na página 29.

FIWARE FOUNDATION. *FIWARE 106: Subscribing to Changes in Context*. 2019. Disponível em: <https://github.com/Fiware/tutorials.Subscriptions>. Citado na página 29.

FIWARE FOUNDATION. *FIWARE 201: Introduction to IoT Sensors*. 2019. Disponível em: <https://github.com/FIWARE/tutorials.IoT-Sensors>. Citado na página 41.

FIWARE FOUNDATION. *FIWARE 202: Provisioning an IoT Agent*. 2019. Disponível em: <https://github.com/FIWARE/tutorials.IoT-Agent>. Citado na página 42.

FIWARE FOUNDATION. *Foundation*. 2019. Disponível em: <https://www.fiware.org/foundation/>. Citado na página 20.

FIWARE FOUNDATION. *How to generate Short-term History*. 2019. Disponível em: <https://fiwaretourguide.readthedocs.io/en/latest/core/sth-comet/how-to-generate-the-history-of-Context-Information-using-STH-Comet/>. Citado 3 vezes nas páginas 32, 33 e 34.

FIWARE FOUNDATION. *How to store data in MySQL using Cygnus*. 2019. Disponível em: <https://fiwaretourguide.readthedocs.io/en/latest/core/cygnus/introduction/>. Citado na página 36.

FIWARE FOUNDATION. *IDAS Introduction*. 2019. Disponível em: <https://fiwaretourguide.readthedocs.io/en/latest/iot-agents/introduction/>. Citado na página 37.

GALÁN, F. *FIWARE NGSI: Managing Context Information at Large Scale*. 2016. Disponível em: <https://pt.slideshare.net/FI-WARE/fiware-ngsi-managing-context-information-at-large-scale>. Citado na página 25.

GARTNER. *Top Trends in the Gartner Hype Cycle for Emerging Technologies, 2017*. 2017. Disponível em: <https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/>. Citado 2 vezes nas páginas 11 e 18.

HUI, J.; CULLER, D.; CHAKRABARTI, S. *6LoWPAN: Incorporating IEEE 802.15.4 Into the IP Architecture – Internet Protocol for Smart Objects (IPSO) Alliance*. 2009. Citado na página 14.

INFSO D.4 Networked Enterprise RFID INFSO G.2 Micro Nanosystems in Co-operation with the Working Group RFID of the ETP EPoSS: Internet of Things in 2020. may 2008. Citado 2 vezes nas páginas 11 e 14.

KUMMITHA, R. K. R.; CRUTZEN, N. How do we understand smart cities? an evolutionary perspective. *Cities*, v. 67, p. 43 – 52, 2017. ISSN 0264-2751. Disponível em: <http://www.sciencedirect.com/science/article/pii/S026427511630378X>. Citado na página 17.

MACWRIGHT, T. *More than you ever wanted to know about GeoJSON*. 2015. Disponível em: <https://macwright.org/2015/03/23/geojson-second-bite.html#multi-geometries>. Citado na página 26.

MDN WEB DOCS. *An overview of HTTP*. 2019. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>. Citado na página 16.

MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, v. 10, n. 7, p. 1497 – 1516, 2012. ISSN 1570-8705. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1570870512000674>. Citado 3 vezes nas páginas 11, 14 e 15.

MQTT. *MQTT*. 2019. Disponível em: <http://mqtt.org/>. Citado na página 16.

OPEN MOBILE ALLIANCE LTD. *NGSI Context Management*. 2012. Disponível em: <http://www.openmobilealliance.org/release/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf>. Citado na página 22.

PRESSER, M.; GLUHAK, A. *The Internet of Things: Connecting the Real World with the Digital World*. 2009. Disponível em: <http://www.eurescom.eu/message>. Citado na página 14.

RAHMAN, L.; OZCELEBI, T.; LUKKIEN, J. Choosing your iot programming framework: Architectural aspects. In: . [S.l.: s.n.], 2016. p. 293–300. Citado 3 vezes nas páginas 15, 16 e 17.

ROBERT, J. et al. Open IoT ecosystem for enhanced interoperability in smart cities - Example of Métropole de Lyon. *Sensors*, MDPI, v. 17, n. 12, p. 2849, 2017. Disponível em: <https://hal.archives-ouvertes.fr/hal-01662220>. Citado na página 17.

TELEFÓNICA I+D. *Orion Context Broker NGSI API v1 Specification*. 2018. Disponível em: <http://telefonicaid.github.io/fiware-orion/api/v1/>. Citado 3 vezes nas páginas 22, 23 e 24.

TERROSO-SAENZ, F. et al. An open iot platform for the management and analysis of energy data. *Future Generation Computer Systems*, v. 92, p. 1066 – 1079, 2019. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X17304181>. Citado 2 vezes nas páginas 18 e 39.

TOMA, I.; SIMPERL, E.; HENCH, G. *A joint roadmap for semantic technologies and the internet of things*. 2009. Citado na página 14.

WEISER. *The Computer of the 21st century*. 1991. Disponível em: <http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html>. Citado na página 13.

ZAMORA-IZQUIERDO, M. A. et al. Smart farming iot platform based on edge and cloud computing. *Biosystems Engineering*, v. 177, p. 4 – 17, 2019. ISSN 1537-5110. Intelligent Systems for Environmental Applications. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1537511018301211>. Citado na página 39.