UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

MARCELO MAZZOCHI HILLMAN

MIDDLEWARE PARA CONECTIVIDADE E INTEROPERABILIDADE EM ECOSSISTEMAS IOT

CAXIAS DO SUL
2018

MARCELO MAZZOCHI HILLMAN

MIDDLEWARE PARA CONECTIVIDADE E INTEROPERABILIDADE EM ECOSSISTEMAS IOT

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Tecnologias Digitais na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientadora: Profa. Dra. Elisa Boff

CAXIAS DO SUL

2018

MARCELO MAZZOCHI HILLMAN

MIDDLEWARE PARA CONECTIVIDADE E INTEROPERABILIDADE EM ECOSSISTEMAS IOT

Trabalho de conclusão apresentado como requisito parcial para obtenção do grau de Bacharel em Tecnologias Digitais pela Universidade de Caxias do Sul, Área do Conhecimento de Ciências Exatas e Engenharias.

Aprovado em 03/12/2018

Banca Examinadora

_____

Profa. Dra. Elisa Boff

Universidade de Caxias do Sul – UCS

_____

Profa. Dra. Carine Geltrudes Webber

Universidade de Caxias do Sul – UCS

_____

Profa. Dra. Maria de Fatima Webber do Prado Lima

Universidade de Caxias do Sul – UCS

Dedico este trabalho a toda a minha família, em especial a minha esposa, Paula Samanta Muller da Rocha, que me apoia em todos os momentos, e a meus pais, pelo suporte, apoio e carinho.

AGRADECIMENTOS

Agradeço a minha orientadora, Profa. Dra. Elisa Boff, pelo apoio e incentivo à realização deste trabalho.

Aos meus colegas de aula e aos familiares, que me deram forças durante esta trajetória.

A todos que acreditam no meu desenvolvimento pessoal e profissional de alguma forma, impulsionando-me em meus estudos e carreira.

“Existem muitas hipóteses na ciência que estão erradas. Isso é perfeitamente aceitável, elas são a abertura para achar as que estão certas.”

Carl Sagan

RESUMO

A IoT (*Internet of Things* – em português, Internet das Coisas) é um novo paradigma tecnológico que propõe revolucionar a computação. Há estudos oriundos de diversas áreas, como computação ubíqua, computação pervasiva e inteligência de ambientes, que suportam os avanços no caminho em busca a ampliar este paradigma. Nesse cenário, a comunicação entre *hardware* e *software* precisa ser mediada, devido aos mais diferentes tipos de padrões e aplicabilidade, e para isso utilizam-se *middlewares*, cuja função é justamente fornecer essa ponte de comunicação. O presente trabalho apresenta um estudo dos conceitos, definições e arquiteturas relacionados à IoT, computação ubíqua, pervasiva e inteligência de ambientes; em seguida, desenvolve-se o estudo de caso sobre um *middleware* implantado para fornecer conexão a um dispositivo, este que irá enviar imagens e dados inerentes ao ambiente de uma sala de aula, disponibilizar estas informações e reportar a outras aplicações para que possam realizar suas tarefas, como reconhecimento facial que retorna a identificação dos usuários presentes nas imagens.

Palavras-chave: Internet das Coisas. Ambientes Inteligentes. Computação Ubíqua. Computação Pervasiva. Middleware.

ABSTRACT

The IoT (Internet of Things) is a new technological paradigm that revolutionizes the way communication is viewed. There are studies in several areas such as ubiquituos computing, pervasive and ambient intelligence that support advances in the way to expand this ecosystems. With this scenario we have a communication between hardware and software to be mediated, due to the different types of standards and aplications, and for this is used a middleware that was thought to provide this bridge of communication. In this work, a study of concepts, definitions and architectures related to the IoT, ubiquitous computing, pervasive and ambient intelligence is developed to work in a use case of a implanted middleware to provide the connection of a device, wich will send images and data inherent to the classroom environment, to make available and report other applications so they can perform their tasks as face recognition for call record or frequency data for air conditioning systems.

Keywords: Internet of Things. Ambient Intelligence. Ubiquitous Computing. Pervasive Computing. Middleware.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CASAGRAS	Coordination and Support Action for Global RFID-related Activities and. Standardisation
DB	Database
IP	Internet Protocol
IPSO	Internet Protocol for Smart Objects
ISTAG	Information Society Technologies Advisory Group
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
JSON	JavaScript Object Notation
LAN	Local Area Network
M2M	Machine-to-Machine
SQL	Structured Query Language
PAN	Private Area Network
PWM	Pulse Width Modulation
RAM	Random Access Memory
REST	Representational State Transfer
RFID	Radio-Frequency IDentification
TCP	Transmission Control Protocol
TICs	Tecnologias da Informação e Comunicação
UPnP	Universal Plug and Play
UWB	Ultrawideband
vCPU	Virtual CPU
VM	Virtual Machine
WAN	Wide Area Network

SUMÁRIO

1 INTRODUÇÃO

Internet of Things (IoT) é um conceito presente nas mais diversas áreas do conhecimento tecnológico. Em geral, é empregado para designar um grupo de tecnologias que integram o real ao virtual. Nesse contexto, os *softwares* são projetados para executar em componentes de pequena escala da maneira mais eficiente possível, de forma que forneçam as funcionalidades necessárias para apoiar o funcionamento de um ecossistema de dispositivos inteligentes em seus objetivos.

Nessa rede de dispositivos e *softwares* com aplicabilidades diferentes que têm de se comunicar entre si para solucionar os mais diversos problemas, torna-se inviável a definição de um padrão de comunicação e organização para todos. Assim, faz-se necessário o desenvolvimento de uma camada de aplicação *middleware* que integre os diversos *gadgets* e *softwares,* a fim de facilitar o desenvolvimento das diversas soluções que permeiam o assunto. Oasis (2014, p. 3) descreve IoT como “Sistemas onde a Internet está conectada ao mundo físico via sensores ubíquos”.

Quando é tratado de IoT aplicada a tornar ambientes inteligentes, inserimo-nos na questão da *Ambient Intelligence* (AmI) – que traduzido, aqui, como *Inteligência do Ambiente* – que se refere à utilização de tecnologia e conectividade para agregar funcionalidades a determinado ambiente, de modo que os dispositivos passem a tomar decisões e beneficiar os usuários com base em históricos e informações coletadas em tempo real. AmI está diretamente alinhada à ideia do computador “desaparecendo” enquanto objeto isolado e tornando-se onipresente na vida humana cotidiana, situação que é designada pelo termo *computação ubíqua*. Tornar ambientes intel1)igentes, portanto, é aumentar o conforto, o rendimento e a precisão nos afazeres diários e técnicos, por meio da aproximação entre usuários e máquinas, criando relações entre tecnologias e conceitos acadêmicos.

1.1 PROBLEMA DE PESQUISA

O estudo da IoT aplicada ao desenvolvimento de AmI é muito importante para a otimização de processos em todos os setores culturais e socioeconômicos. Por isso, encontrar formas de conexões entre os mais diversos tipos de sensores e aplicações, de forma que busquem se tornar interoperáveis e independentes, é imprescindível para a confecção de ambientes inteligentes por meio de sistemas IoT.

1.2 QUESTÃO DE PESQUISA

O que é necessário para um ambiente digital fornecer interoperabilidade e independência das “coisas” em um ecossistema IoT?

1.3 OBJETIVO

Implementar uma camada de *middleware* que possa prover o ambiente digital de conectividade e interoperabilidade a ecossistemas IoT, a fim de tornar ambientes físicos inteligentes, conectando dispositivos que identifiquem e capturem imagem de alunos e uma aplicação de reconhecimento facial que retorne a identificação do aluno na imagem.

1.3.1 Objetivos específicos

Para atingir o objetivo geral mencionado, foram definidos os seguintes objetivos específicos, a fim de que sejam subprodutos deste trabalho:

a) Selecionar tecnologias para o desenvolvimento de aplicações voltadas para IoT e AmI;

b) Implementar uma camada de *middleware*, utilizando o caso de uso de reconhecimento facial de alunos em uma sala de aula;

c) Avaliar as soluções empreendidas.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado como segue. O Capítulo 2 apresenta o estado da arte sobre a área de Internet das Coisas, levando o foco para a inteligência dos ambientes. O Capítulo 3 aborda as tecnologias utilizadas como *middleware* para resolver a complexidade de interoperabilidade em sistemas distribuídos. O Capítulo 4 detalha a implementação desenvolvida para o *middleware* proposto neste trabalho e a validação da proposta por meio de um caso de aplicação de reconhecimento facial em ambientes de aprendizagem. Por fim, são apresentadas as considerações finais,

destacando as contribuições desta pesquisa, bem como as possibilidades de trabalhos futuros.

2 INTERNET DAS COISAS

Kevin Ashton foi o primeiro especialista a utilizar o termo *Internet of Things*, em uma palestra da Procter & Gamble no ano de 1999, além de apresentar uma de suas principais definições: um sistema capaz de conectar o real ao virtual, criando um mundo mais inteligente. Segundo Ashton (2009 apud DIAS, 2016, p. 15), “a IoT tem o potencial de mudar o mundo, assim como a Internet fez. Talvez até mais”. Ele ainda considera que, com a IoT, “os objetos – as ‘coisas’ – estarão conectados entre si e em rede, de modo inteligente, e passarão a sentir o mundo ao redor e a interagir”, e que

> Hoje em dia, os computadores e a Internet são quase que completamente dependentes dos seres humanos para obter informação. Se tivéssemos computadores que conhecessem tudo o que existe para se saber sobre as coisas reais - usando dados que eles mesmos agrupem, sem nossa ajuda, nós poderíamos, por exemplo, acompanhar tudo, o que reduziria imensamente o desperdício, perdas e custos (ASHTON, 2014, p. 6).

A International Telecommunication Union (ITU) publicou, em 2005, seu primeiro *Technical Report* sobre o assunto, numa abordagem abrangente, sugerindo que a IoT poderia conectar os objetos do mundo, tanto de forma sensorial quanto inteligente, por meio de combinações de tecnologias. Além disso, identificou alguns desafios para a exploração em potencial da IoT, como padronizações, espectro de frequência, privacidade e questões sociais e éticas (DIAS, 2016).

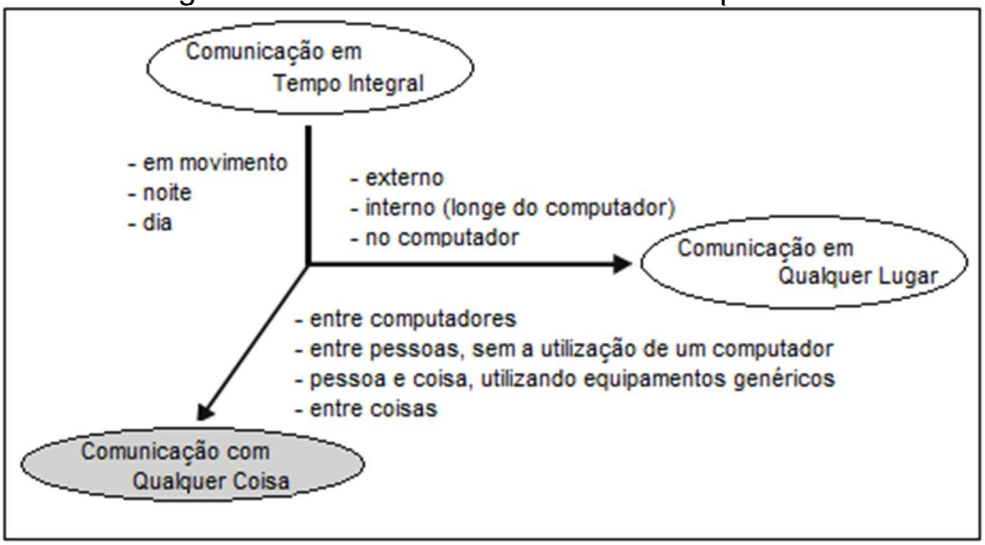
A ITU Telecommunication Standardization Sector (ITU-T) (2012), setor de normatização da ITU, define IoT, na recomendação ITU-T Y.2060, como uma infraestrutura global para a sociedade da informação, permitindo serviços avançados por meio da interligação das coisas (físicas e virtuais). Estas coisas são independentes uma das outras, mas baseadas na interoperabilidade das tecnologias de informação e comunicação e em evolução por meio da exploração de capacidades de identificação, captura de dados, processamento e comunicação.

As coisas físicas são as existentes em nosso mundo não virtual, que podem ser percebidas, atuadas e conectadas, como sensores de temperatura, acionadores e demais equipamentos eletrônicos físicos. Enquanto as virtuais existem no mundo da informação e podem ser armazenadas, processadas e acessadas, como conteúdos multimídia e *softwares*. O termo *device* (traduzido como *dispositivo*) relacionado à IoT, designa os equipamentos com capacidades obrigatórias de comunicação e opcionais de sensoriamento, atuação e processamento de dados. O termo *thing (*traduzido como

coisa) refere-se a um objeto do mundo físico que é capaz de ser identificado e integrado em redes de comunicação.

A publicação da *ITU-T* ainda trata de como a IoT é capaz de criar uma nova dimensão nas Tecnologias da Informação e Comunicação (TICs) (vide Figura 1). Resultante dos avanços dessas tecnologias que nos possibilitam a comunicação em movimento a todo momento e em qualquer lugar, a chamada “Comunicação com qualquer coisa” diz respeito às interações entre dispositivos, coisas e pessoas.

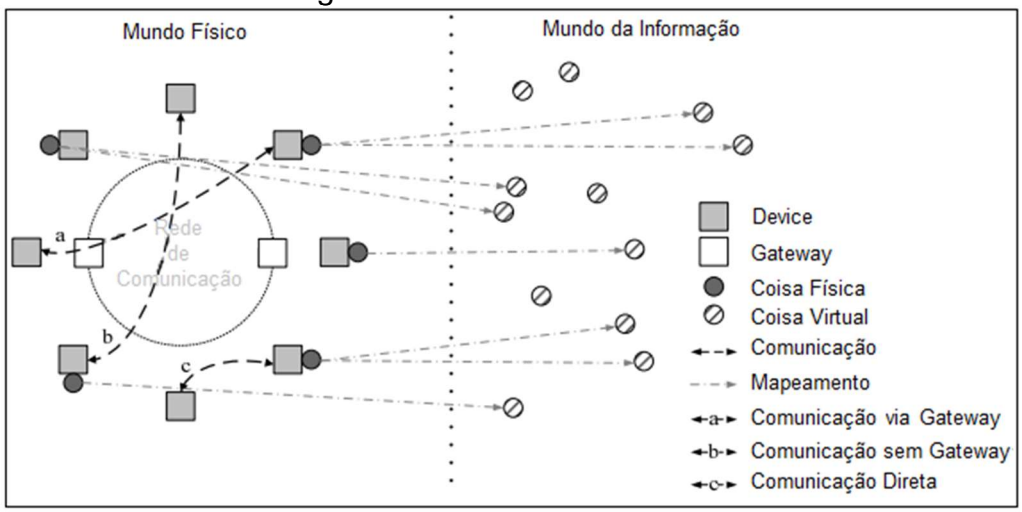
Figura 1 - A nova dimensão introduzida pela IoT



Fonte: Adaptado de ITU-T (2012).

A arquitetura de soluções IoT proposta pela recomendação da ITU-T é ilustrada pela Figura 2, que exibe o processo de representação de coisas físicas no meio digital por meio do mapeamento. Este trata de criar uma versão virtual de objetos reais, isto é, armazenar em uma base de dados suas informações inerentes. Os objetos virtuais, contudo, nem sempre são relacionados diretamente aos físicos; eles também podem existir de forma independente.

Figura 2 – Visão técnica da IoT



Fonte: Adaptado de ITU-T (2012).

Nessa situação, os dispositivos são aqueles que coletam e fornecem informações à rede para o processamento adicional, podendo, em alguns casos, efetuar operações com base em informações recebidas da rede. Eles trocam as informações entre si de algumas formas, e podem até mesmo utilizar combinações. No caso *a* um dos dispositivos utiliza um *gateway* (um nó de rede para interfacear diferentes protocolos) para a conexão à rede de comunicação. No caso *b*, os dispositivos conseguem realizar a comunicação diretamente na rede, sem a mediação de um *gateway*. O caso *c*, por sua vez, não apresenta utilização da rede de comunicação. Embora a Figura 2 demonstre apenas comunicação entre o meio físico, esse evento também ocorre no mundo da informação, isto é, entre as coisas virtuais.

O projeto Coordination and Support Action for Global Radio-Frequency IDentification (RFID) – related Activities and Standardisation (CASAGRAS), da *European Framework 7,* define a IoT como

> uma infraestrutura de rede global, interligando objetos físicos e virtuais por meio da exploração de captura de dados e capacidades de comunicação. Essa infraestrutura inclui a Internet existente e em evolução, bem como os desenvolvimentos de rede. Ela oferecerá identificação de objetos especifica, capacidade de sensoriamento e de conexão como base para desenvolvimento de aplicações e serviços independentes cooperativos. Estes serão caracterizados por um elevado grau de captura autônoma de dados, transferência de eventos, conectividade e interoperabilidade de rede (CASAGRAS, 2009, p. 10).

Wootton (2016) define IoT, de forma simplificada, como uma série de dispositivos com inteligência embarcada que estão prontas e interessadas em trocar

dados. Esses dispositivos são munidos de sensores e/ou atuadores, existindo algum tipo de conectividade. Ele também trata *internet* como forma de comportamento e não característica de comunicação física, isto é, os dispositivos se comunicam entre si como internet e não somente através da internet. Podem-se utilizar, também, diversos tipos de protocolos em uma Private Area Network (PAN), traduzida como rede de área privada. Estes protocolos – bluetooth, zigbee e Ultrawideband (UWB) – empregados a um sistema IoT são utilizados para a troca de informação entre os próprios dispositivos em uma comunicação Machine-to-Machine (M2M), termo utilizado para comunicação entre dispositivos. Estes mesmos dispositivos utilizam um de um *gateway* para disponibilizar a conexão com uma rede Wide Area Network (WAN).

O autor apresenta como exemplo a situação de um fabricante que tenha de medir o consumo de energia de uma residência ou estabelecimento. Esse fabricante instala sensores de medição em tempo real e, através da comunicação com um *gateway* de internet, os dados obtidos são enviados a um servidor na nuvem. O cliente, por meio de um aplicativo (*mobile, desktop* ou *web*), tem acesso às informações tratadas em gráficos e relatórios nos mais diversos moldes, de acordo com sua necessidade. Muitos fabricantes vendem isso como uma solução IoT, mas Wootton (2016) discorda de que a IoT efetivamente se enquadre em casos como esse. Se, porém, agregarmos a essa solução a capacidade de analisar os dados e detectar alterações no perfil de consumo, ou um funcionamento fora dos padrões, de um eletrodoméstico (como uma bomba de piscina que deixa de funcionar, por exemplo), e o sistema notificar essa avaria a seu usuário, então teremos um contexto em que as coisas se comunicam com a nuvem. Esta, por sua vez, aprende sobre e analisa seu usuário, estando pronta para algum tipo de mensagem ou ação através de dispositivos ou *softwares* e caracterizando um contexto em que a IoT se efetiva.

A definição do termo se torna ainda mais difusa se for considerado a quantidade de temas e áreas de trabalho em que podemos envolvê-lo. López (2010), professor da Universidade de Cardiff, relata, em seu *blog*, acreditar ser mais fácil enumerar o que a IoT não é do que tudo aquilo que ela pode ser. Ela não é apenas computação ubíqua ou pervasiva. Não se trata necessariamente de um protocolo de comunicação Transmission Control Protocol/Internet Protocol (TCP/IP). Também não são etiquetas RFID incorporadas a objetos como quando o termo foi cunhado pelo Ashton, ou aplicações como frascos de remédios que emitem avisos.

Atzori, Iera e Morabito (2010) a definem como uma presença difusa em torno de todos nós, através de uma variedade de objetos que utilizam de RFID, *tags*, sensores, atuadores, *smartphones,* que, por meio de endereçamentos, são capazes de interagir uns com os outros e cooperar para atingir objetivos comuns. Os estudiosos destacam que há múltiplas definições encontradas para a expressão na comunidade de pesquisa e que a razão aparente dessa imprecisão seria o próprio termo *internet das coisas* que sintetiza dois tópicos. O primeiro se volta para o campo de redes; o segundo, para objetos a serem integrados, e isso cria questões desafiadoras, como a semântica (a significação da comunicação entre os objetos). Os três aspectos ilustrados na Figura 3 são considerados os paradigmas da IoT.

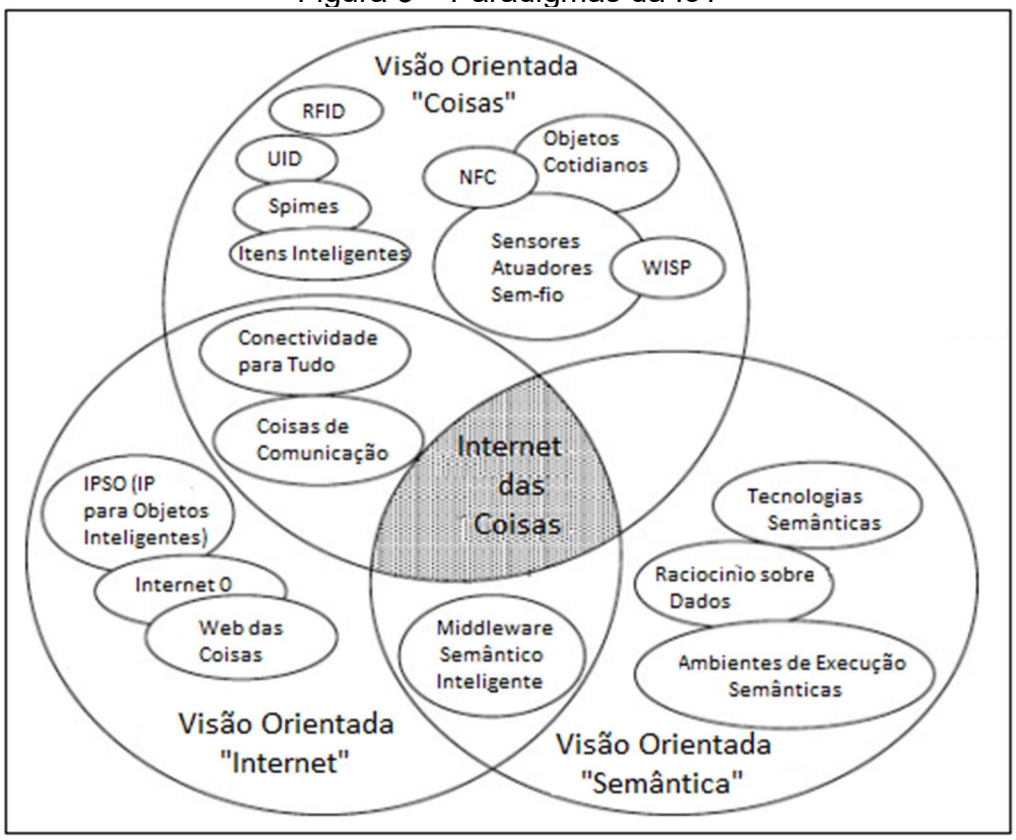
O primeiro se trata da visão orientada às coisas, pois ela está associada à comunicação e identificação de objetos no ambiente. Esses objetos podem ser desde itens que utilizam de RFID para sua identificação e rastreabilidade, também chamado de *spimes*, até dispositivos que são capazes de capturar dados e interagir com o ambiente e o usuário.

A definição de CASAGRAS (2009) vem ao encontro da visão orientada à internet, que define como os dispositivos e objetos são conectados entre si. Nessa perspectiva, é abordado temas como padronização de protocolos (como, por exemplo, o Internet Protocol for Smart Objects (IPSO), que trata da utilização do protocolo TCP/IP para objetos inteligentes) e configurações dos dispositivos e aplicações para operem de forma independente e interoperável.

As informações geradas por esses dispositivos precisam ser aplicadas, a fim de se obter um contexto e aplicabilidade. A visão orientada à semântica trata de aplicações que trabalharão com esses dados em suas soluções, necessitando, portanto, de ferramentas e algoritmos para interpretação de uma quantidade de dados em alta escala.

Na intersecção entre os conceitos de semântica e conectividade entra o *middleware*, que conecta os dispositivos e gerência os fluxos de informação que as aplicações irão receber para efetuarem suas operações.

Figura 3 – Paradigmas da IoT

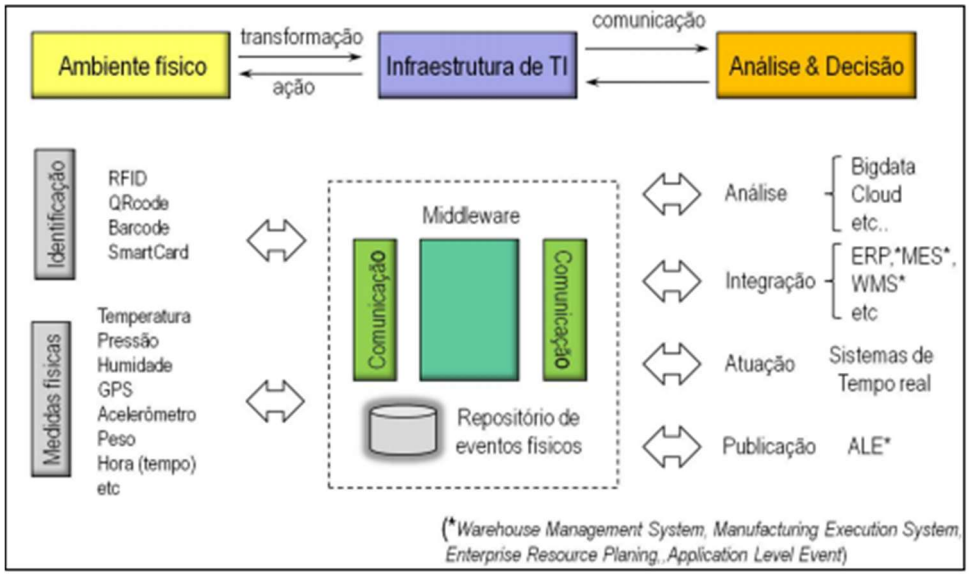


Fonte: Adaptado de Atzori. Iera e Morabito (2010).

As soluções IoT são compostas por dispositivos heterogêneos, ou seja, com diferentes características computacionais e de comunicação. Tendo em vista que eles interagem com o usuário, com o ambiente e entre si, surge a questão da interoperabilidade. A melhor forma de tratar heterogeneidade e interoperabilidade é pela definição de padrões, protocolos e *middlewares*. E, considerando a relevância atual do tema desta pesquisa, *middlewares* e um modelo arquitetural se tornam questões de discussão extremamente importantes, para que se elucidem os principais desafios no desenvolvimento e aplicação de soluções IoT.

Conforme pode-se visualizar na Figura 4, uma solução IoT integra um ambiente físico a um sistema de análise e decisão. Os objetos físicos são identificados ou têm suas informações convertidas para o meio digital e enviadas a um repositório, onde são armazenadas e distribuídas para as mais diversas aplicações poderem realizar tratamentos, análises e até mesmo atuações no meio físico através de atuadores (CARISSIMI, 2016, p. 12).

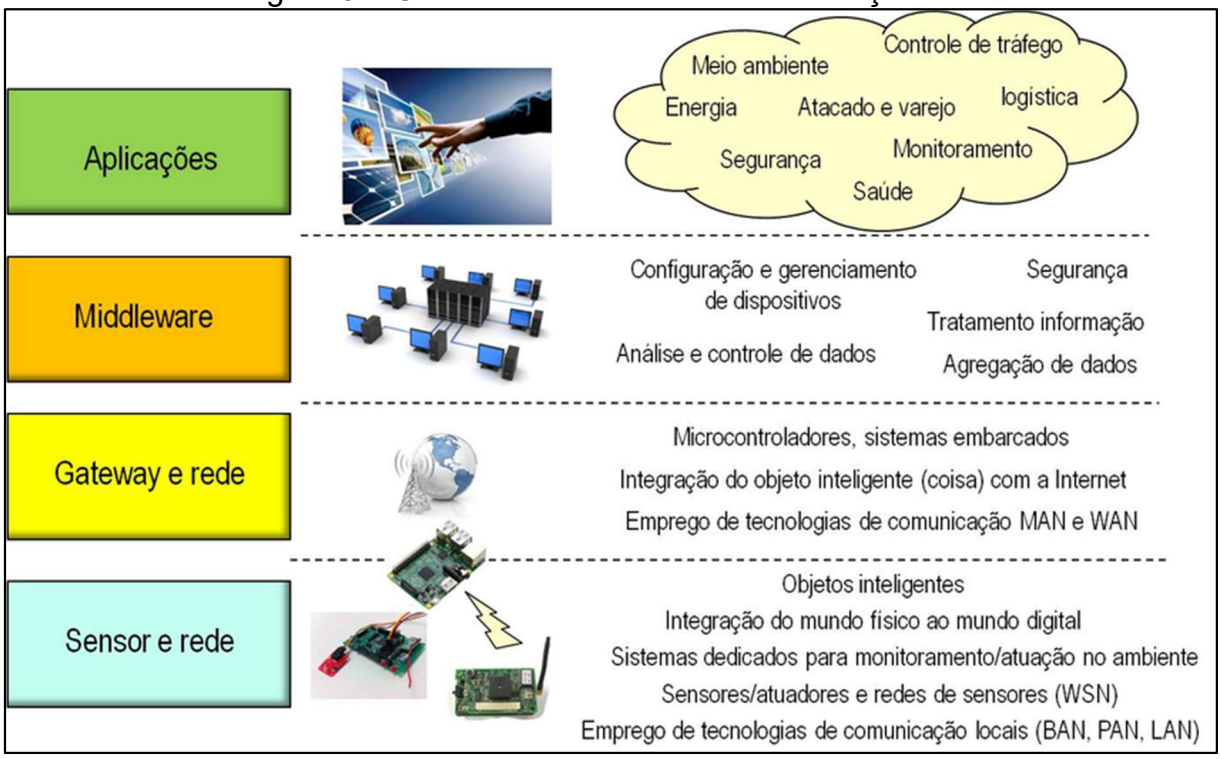
Figura 4 – Anatomia de uma solução IoT



Fonte: Carissimi (2016, p. 12).

Carissimi (2016) sugere organizar e analisar uma solução IoT a partir de um modelo de quatro camadas, conforme Figura 5. A camada de sensor e rede é a responsável por obter as informações de um objeto ou ambiente e enviá-las para um dispositivo externo. Ela se encarrega da leitura de sensores, realiza algum tratamento nos dados, quando necessário, e envia essas informações para outros sistemas através de redes Local Area Network (LAN) ou PAN. O sistema receptor dessas informações é o *gateway*, que as integra a uma infraestrutura maior através de redes WAN. É comum haver sistemas cujas camada de sensor e *gateway* acabem sendo mescladas por possibilidade do *hardware* existente. A terceira camada é constituída pelo *middleware*, que configura e gerencia os dispositivos. Também armazena e garante a segurança, autenticidade, integridade, disponibilidade e confidencialidade dos dados gerados pelo ecossitema. A quarta e última camada é referente às aplicações que irão acessar os dados disponíveis e realizar seu processamento, de acordo com a necessidade.

Figura 5 – Camadas funcionais de uma solução IoT



Fonte: Carissimi (2016, p. 12).

2.1 COMPUTAÇÃO UBÍQUA

O conceito de computação ubíqua diz respeito à evolução da computação em níveis tão altos que os computadores se tornem onipresentes, isto é, integralmente presentes, em todos os possíveis lugares, de forma invisível aos nossos olhos, contendo as características de invisibilidade, proatividade, sensibilidade ao contexto do ambiente, interfaces naturais e processamento descentralizado (SILVA, 2015). Os antigos *mainframes*, atuais computadores pessoais, embora cada vez mais disseminados em nosso cotidiano, ainda não cumprem esses requisitos. Weiser (1991) comenta que não há uma explicação definitiva dos princípios da computação ubíqua nem uma lista de tecnologias envolvidas num possível mundo permeado por dispositivos invisíveis, principalmente porque está em concepção um novo modo de pensar sobre computadores no mundo, que leve em conta o ambiente humano natural e permita que os próprios computadores desapareçam em segundo plano:

> A computação ubíqua é a terceira onda da computação, que está apenas começando. Primeiro tivemos os mainframes, compartilhados por várias pessoas. Estamos na época da computação pessoal, com pessoas e máquinas estranhando umas às outras. A seguir vem a computação ubíqua, a era da tecnologia 'calma', quando a tecnologia recua para o plano de fundo das nossas vidas (WEISER, 1991, p. 94).

Araujo (2003) aponta que o avanço da computação ubíqua se deve aos da computação móvel e pervasiva (Quadro 1). Ela surge da necessidade de mobilidade e funcionabilidade com alto grau de embarcamento dos dispositivos. Considera-se, aqui, computação móvel como o aumento de nossa capacidade de transportar serviços computacionais conosco. A computação pervasiva refere-se ao computador embarcado ao ambiente, isto é, ele obtém informações (contexto) do ambiente e as utiliza dinamicamente.

Quadro 1 – Dimensões da computação ubíqua

	Computação pervasiva	**Computação móvel**	**Computação ubíqua**
Mobilidade	Baixa	Alta	Alta
Grau de "embarcamento"	Alto	Baixo	Alto

Fonte: Araujo (2003, p. 50).

A computação ubíqua é regida por três princípios básicos:

O primeiro diz respeito à diversidade, ou seja, à existência de funções especificas para seus usuários, que poderão gerenciá-las de diferentes dispositivos (diferentemente dos computadores, que possuem propósito geral). O segundo consiste na descentralização das responsabilidades, que passam a ser distribuídas entre os diversos dispositivos que executam determinadas tarefas, cooperando entre si para a construção de inteligência no ambiente (para isso é formada uma rede de dispositivos, num sistema distribuído). O terceiro é o da conectividade sem fronteiras, que caracteriza dispositivos móveis, que precisam de conectividade de forma transparente e entre diversas redes heterogêneas, tais como as redes sem fio de longa ou curta distância (ARAUJO, 2003).

Para se obter onipresença, a solução IoT exige o compartilhamento da compreensão da situação de seus usuários e dispositivos, arquiteturas de *software* e rede disponível nos locais de relevância, além de visar a um sistema autônomo e

inteligente. Cumprindo esses requisitos indispensáveis, torna-se possível uma conectividade inteligente e computadores cientes do contexto em que estão inseridos (GUBBI et al., 2013).

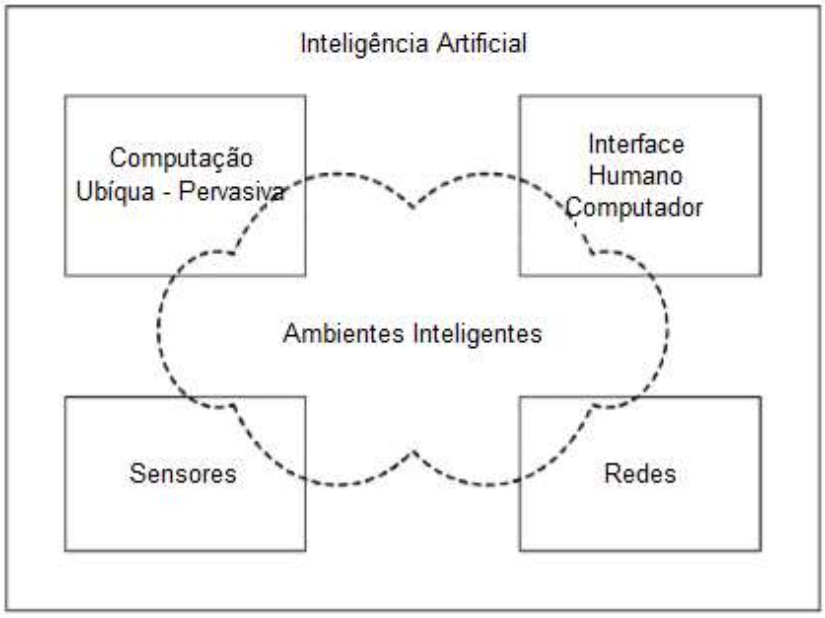
Duas tecnologias críticas para o crescimento da computação ubíqua são a IoT e a chamada *Computação em nuvem* (CÁCERES; FRIDAY, 2011). Esta última consiste em um conceito recente, prometendo serviços confiáveis de virtualização e armazenamento de dados, sem a necessidade de aquisição de equipamentos físicos, por meio de redes de *clusters*, que são aglomerados de computadores com alta capacidade de processamento e conectividade. Empresas como o Google fornecem esse tipo de serviço. Assim, tem-se confiabilidade, escalabilidade e autonomia para fornecer acesso onipresente e descoberta de recursos dinâmicos necessários para as aplicações desenvolvidas (GUBBI et al., 2013).

No entanto, para a IoT emergir com sucesso, a computação precisará ir além dos cenários tradicionais de computação móvel que usam telefones inteligentes e portáteis, evoluindo para a conexão entre objetos existentes todos os dias e incorporando inteligência aos ambientes (GUBBI et al., 2013).

2.2 INTELIGÊNCIA DE AMBIENTES

Schuster (2007) define Inteligência de Ambientes (AmI) como uma área em acelerado crescimento atualmente, e que alia diversos campos de pesquisa e desenvolvimento tecnológico. Esse conceito envolve, sobretudo, o enriquecimento de ambientes mediante sensores e atuadores que analisam e interagem com o próprio ambiente, outros dispositivos ou até mesmo seus usuários, de acordo com as informações coletadas e processadas com o passar do tempo. As principais áreas de pesquisas relacionadas aos AmI estão ilustradas na Figura 6, e seu conceito não esta direcionado a utilização de uma delas em especifico, e sim na utilização destas tecnologias para prover flexíveis e inteligentes ao usuário e ambiente em questão.

Figura 6 – Relação entre AmI e outras áreas



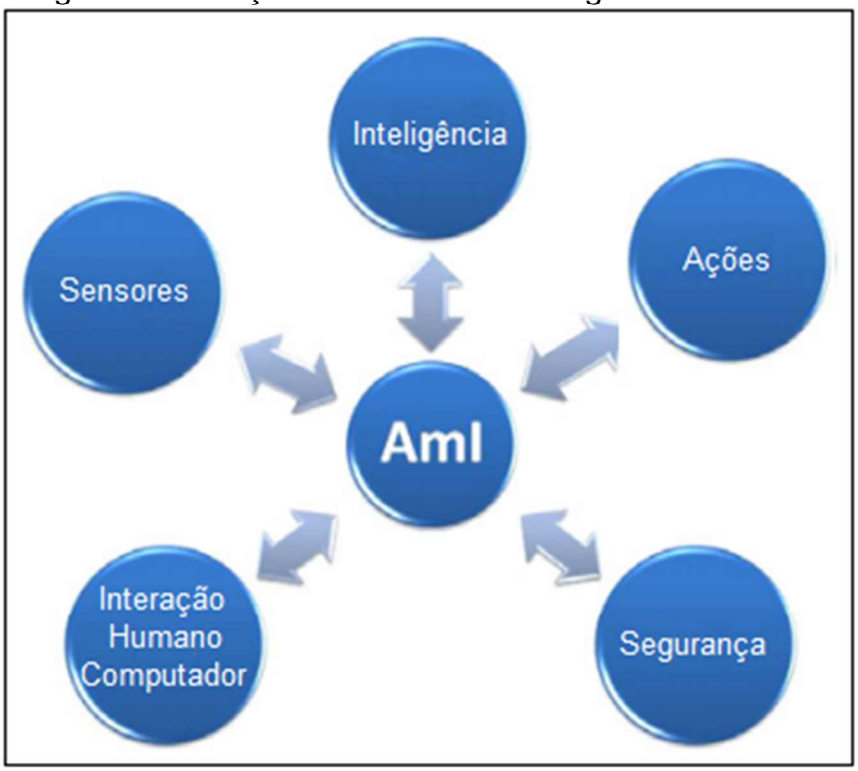
Fonte: Adaptado de Schuster R (2007, p. 214).

Cook, Augusto e Jakkulaa (2009, p. 278) tratam AmI como

> a idéia básica por trás do Ambient Intelligence (AmI) é que, ao enriquecer um ambiente com tecnologia (por exemplo, sensores e dispositivos interconectado através de uma rede), um sistema pode ser construído de tal forma que atua como um "mordomo eletrônico", que detecta características dos usuários e seu ambiente, então raciocina sobre os dados acumulados e, finalmente, seleciona as ações a serem executadas para beneficiar os usuários no ambiente.

Sua principal característica é a interação humano-computador baseada nos conceitos de computação ubíqua e pervasiva, mas há um aspecto importante a mais: o da inteligência. AmI incorporam os estudos e avanços em inteligência artificial baseada em agentes de *software* e robótica. Cook, Augusto e Jakkulaa (2009) também destacam os recursos esperados no sistema como um todo: ele tem de ser sensível, responsivo, adaptável, transparente, onipresente e inteligente. Além disso, definem as tecnologias contribuintes da ciência da computação ao desenvolvimento de AmI em cinco grandes áreas, conforme a Figura 7.

Figura 7 – Relação entre AmI e tecnologias contribuintes



Fonte: Adaptado de Cook, Augusto e Jakkulaa (2009, p. 280).

Essas tecnologias, da Figura 7, são definidas da seguinte maneira:

a) **sensores:** o uso de sensores é a base para o desenvolvimento do sistema. Sem eles, não há como obter informações do meio para que os agentes inteligentes alimentem seus algoritmos;

b) **ações:** os sistemas inteligentes vinculam seu raciocinio através da percepção e ação, isto é, fazem-se necessários dispositivos capazes de efetuar ações executivas em prol de afetar os usuários do sistema;

c) **inteligência:** os sensores fornecem dados convertidos para o meio digital, no qual os algoritmos envolvidos criam links para utilizar os atuadores em beneficio dos usuários no mundo real, a partir de raciocinios como modelagem de usuário, previsões, reconhecimento de atividades, tomada de decisão e raciocínio espaço-temporal;

d) **interação humano-computador:** o Information Society Technologies Advisory Group (ISTAG) (2010) destacou a necessidade do desenvolvimento de interfaces de computador centradas no contexto consciente e natural para uma difusão de AmI. Essas interfaces não

demandam controle do operador e estão integradas à consciência inerente ao ambiente localizado; e

e) segurança: devido aos sensores, os usuários poderão estar compartilhando a todo momento informações pessoais com o sistema, e essas informações precisam estar seguras para que não sejam expostas a terceiros.

Essas grandes áreas do desenvolvimento humano e tecnológico, definidas e explicadas por Cook, Augusto e Jakkulaa (2009), vêm ao encontro de aspectos politicos-sociais muito importantes, bem como o relatório emitido pela ISTAG, em 2010, que explana esses aspectos, tratando de modelos de negócios e tecnológicos determinantes para a aceitação e desenvolvimento do AmI em uma sociedade. São eles:

a) facilitar o contato humano;

b) orientar-se para a melhoria comunitária e cultural;

c) ajudar a construir conhecimento e habilidades para o trabalho, melhorar a qualidade do trabalho, cidadania e escolha do consumido;

d) inspirar confiança e sigilo;

e) ser consistente a longo prazo nos aspectos pessoais, sociais, ambientais e aprendendo ao longo de sua vida, tendo como desafio o desenvolvimento de um ambiente com interações fáceis e simples entre humanos e computadores; e

f) permitir operação por parte de pessoas comuns, ou seja, essa tecnologia deve estar ao alcance do usuário que a possa manipular, criando, assim, a sensação de controle.

Os modelos de negócios derivados de AmI ainda estão em fase de desenvolvimento, mas com a diversidade tecnológica que eles possibilitam a variedade que deve surgir é ampla. O mesmo relatório da ISTAG pontua, ainda, alguns desafios para o pleno densenvolvimento da área:

a) mercados de nicho específico em aplicações industriais, comerciais e públicas que apoiem o desenvolvimento ágil de atividades humanas;

b) *start-up* e *spin-off* de análise e identificação de soluções, reunindo os serviços que atendam às demandas;

c) escalabilidade para conquistar preços acessiveis;

d) economia de atenção do cliente, isto é, prestação de serviços grátis a usuários em troca de publicidades ou serviços complementares;

e) autoprovisão, que utilize sua própia rede de dispositivos e informações para as soluções a um custo próximo de zero.

Ainda no relatório em questão, o ISTAG lista cinco requisitos tecnológicos de AmI:

a) *hardware* muito discreto, o que adentra, de certa forma, os âmbitos da computação ubiqua e pervasiva;
b) infraestrutura de comunicação móvel e fixa sem interrupções;
c) redes de dispositivos dinâmicas e massivamente distribuídas;
d) interfaces humano-computador que tornem as interações mais naturais; e
e) confiabilidade e segurança dos sistemas e informações.

Augusto et al. (2013) diminuem o escopo de AmI para *software,* que apoia os usuários em suas tarefas cotidianas, colocando-o como parte de outro conceito, chamado *Smart Environment.* Eles utilizam o termo SE*,* mas as definições estudadas pelos autores são as mesmas listadas neste trabalho pelo termo AmI. O mais importante, entretanto, é que eles enfatizam que, para o ambiente ser inteligente, há necessidade de uma atitude proativa, isto é, que o ambiente esteja constantemente em busca de maneiras de auxiliar o usuário. Deve ter um comportamento autônomo, mas não a ponto de alterar as rotinas do ambiente sem a liderança do indivíduo, sempre privando por sua saúde e bem-estar.

2.3 CONCLUSÕES SOBRE O CAPÍTULO

Um ecossistema IoT é uma estrutura que têm tendência a aumentar de forma escalável ao número de aplicações envolvidas. Estas aplicações podem utilizar das informações fornecidas por diversos dispositivos diferentes e também utilizadas e fornecidas por outras aplicações. Visto esse cenário e quantidade de tecnologias envolvidas tanto de hardware como de software, é necessário um suporte, ambiente que apoie a interoperabilidade e independência dos dispositivos e aplicações.

3 *MIDDLEWARE*

Bakken (2001, p. 1) define o termo *middleware* como “[...] uma classe de tecnologias de *software* projetadas para resolver as complexidades inerentes à heterogeneidade em sistemas distribuídos”. Em outras palavras, trata-se de uma camada de *software* utilizada para abstrair as dificuldades do processamento cooperativo.

3.1 REQUISITOS

O *middleware* visa a facilitar o desenvolvimento de soluções, fazendo com que os desenvolvedores deixem de lado as preocupações de *hardware* e *software* referentes à comunicação e computação em geral. Para que ele possa suportar a IoT, RAZZAQUE et al. (2015) sugerem uma série de requisitos, que podem ser separados em dois grupos: de serviços e de arquitetura.

Os requisitos de serviços são classificados em funcionais, que implicam em desenvolvimento de atuações do sistema nos dispositivos, rede e dados, e não funcionais, que dão suporte ao funcionamento do sistema.

Os requisitos de serviços funcionais são:

a) **descoberta de recursos:** a IoT utiliza de dispositivos heterogêneos com funções especificas. Sua quantidade, portanto, deve ser exponencial, para que ofereçam as informações ao ecossistema, tornando inviável o cadastro manual de recursos e dispositivos. É necessário que, ao estarem ativos e conectados a alguma rede, eles anunciem presença para automatizar a função;
b) **gerência de recursos:** como ele (quem?) servirá de suporte para diversas aplicações, o uso de recursos de rede e processamento deverá ser gerenciado para poder priorizar algumas aplicações em detrimento de outras;
c) **gerência dos dados:** os dados serão recebidos dos dispositivos e precisam ser recebidos, compactados, filtrados, agregados, armazenados, etc;
d) **gerência de eventos:** há, na IoT, potencialidade para diversos eventos simultâneos a serem observados em tempo real, e deve-se oferecer essa análise de dados em alta velocidade para o processamento da aplicação;

e) **gerência de código:** alocação e reprogramação de códigos auxiliam a selecionar e identificar o conjunto de dispositivos, sensores a serem utilizados para realizar uma tarefa no ambiente.

Os requisitos de serviços não funcionais, por sua vez, são os seguintes:

a) **escalabilidade:** devem comportar o aumento exponencial do ecossistema de dispositivos IoT;
b) **tempo real:** devem prover acesso aos dados e execuções para aplicações que não dependem apenas de como a ação ou o dado são tratados, mas também de quando e de em quanto tempo isso ocorre;
c) **confiabilidade:** devem manter-se operacionais, mesmo na presença de falhas, promovendo comunicação, integração e processamento confiáveis para os mais diversos tipos de aplicação;
d) **disponibilidade:** devem sempre estar disponíveis, principalmente para as aplicações críticas; mesmo em caso de falha em pontos do sistema, sua recuperação deve ocorrer rapidamente, sem obstruir o restante dos serviços;
e) **segurança e privacidade:** devem garantir a segurança das informações recebidas, armazenadas e disponibilizadas, visto que há dados sobre o contexto dos usuários trafegando pela solução.

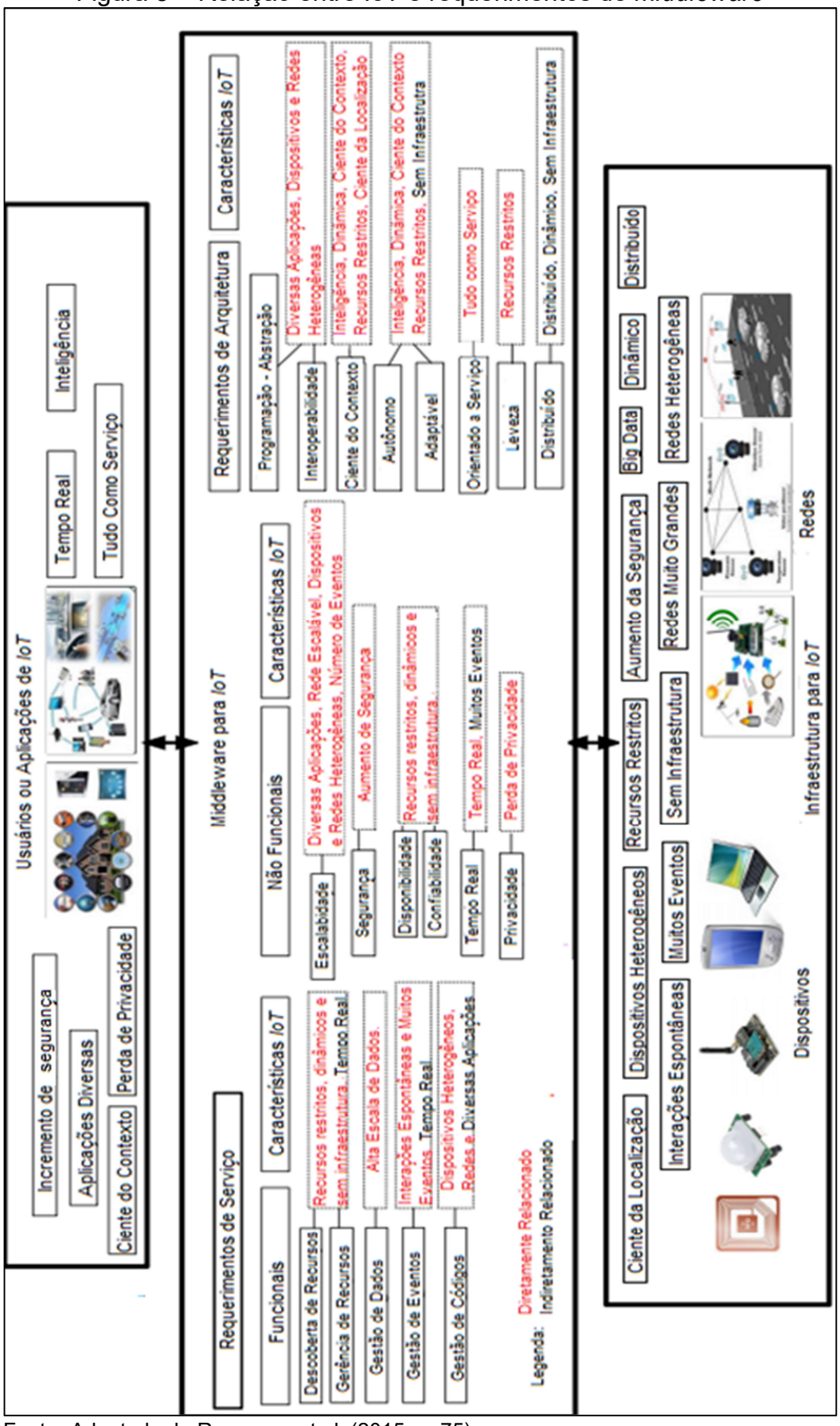
Referentes ao segundo grupo, os requisitos de arquitetura são pensados para suportar o funcionamento do *middleware*. São as preocupações, em nível de implantação:

a) **abstração de programação:** principalmente fornecendo uma Application Programming Interface (API) para desenvolvedores. É necessário isolar o desenvolvimento de aplicações das implicações de envolvidas com a rede e funcionamento do *middleware*, fornecendo recursos, por exemplo, de publicação e recepção de dados;
b) **interoperabilidade:** diz respeito à forma de funcionamento do *middleware* com dispositivos heterogêneos, que podem trocar dados e serviços. Podemos observar esse aspecto sob três pontos: rede, sintáticas e semânticas. A solução poderá passar informações por mais de uma rede de tipos diferentes, com estruturas de formatação e codificação de troca de informações, que analise o contexto a partir do intercâmbio do crescimento de dispositivos, informações e aplicações;

c) **embasamento em serviço:** para oferecer alta flexibilidade na implementação de novas funções, deve fornecer abstrações para o *hardware*, por meio de um conjunto de serviços, como gerenciamento de dados, confiabilidade e segurança;
d) **adaptabilidade:** um ecossistema IoT deve estar em constante mudança, devido a sua ampliação própria e aos desenvolvimentos tecnológicos, tanto de dispositivos quanto de aplicações; por isso, arquitetura de *hardware* deve ser capaz de ajustar-se a todas essas situações;
e) **ciência do contexto:** requisito fundamental para a IoT, a arquitetura de *middleware* deve estar ciente de usuários, dispositivos e ambientes para ofertar de serviços;
f) **autonomia:** as tecnologias participantes de um ecossistema IoT são ativas e devem comunicar-se entre si, sem a necessidade de intervenção humana; e
g) **distribuição:** as tecnologias de uma rede em grande escala trocam informações entre si, devendo ser pensadas para se configurarem de forma geograficamente distribuída.

A Figura 8 relaciona conceitos apresentados por Razzaque et al. (2015), abordados no capítulo **Internet das Coisas**, com os requisitos citados acima, direta (vide texto em vermelho) e indiretamente (texto em preto).

Figura 8 – Relação entre IoT e requerimentos de *middleware*



Fonte: Adaptado de Razzaque et al. (2015, p. 75).

Como o desenvolvimento de *middleware*s é uma área de pesquisa muito ativa, há diversos focos no desenvolvimento de seus sistemas, com diferentes abordagens. Os autores os agrupam em sete diferentes categorias:

a) **embasamento em evento:** neste caso, os componentes do ecossistema comunicam-se por meio de eventos, que são propagados pela aplicação ou componente produtor e recebido pelos consumidores; tipicamente, esses *middlewares* utilizam o padrão *publish/subscribe*;
b) **orientação a serviços:** os *middlewares* deste grupo oferecem suporte à dissociação entre produtores e consumidores e são baseados na arquitetura orientada a serviços, utilizando de suas características, como neutralidade tecnológica, baixo acoplamento, capacidade de reutilização do serviço e descoberta de serviço;
c) **embasamento em Virtual Machine (VM):** aqui, seu desenvolvimento deve oferecer suporte à programação de aplicações em um ambiente seguro para o usuário, utilizando uma infraestrutura virtualizada, em que as aplicações são segmentadas em pequenos módulos e distribuídas pela rede;
d) **embasamento em agentes:** nesta categoria, as aplicações são divididas modularmente para facilitar a inserção e distribuição através da rede, utilizando agentes móveis que migram de nós na rede, mantendo seu estado de execução;
e) **espaço de tuplas:** o espaço de tuplas se trata de um repositório de dados que pode ser acessado simultaneamente; nesta abordagem, cada integrante da arquitetura mantém uma estrutura desse repositório;
f) **orientação a banco de dados:** a rede de sensores e atuadores, neste caso, é vista como um sistema de banco de dados relacional virtual, que utiliza o banco de dados para interoperar dispositivos; e
g) **aplicações especificas:** arquiteturas pensadas para o gerenciamento de recursos das aplicações, implementando técnicas que aprimorem a utilização de rede, infraestrutura para utilização dos aplicativos e dispositivos.

Pires et al. (2015) definem as plataformas de *middleware* como responsáveis pela coleta, gerenciamento e processamento das informações de contexto providas por múltiplas fontes, fornecendo acesso das aplicações a esses dados. Eles também consideram que as plataformas devem cumprir um conjunto de requisitos,

considerados fundamentais na literatura acadêmica em geral, que são: interoperabilidade, descoberta e gerenciamento de dispositivos, interfaces de alto nível, ciência de contexto, escalabilidade, gerenciamento de grandes volumes de dados, segurança e adaptação dinâmica. Muitos deles, como a interoperabilidade e escalabilidade, são inerentes a todas as plataformas de *middlewares;* outros, entretanto, são partilhados pela computação ubíqua e pelo paradigma de IoT, como a ciência de contexto.

Os estudiosos também enfatizam que a adaptação dinâmica do *middleware* se deve a

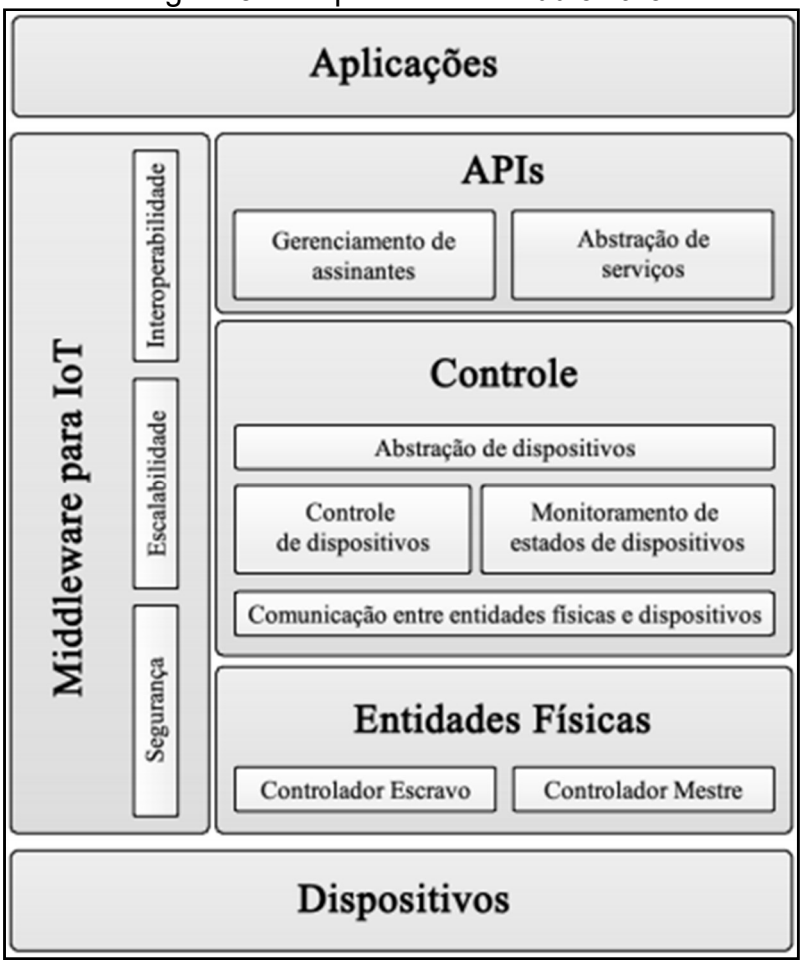
> uma característica comum da infraestrutura de comunicação em ambientes de IoT diz respeito ao fato de que sua topologia é dinâmica e frequentemente desconhecida, visto que dispositivos podem ser integrados ao ambiente e utilizados de maneira oportunista e não previamente planejada (DELICATO et al. apud PIRES et al., 2015, p. 5).

Sendo assim, é extremamente importante que ele suporte a descoberta desde dispositivos presentes no ecossistema dinamicamente a mecanismos para o gerenciamento como desconexão da infraestrutura, alteração de configurações, atualizações e manipulação dos dados.

3.2 ARQUITETURA

Consoante Ferreira (2014) relata que os principais problemas de arquitetura de IoT ocorrem em relação a interoperabilidade, escalabilidade e segurança, e uma arquitetura de *middleware* de sucesso deve endereçá-los, utilizando o máximo de tecnologias e padrões existentes, para segmentar os desenvolvimentos tecnológicos referentes à IoT. A interoperabilidade diz respeito aos requisitos de agilidade e adaptabilidade, e a escalabilidade se refere à disposição para englobar redes de dispositivos e aplicações de qualquer magnitude e segurança em relação à confidencialidade, autenticidade e integridade das informações trafegadas no ecossistema.

Figura 9 – Arquitetura de *middleware*



Fonte: Ferreira (2014, p. 42).

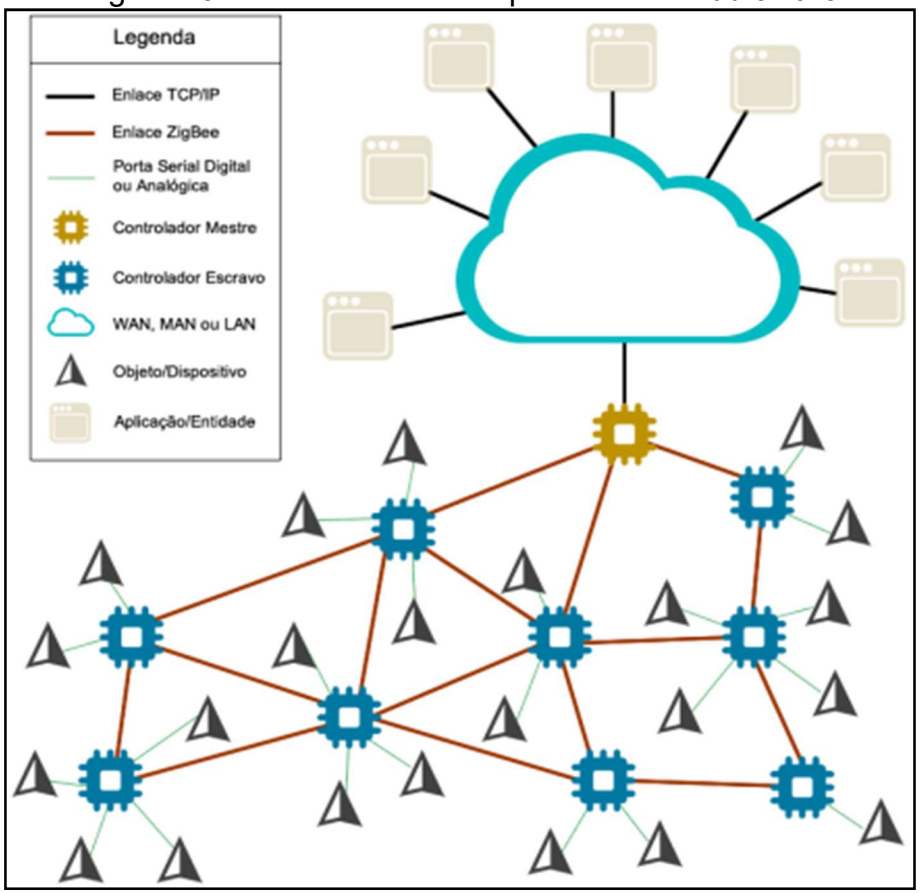
Conforme a Figura 9, Ferreira (2014) apresenta uma proposta de arquitetura de *middleware* voltada à integração de dispositivos e aplicações, utilizando de duas rotinas fundamentais: a de controle e a de monitoramento. O controle consiste em enviar as requisições das aplicações para um dispositivo, por meio de abstrações providas por APIs, utilizando um controlador-mestre. O monitoramento é a verificação do estado de um dispositivo com vistas a obter a ciência de contexto, isto é, a situação dos sensores. Para tanto, os dispositivos enviam seus estados através de um controlador-escravo, que recebe as informações, e o controlador-mestre as envia para as aplicações. Ferreira (2014, p. 46) afirma que essa arquitetura

> foi pensada para dar flexibilidade ao sistema. O controlador-mestre retém as APIs e a abstração lógica dos dispositivos. É nele onde ocorre a maior parte do processamento e onde se localiza a maior parte da inteligência da arquitetura. Os controladores-escravos se conectam fisicamente a um ou vários dispositivos e são responsáveis exclusivamente por ler estados de dispositivos e executar ações neles. Controladores-mestres e escravo podem

> se comunicar via rede sem fio para gerenciar todos os dispositivos e prestar serviços (FERREIRA, 2014, p. 46).

Extensibilidade e transparência são o que tornam o *middleware* abrangente no que se diz a múltiplos dispositivos e aplicações. Adicionando APIs transparentes, as soluções podem ser desenvolvidas de maneira independente a sua plataforma. Pervasividade, unida a tecnologias móveis utilizadas para comunicar todas as partes do *middleware*, torna os dispositivos ubíquos e interoperáveis. Nesse ponto, a proposta de Ferreira (2014) trata os dispositivos que realizam a conexão das coisas a um gateway ou à internet como parte do *middleware*, conforme ilustrado na Figura 10.

Figura 10 – Visão física da arquitetura de *middleware*



Fonte: Ferreira (2014, p. 46).

Conforme a Figura 10, diversos controladores-escravos conectam-se entre si através da tecnologia *ZigBee* em redes no formato de malha; esses dispositivos se conectam às aplicações por meio do controlador-mestre, que realiza a etapa de trazer os dados da rede *ZigBee* para TCP/IP e vice-versa.

As funções do controlador-mestre, segundo Ferreira (2014), são:

a) Trocar informações com qualquer aplicação com que consiga se comunicar, utilizando API, Representational State Transfer (REST) ou Universal Plug and Play (UPnP), via redes TCP/IP;

b) Criar, manter e atualizar a abstração lógica de dispositivos (representação no ambiente virtual que será utilizada nas APIs);

c) Comunicar-se com os controladores-escravos;

d) Tratar informações de requisições de aplicações, criar os comandos corretos para o controlador-escravo e enviar os dados finais ao controlador-escravo correto do dispositivo;

e) Retornar o último estado lido de um dispositivo para a aplicação que requisite tal dado;

f) Registrar aplicações que desejem receber atualizações de estados de dispositivos; e

g) Notificar alterações nos estados de um dispositivo para todas as aplicações registradas, a fim de receber tais notificações.

A forma física do controlador-mestre varia de acordo com a aplicação: pode ser a de um microcomputador ou até mesmo de um *cluster* de servidores, no caso de grandes aplicações. Tendo como demanda de *hardware* a capacidade de operar interfaces TCP/IP[1], ZigBee[2], espaço em disco, processamento e memória Random Access Memory (RAM) condizentes com a solução ou ecossistema envolvido (FERREIRA, 2014).

As funções do controlador-escravo, segundo Ferreira (2014), são:

a) Possuir uma interface, ou várias, de leitura ou escrita digital ou analógica, e escrita Pulse Width Modulation (PWM), para que possa comunicar-se com sensores, atuadores ou diretamente com dispositivos inteligentes conectados diretamente a ele;

b) Rotear, quando requisitado, dados entre outros controladores-escravos e o controlador-mestre;

c) Receber comandos do controlador-mestre, mesmo que antes passem por outros controladores-escravos, e executar os comandos nas interfaces corretas, para, assim, aplicar a ação no dispositivo correto, retornando o

[1] Conjunto de protocolos de comunicação entre dispositivos em rede.
[2] Protocolo está sendo projetado para permitir comunicação sem fio confiável, com baixo consumo de energia e baixas taxas de transmissão para aplicações de monitoramento e controle.

último estado lido de um dispositivo para uma aplicação que requisite tal dado; e

d) Monitorar suas interfaces com dispositivos e, quando perceber alterações, notificar o controlador-mestre sobre os novos valores lidos.

O controlador-escravo é a modalidade de *hardware* mais simples do ecossistema; precisa respeitar os requisitos de recursos escassos, o que significa consumir a menor quantidade possível de energia e de processamento. Não é necessário o armazenamento de informações; ele apenas realiza leituras ou atuações nos objetos ou ambientes (FERREIRA, 2014).

3.3 CONCLUSÕES SOBRE O CAPÍTULO

Em geral, o consenso é de que o *middleware* pode ser considerado uma camada de *software* que tem como objetivo abstrair a comunicação entre dispositivos e aplicações. Segundo Gartner (apud COMPUTERWORLD, 2018), a receita do mercado de *middlewares* atingiu US$ 28,5 bilhões em 2017, um aumento de 12,1% em relação ao ano anterior, e chegará a US$ 30 bilhões em 2018. Essa demanda crescente gera a necessidade de aprofundamento nos estudos da área.

Em suma, o *middleware* tem de prover meio de comunicação, armazenamento de eventos e disponibilização dos dados armazenados em tempo real. Além disso, é importante ter formas de mapeamento e gerenciamento do ecossistema de coisas, provendo a interoperabilidade entre dispositivos heterogêneos e aplicações com diferentes finalidades.

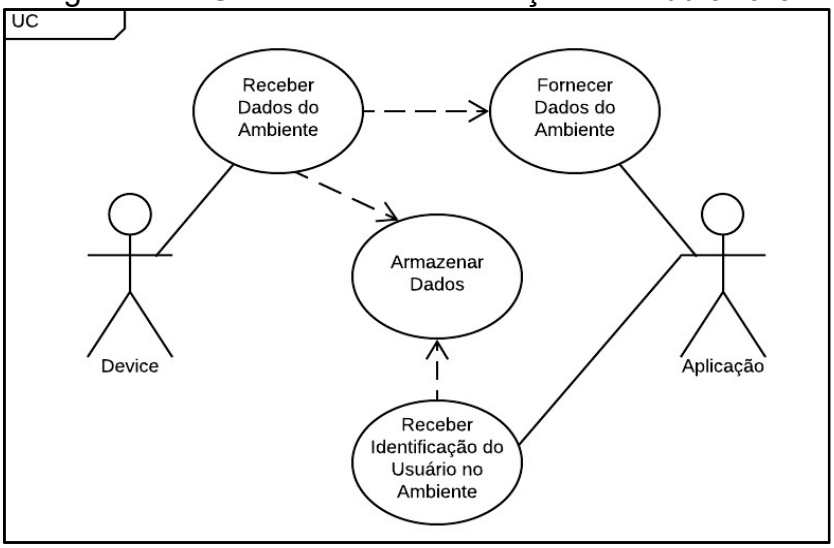
4 IMPLEMENTAÇÃO DE UMA CAMADA DE *MIDDLEWARE*

O principal objetivo deste trabalho é proporcionar conectividade e interoperabilidade a um ecossistema, no qual dispositivos capturem informações referentes aos usuários que presentes no ambiente e um sistema de reconhecimento facial realize sua identificação. Essa camada de *middleware* é implementada para fornecer funcionalidades – a saber, conectividade, interoperabilidade, persistência e contexto da informação adquirida – aos dispositivos de captura de imagem desenvolvidos por Suzin (2018) e aplicação de reconhecimento facial desenvolvida por Chaves (2018), conforme será explicado na Seção 4.3.

4.1 CASO DE USO

Os casos de uso ilustrados na Figura 11 abrangem os fluxos que o *middleware* precisa cumprir para enquadrar-se no cenário de validação de seu funcionamento. **Receber os dados do ambiente**, **fornecer os dados desse mesmo ambiente**, **armazenar esses dados** e **receber identificação do usuário no ambiente** são as funcionalidades para que ele possa persistir o contexto do ambiente. Esse contexto se trata da identificação e indexação dos dados de usuários, dispositivos, data, hora e localização de uma sala de aula, realizada por dispositivos e aplicações.

Figura 11 – Caso de uso de validação do *middleware*



Fonte: elaborada pelo autor.

4.1.1 Receber dados do ambiente

Para que o *middleware* receba os dados do ambiente é necessário que dispositivos capturem essas informações e enviem-nas, possibilitando, assim, a interação. Esses dispositivos estão conectados à rede de internet e registrados no tópico MQTT (protocolo de comunicação que será descrito na Seção 4.2.3), referente ao fluxo de dados programado no Node-RED[3]. O tópico é a maneira como o protocolo MQTT gerência a comunicação, através de inscrições de quem deseja receber os dados e publicações de quem deseja transmitir.

Quadro 2 – Receber dados do ambiente: caso de uso

Ator	Dispositivo.
Descrição	O dispositivo, após reconhecer uma presença, publica a imagem da câmera, data/hora, dispositivo e localização no tópico MQTT.
Pré-condições	Estar conectado, registrado e ter detectado presença.
Pós-condições	Ter enviado informações do ambiente ao *middleware.*
Fluxo principal	a) Captura da imagem e das informações; b) Criação de mensagem com os dados, no padrão JavaScript Object Notation (JSON[4]); c) Publicação dos dados.

Fonte: elaborado pelo autor.

4.1.2 Fornecer dados do ambiente

Ao receber as informações do dispositivo, o middleware as transmite às aplicações que tenham interesse. Para tanto, elas deverão estar inscritas no tópico MQTT, referente ao fluxo de informações do Node-RED desejado.

[3] Ferramenta de programação baseada em fluxos disponível em: https://nodered.org/.
[4] Padrão de troca de dados entre sistemas, maiores informações em: https://www.json.org/.

Quadro 3 – Fornecer dados do ambiente: caso de uso

Ator	Aplicação.
Descrição	Recebe os dados referentes ao ambiente fornecidos pelo dispositivo.
Pré-condições	Estar conectada e registrada.
Pós-condições	Obtenção das informações.
Fluxo principal	Os dados são recebidos para a aplicação utilizá-los.

Fonte: elaborado pelo autor.

4.1.3 Receber identificação do usuário no ambiente

Após a aplicação receber as informações do ambiente, ela realiza seu processamento para identificação do usuário e, então, envia um identificador para que o *middleware* gere o contexto das informações recebidas, por meio de seu banco de dados relacional.

Quadro 4 – Receber identificação do usuário no ambiente: caso de uso

Ator	Aplicação.
Descrição	Após ter recebido as informações do ambiente, a aplicação processa essas informações e identifica o usuário.
Pré-condições	Estar conectada, registrada e ter identificado o usuário.
Pós-condições	Ter enviado informações do usuário.
Fluxo principal	Enviar informações do usuário identificado.
Fluxo alternativo	Enviar informação de que o usuário é desconhecido.

Fonte: elaborado pelo autor.

4.1.4 Armazenar dados

Todas as informações recebidas pelo *middleware* são armazenadas e interagem entre si por meio de um banco de dados relacional, para que os dados tenham o contexto do evento de sua origem. As informações são fornecidas pelos dispositivos e aplicações para que sejam utilizadas em diversas tarefas que acessarão e complementarão o contexto quando necessário. Neste caso, são dois os eventos armazenados: os **dados recebidos do ambiente** e a **identificação do usuário**.

Quadro 5 – Armazenar dados: caso de uso

Atores	Aplicação e dispositivo.
Descrição	Toda informação recebida é armazenada por meio de uma base de dados relacional.
Pré-condições	Ter recebido informações.
Pós-condições	Informações recebidas são armazenadas.
Fluxo principal	A informação é persistida na base de dados.
Fluxo alternativo	A informação é invalidada e descartada.

Fonte: elaborado pelo autor.

4.2 TECNOLOGIAS

Um *middleware* é composto por um conjunto de soluções, buscando atender às características já mencionadas no Capítulo 3. Para isso, é necessário um conjunto de tecnologias de *hardware* e *software* que sejam aplicadas conforme a arquitetura utilizada por Carissimi (2016) na Figura 4. Essas tecnologias serão detalhadas nas próximas seções.

4.2.1 Google Compute Engine

Inicialmente, este projeto estava sendo desenvolvido em uma máquina virtual em um computador comum. Entretanto, para uma implementação definitiva, essa situação não cumpria os requisitos de disponibilidade e escalabilidade de que um ecossistema IoT necessita. Portanto, para atender a esses requisitos, escolheu-se utilizar o Google Compute Engine. Por meio dessa ferramenta, é possível criar máquinas Linux Server flexíveis, em que, conforme aumenta a necessidade de recursos, podem-se aumentar as especificações e o armazenamento do servidor.

Um ecossistema IoT dispõe de dispositivos novos sendo implantados a todo momento, e, assim, a infraestrutura do *middleware* obtém os requisitos acima descritos por meio da infraestrutura do Google. Como partida, a máquina foi configurada, como ilustrado na Figura 12, como uma *g1-small* com 1,7 GB de memória e disco de 10 GB. O modelo *g1-small* tem 0,5 de uma Virtual CPU (vCPU) completa,

podendo ter picos de utilização de uma vCPU inteira por períodos curtos. Essa configuração pode ser ampliada conforme se expande a escala do ecossistema.

Figura 12 – Especificações da VM

Fonte: elaborado pelo autor.

4.2.2 Ubuntu Server

O sistema operacional Ubuntu[5] baseado em Linux[6] foi escolhido devido a sua rotina semestral de atualizações que corrigem falhas de segurança e realizam otimizações do sistema. Além disso, é um sistema de código aberto e a sua arquitetura proporciona uma maior disponibilidade, visto que não é preciso interromper os serviços para instalar ou atualizar *softwares*.

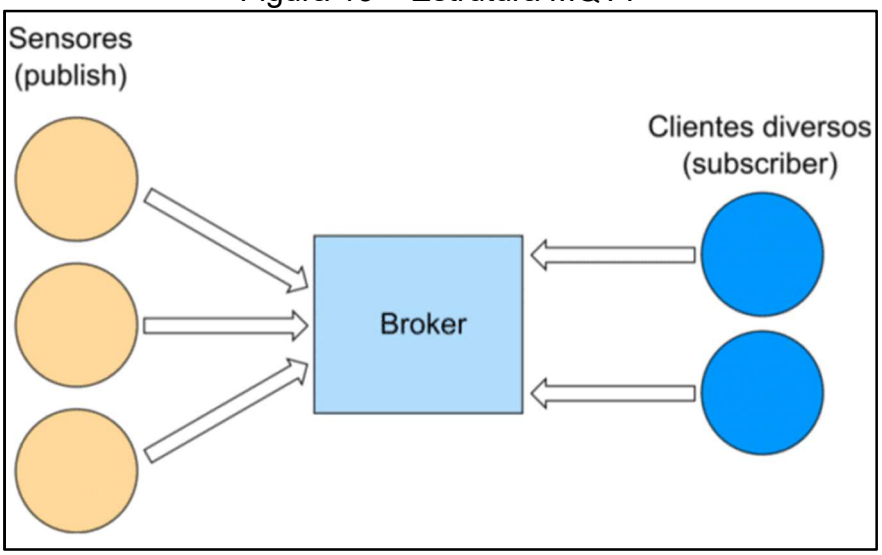
4.2.3 Mosquitto

A comunicação da camada de *middleware* utiliza do MQTT Broker, de código aberto – Mosquitto[7]. O MQTT, acrônimo de *Message Queuing Telemetry Transport*, se trata um protocolo de mensagens da camada de aplicação, projetado para ser utilizado em sensores e dispositivos com poucos recursos computacionais. Ele é responsável por gerenciar a comunicação através do paradigma *pub/sub*. Este paradigma trata de como o protocolo gerencia a comunicação em tópicos. Os dispositivos ou aplicações interessadas em enviar informações, publicam em um

[5] Sistema operacional de código aberto construído a partir do núcleo Linux pela Canonical e disponível em: https://www.ubuntu.com/.
[6] Termo geralmente empregado utilizado para sistemas operacional que utilizam o Kernel Linux.
[7] Broker MQTT de código aberto disponível em: https://mosquitto.org/.

determinado tópico e as interessadas em receber este determinado tipo de informação, inscrevem-se neste mesmo tópico e ficam o monitorando, para receber as publicações realizadas a qualquer momento. Conforme pode-se visualizar na Figura 13, sensores publicam em um determinado tópico no MQTT Broker, e os clientes inscritos nesses tópicos recebem as informações, para utilizá-las conforme sua aplicação.

Figura 13 – Estrutura MQTT



Fonte: Barros (2015).

O MQTT é otimizado para redes TCP/IP não confiáveis ou de alta latência, será responsável pela comunicação entre as “coisas” do sistema IoT. Por meio de seu paradigma *pub/sub*, as aplicações e dispositivos poderão transmitir e receber as informações necessárias sem interferência de mais recursos, a não ser que isso se faça necessário.

4.2.4 *Node-RED*

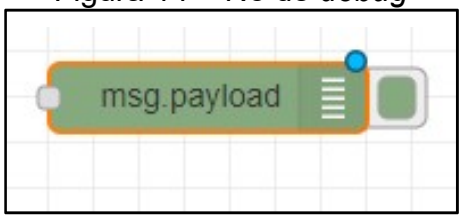
O Node-RED é uma ferramenta de programação baseada em fluxos para dispositivos de *hardware*, APIs e serviços online. Ele é um sistema de código aberto que fornece uma interface *web*, facilitando o desenvolvimento de fluxos por meio de uma ampla variedade de nós disponíveis. Seu objetivo é facilitar a implementação de projetos em ecossistemas IoT em tempo de execução.

A programação baseada em fluxos foi desenvolvida originalmente por J. Paul Morrison na década de 1970 segundo o próprio portal do Node-RED. O paradigma é uma maneira de descrever a funcionalidade de um aplicativo através de “nós”, que possuem seu propósito bem definido. Recebendo alguns dados, efetuam processamentos e retornam seu resultado. É, em síntese, uma rede de nós que irá compor uma solução.

No presente trabalho, foram utilizados diferentes tipos de nós – a saber, ***debug*, MQTT, JSON, *template*, *string* e Database (DB)**:

a) O **nó de *debug*** (Figura 14) permite que se identifique, por meio da interface gráfica ou de terminal de texto, o que está acontecendo com as informações no fluxo de trabalho; ele pode ser adicionado e conectado a todos os demais nós do *Node-RED* para acompanhamento e identificação de problemas.

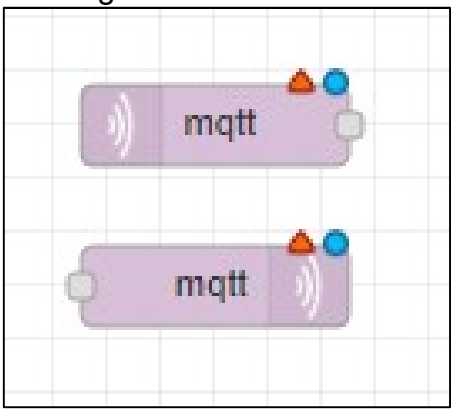
Figura 14 – Nó de *debug*



Fonte: elaborado pelo autor.

b) O **nó MQTT** (Figura 15) possui duas variações. A primeira é de **publicação**, em que se configura o tópico do MQTT Broker, que publicará os dados recebidos. A segunda é de **inscrição**, responsável por monitorar o tópico do MQTT Broker configurado e por, em qualquer publicação, repassar esses dados a sua próxima conexão.

Figura 15 – Nó MQTT



Fonte: elaborado pelo autor.

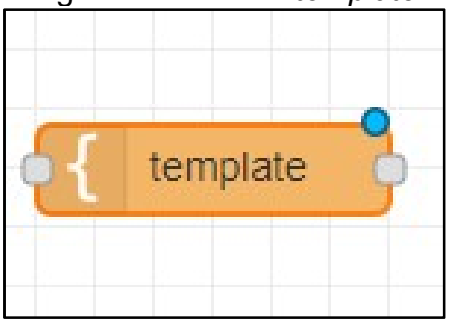
c) O **nó de JSON** (Figura 16) tem a função de receber uma s*tring* formatada no padrão JSON, e convertê-lo em um objeto JSON, e vice-versa, para facilitar a manipulação dos dados pelos nós seguintes.

Figura 16 – Nó JSON

Fonte: elaborado pelo autor.

d) O **nó de *template*** (Figura 17) serve para realizar a formatação de *Strings* com os dados recebidos; nesse caso, são formatadas as *querys* para inserção das informações na base de dados, sua diferença perante o nodo *String* é que não possui operações básicas de manipulação de *Strings*, como concatenação e corte de palavras.

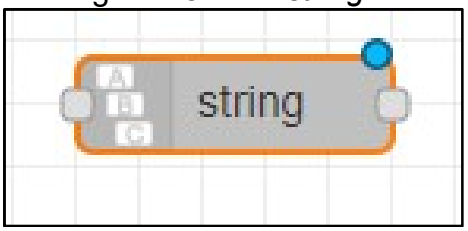
Figura 17 – Nó de *template*



Fonte: elaborado pelo autor.

e) O **nó *string*** (Figura 18) aplica, justamente, métodos de manipulação de *string* para alterar o contexto da mensagem.

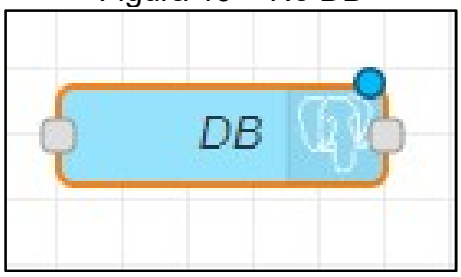
Figura 18 – Nó *string*



Fonte: elaborada pelo autor.

f) O **nó Database (DB)** (Figura 19) é utilizado para gerar a conexão, realizar consultas e inserções no banco de dados PostgreSQL[8], o qual será abordado na Seção 4.2.6. Ele recebe as *querys* já formatadas e as executa. Caso possua algum dado de retorno, repassa-o em sua saída.

Figura 19 – Nó DB



Fonte: elaborado pelo autor.

4.2.5 JSON

O termo JSON diz respeito a um padrão para troca de dados simples e rápida entre sistemas, possibilitando o envio de informações de diversos tipos ao MQTT Broker e podendo ser tratado pelo Node-RED. Seu modelo de representações será detalhado para cada aplicação no decorrer da Seção 4.3.

4.2.6 PostgreSQL

Todas essas informações e contextos são armazenados no PostgreSQL, um sistema gerenciador de banco de dados relacional de código aberto. Ele será utilizado como repositório de eventos para armazená-las e disponibilizá-las ao ecossistema. Estarão mais bem detalhadas sua utilização e aplicação no caso de uso apresentado na Seção 4.3.4.

4.3 ARQUITETURA

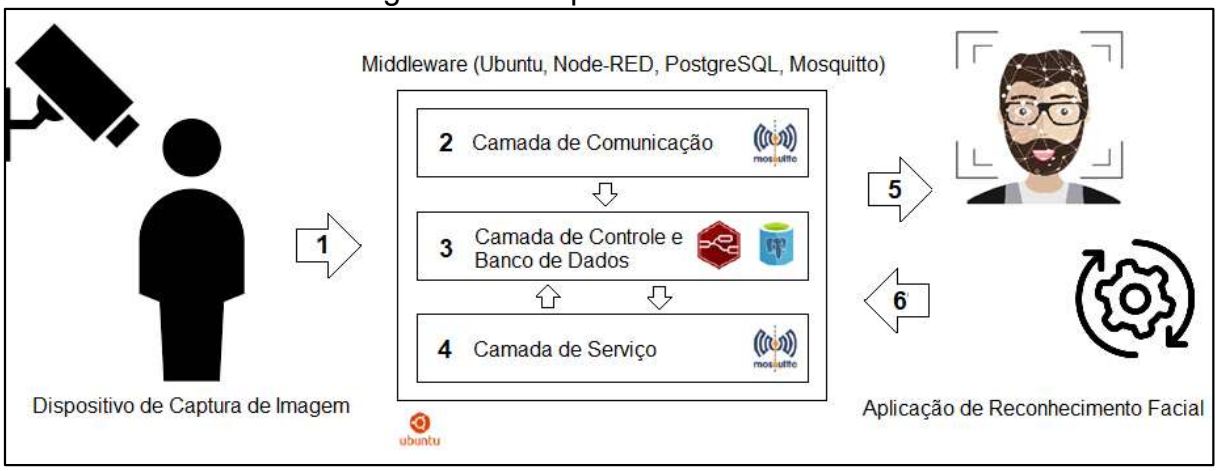
A arquitetura pode ser observada em três camadas (conforme Figura 20), adaptada de Carissimi (2016), Figura 24 e complementada com a arquitetura proposta

[8] Sistema gerenciador de banco de dados relacional de código aberto disponível em: https://www.postgresql.org/.

por Ferreira (2014), abstraindo a implementação e modularidade da camada de *middleware*, facilitando sua visualização para implementações e manutenções.

A camada de comunicação (2) é responsável pelo recebimento das informações enviadas pelos dispositivos (1), neste caso, de captura de imagem e do ambiente. A camada de controle e banco de dados (3) realiza o gerenciamento dos dispositivos e suas abstrações, traduzindo suas requisições em comandos para serem enviados aos dispositivos ou aplicações e armazenando no banco de dados. Já a camada de serviço (4) é responsável por enviar as informações referentes ao dispositivo de captura de imagem (5), neste caso, à aplicação de reconhecimento facial, que irá identificar qual o usuário referente a imagem. Ao terminar seu procedimento a aplicação irá retornar o identificador (6) através da camada de serviço (4) que irá entregar novamente ao Node-RED (3) para que efetue o relacionamento com o evento da captura de imagem e armazená-lo no banco de dados.

Figura 20 – Arquitetura *middleware*



Fonte: elaborado pelo autor.

O *middleware*, nesta arquitetura, está implementado independente da aplicação ou do tipo de dispositivo, operando de forma genérica. Para ser utilizado, é necessário que os dispositivos e aplicações tenham suas funções e estrutura de dados bem definidas e utilizar de JSON para transmiti-las e recebê-las por meio do protocolo MQTT. Também é necessária a construção do fluxo no *Node-RED* e a preparação das tabelas no *PostgreSQL* pelo administrador do sistema.

4.3.1 Camada de serviço

A camada de serviço realiza a integração do *middleware* com aplicações já existentes e interessadas nas informações geradas pelo sistema IoT. Serve-se de APIs ou protocolos da camada de aplicação para enviar e receber dados. Na implementação no *middleware*, o Broker MQTT abrange essa camada, sendo necessário que as aplicações se inscrevam em um tópico para receber os dados e realizar a identificação do usuário em um JSON.

Para validar a camada de serviço, desenvolveu-se uma aplicação em *Python*[9] que recebe o JSON (Quadro 6), identifica a imagem e realiza a publicação (Quadro 7) de uma identificação referente à matrícula para o usuário.

Quadro 6 – JSON gerado pelo *Node-RED*

Nome do campo	**Tipo do campo**
id_Presenca	Número Inteiro
Data	Texto
Device	Texto
Localização	Texto
Imagem	Texto

Fonte: elaborado pelo autor.

Quadro 7 – JSON gerado pela aplicação de reconhecimento facial

Nome do campo	**Tipo do campo**
id_Presenca	Número Inteiro
Matricula	Texto

Fonte: elaborado pelo autor.

4.3.2 Camada de controle

A camada de controle encarrega-se da abstração lógica dos dispositivos com os *softwares* e provê formas de os usuários administradores criarem novos fluxos por

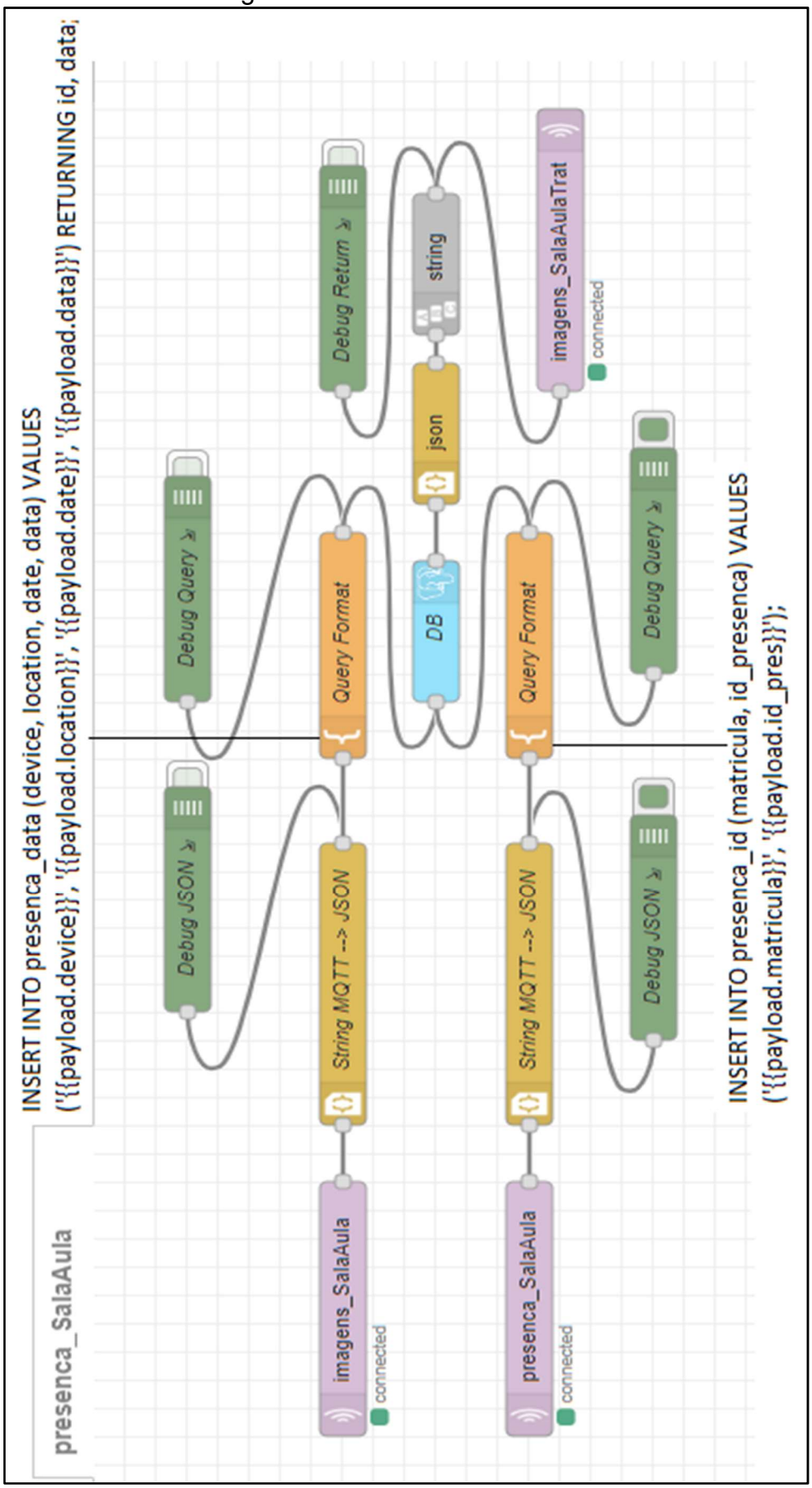
[9] Linguagem de programação, maiores informações em: https://www.python.org/.

meio de uma interface *web*[10]. Utiliza o *Node-RED* como responsável por receber esses dados, nos formatos JSON definidos nas Seções 4.3.1 e 4.3.3.

Conforme a Figura 21 (na página 53), por meio da inscrição no tópico “imagens_SalaAula”, o Node-RED recebe as informações do ambiente, disparadas pelos dispositivos em formato de texto. Ele remonta a *string* em um JSON e formata a *query* (vide Figura 21) para inserir essas informações na base de dados. Assim, é retornado o índice da tabela para enviar os dados às aplicações interessadas, por meio de uma publicação no tópico “imagens_SalaAulaTrat”. A aplicação que irá identificar o usuário está inscrita nesse tópico e recebe a informação para posteriormente publicá-la em “presença_SalaAula”. Este último realiza o mesmo procedimento de remontagem do JSON e formatação da *query* para inserção da informação na base de dados.

[10] Acessada pela URL: http://35.199.123.29:1880.

Figura 21 – Fluxo do *Node-RED*



Fonte: elaborado pelo autor.

4.3.3 Camada de comunicação

A camada de comunicação é responsável por conduzir mensagens dos dispositivos à camada de controle e vice-versa. Tem de tornar a comunicação mais eficiente para a rede. O Broker MQTT Mosquitto é utilizado para fornecer essa conectividade, que demanda poucos recursos de processamento e rede, podendo ser empregada até em redes móveis. Em nosso caso de uso, os dispositivos irão enviar um JSON (conforme o Quadro 8) através do tópico “imagens_SalaAula”, para que o Node-RED, que está na camada de controle, execute suas tarefas.

Quadro 8 – JSON gerado pelo *Raspberry Pi*[11]

Nome do campo	**Tipo do campo**
Data	Texto
Device	Texto
Localização	Texto
Imagem	Base 64 – texto

Fonte: elaborado pelo autor.

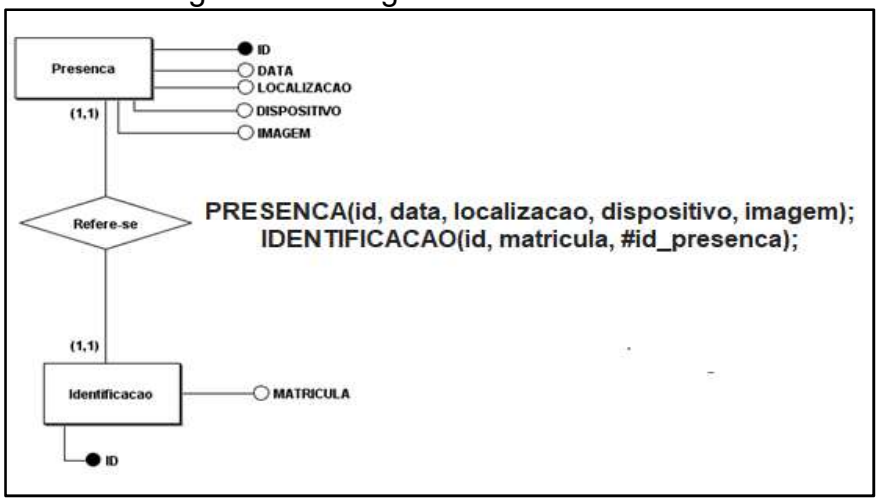
4.3.4 Banco de dados

As informações do ambiente e dos usuários são armazenadas no banco de dados relacional PostgreSQL, escolhido por se tratar de um banco de dados de código aberto, como as demais ferramentas deste projeto. Utilizou-se sua lógica relacional entre eventos físicos e informações resultantes das aplicações para gerar ciência de contexto.

Conforme o modelo apresentado na Figura 22, a tabela PRESENCA armazena o evento físico, gerado por algum dispositivo, referente à presença de aluno na sala de aula. São armazenadas informações como índice, data, localização, dispositivo e a imagem capturada. Essa tabela se relaciona com a identificação do aluno pela chave estrangeira “#id_presenca” com a tabela IDENTIFICACAO, que possui a matrícula do aluno.

[11] Uma série de computadores de placa única de tamanho reduzido, maiores informações em: https://www.raspberrypi.org/.

Figura 22 – Diagrama relacional do DB



Fonte: elaborado pelo autor.

A partir dessas relações, um contexto das informações coletadas pelos dispositivos e pela identificação facial é gerado; este pode ser acessado e utilizado por outras aplicações. Na Figura 23, verifica-se uma *query* que pode ser executada no pgAdmin[12], um software de gerência do PostgreSQL, que irá consultar as informações no banco de dados.

Figura 23 – *Query* do contexto do caso de uso

```
1  select presenca_data.id, presenca_data.date, presenca_data.location,
2  presenca_data.device, presenca_data.data, presenca_id.matricula
3      from presenca_data
4      inner join presenca_id
5          on (presenca_id.id_presenca = presenca_data.id)
```

Fonte: elaborado pelo autor.

O contexto das informações obtidas no presente caso está disponível na Figura 24. Dessa forma, verifica-se que o aluno X, ou usuário desconhecido, esteve na sala de aula Y em um determinado momento.

[12] Sistema Gerenciador de Banco de Dados (SGDB) do PostgreSQL, acessado pela URL: http://35.199.123.29:5050.

Figura 24 – Retorno da *query* de contexto no pgAdmin

Data Output | Explain | Messages | Notifications | Query History

	id bigint	date timestamp without time zone	location character (20)	device character (20)	data text	matricula character (20)
1	49	2018-10-16 21:23:58	71 - 101	raspSala101	&#x...	123456789
2	50	2018-10-16 21:25:16	71 - 101	raspSala101	&#x...	123456789
3	52	2018-10-16 22:31:09	71 - 101	raspSala101	&#x...	123456789

Fonte: elaborado pelo autor.

4.4 CASO DE TESTE

Com o objetivo de validar seu funcionamento, desenvolveu-se um *software* em *Python* e a instalação de um *Raspberry Pi* em uma sala, com vistas a simular o dispositivo. Para simular a aplicação, foi desenvolvido um *software* em *Python* que fica em execução em um computador. Esses dois serão mais bem descritos nas Seções 4.4.1 e 4.4.2.

4.4.1 Dispositivo

Para ser utilizado como dispositivo, foi selecionado um Raspberry Pi, pois seu sistema operacional, o *Raspbian*[13], é baseado em Linux, e porque estava disponível para utilização no projeto. Nele efetuou-se a instalação de uma webcam USB, conforme ilustrado na Figura 25, e desenvolveu-se uma aplicação em *Python*, que, ao ser executada, captura a imagem da webcam (Linha 36 da Figura 26), codifica-a através do método Base64 (Linha 41 da Figura 26) e formata um JSON (Linha 17 da Figura 26) possuindo as informações referentes a data, horário, localização, identificação do dispositivo e imagem. Em seguida, isso é publicado no tópico “imagens_SalaAula” (Linha 25 da Figura 26), receptor das informações dos dispositivos que contenham esse formato de informação no ecossistema IoT em questão. A execução dessa aplicação e seu resultado podem ser verificados na Figura 27 que ilustra o processo de captura de imagem e o JSON formatado e publicado.

[13] Sistema operacional baseado em Linux para Raspberry Pi, disponível em: https://www.raspberrypi.org/downloads/raspbian/.

Figura 25 – *Raspberry Pi* e *webcam*

Fonte: acervo pessoal.

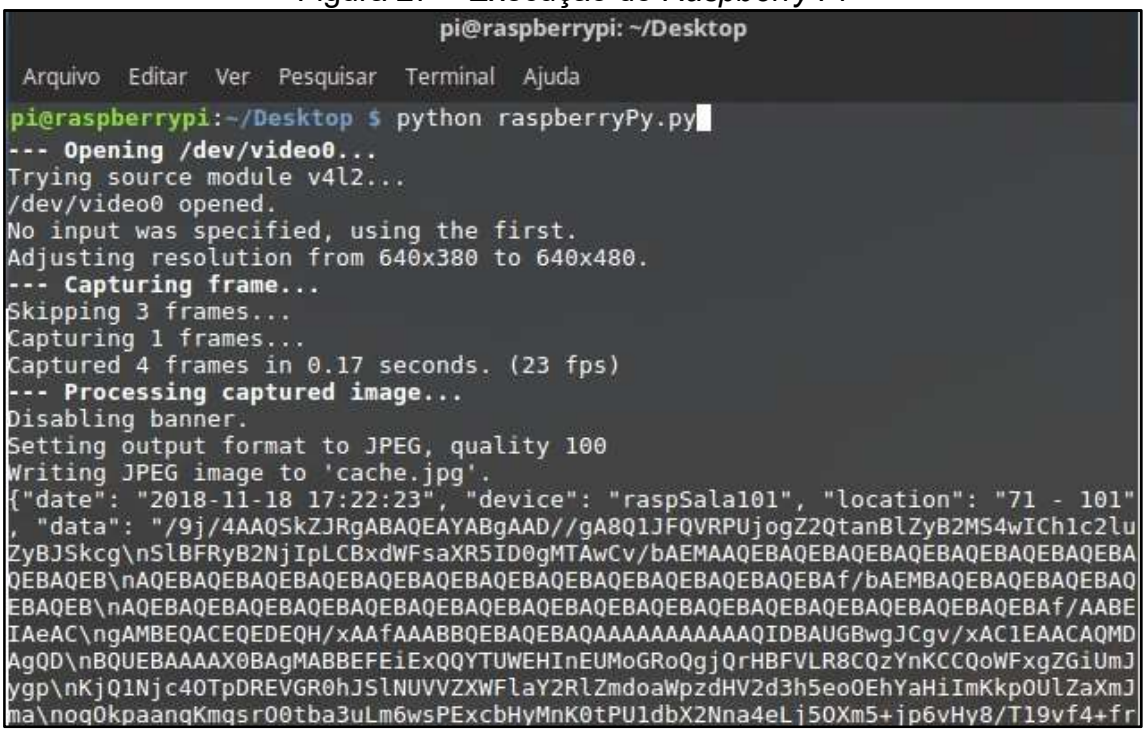
Figura 26 – Código *Python* do *Raspberry Pi*

raspberryPy.py

```
#-------------------------------------------------------------------------#
import time
import datetime
import paho.mqtt.client as paho
import base64
import json
import string
import os
#-------------------------------------------------------------------------#
broker = "35.199.123.29"
topic = "imagens_SalaAula"
device = "raspSala101"
location = "71 - 101"
#-------------------------------------------------------------------------#
def pub_mqtt(date, device, location, image):
        #GERA STRING JSON
        data = '{"date": "'+str(date)+'", "device": "'+str(device)+'",
        "location": "'+str(location)+'", "data": "'+image+'"}'
        print(data)
        #CONECTAR AO BROKER MQTT
        client= paho.Client(device)
        client.connect(broker)
        client.loop_start()
        #PUBLICA NO TOPICO MQTT
        client.publish(topic,data)
        #FINALIZA COMUNICACAO
        time.sleep(4)
        client.disconnect()
        client.loop_stop()
#-------------------------------------------------------------------------#
#GERA DATA E HORA
ts = time.time()
st = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
#CAPTURA IMAGEM DA WEBCAM
os.system('fswebcam -r 640x380 --no-banner -S 3 --jpeg 100 --save cache.jpg')
#GERA IMAGEM EM BASE64
image = open('cache.jpg', 'rb')
image_read = image.read()
image_64_encode = base64.encodestring(image_read)
buf =  image_64_encode.replace('\n', '\\n')
pub_mqtt(st, device, location, buf)
#-------------------------------------------------------------------------#
```

Fonte: acervo pessoal.

Figura 27 – Execução do *Raspberry Pi*



Fonte: acervo pessoal.

4.4.2 *Middleware*

A camada de *middleware* disponibiliza o Broker MQTT para que o dispositivo e as aplicações efetuem e recebam suas publicações. Após o dispositivo publicar um evento, o *Node-RED*, conforme ilustrado na Figura 21, formata o JSON e gera uma *query* para persistência das informações no PostgreSQL. Esse processo retorna um identificador, e o *Node-RED* remonta uma publicação incluindo esse identificador para que as aplicações que estejam inscritas recebam as informações dos dispositivos. Na Figura 28, temos um registro de um evento de um dispositivo e de uma aplicação.

Essas aplicações realizam os processamentos e publicam-nos em seu tópico MQTT respectivo. Neste caso, a aplicação realiza o reconhecimento do aluno a partir das informações e publica um JSON em que o *Node-RED* relaciona o identificador do usuário com as informações do dispositivo (vide Figura 21). Todo esse procedimento gera um contexto de informação do ambiente, conforme se ilustra na Figura 29.

Figura 28 – *Debug Node-RED*

Fonte: acervo pessoal.

Figura 29 – Exemplo de contexto

	id bigint	date timestamp without time zone	location character (20)	device character (20)	matricula character (20)
1	62	2018-11-18 17:22:23	71 - 101	raspSala101	29051992

Fonte: acervo pessoal.

4.4.3 Aplicação

A aplicação é um processo, desenvolvido em *Python*, que é executada em um computador. Essa aplicação, está inscrita no tópico MQTT “imagens_SalaAulaTrat” (Linha 42 da Figura 30) e receberá qualquer notificação publicada (Linha 15 da Figura 30). Ao receber a notificação, ela formata o JSON (Linha 18 da Figura 30), com as informações ilustradas na Figura 31, e remonta a imagem recebida em um arquivo de formato .JPG (Linha 29 da Figura 30). Depois, publica no tópico MQTT “presença_SalaAula” um número de matrícula (Linha 35 da Figura 30).

Todo esse processo tem como objetivo simular o fluxo que o *middleware* irá fornecer para uma aplicação de reconhecimento facial. Os dados do dispositivo serão

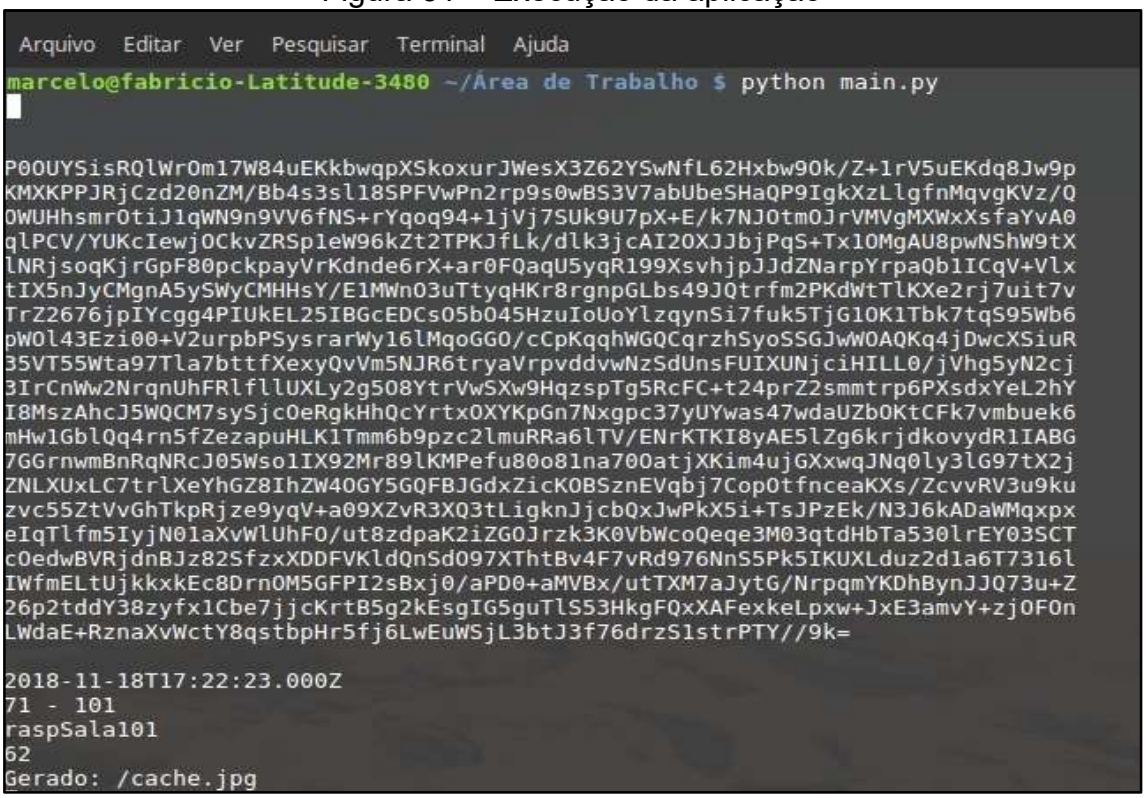
recebidos e efetuar-se-á seu procedimento para reconhecimento do aluno. Após o processamento, o resultado é publicado, sendo relacionado com as informações do dispositivo.

Figura 30 – Código *Python* da aplicação do notebook

```python
#-----------------------------------------------------------------------------#
import time
import paho.mqtt.client as paho
import base64
import json
import string
#-----------------------------------------------------------------------------#
broker = "35.199.123.29"
topicSubscribe = "imagens_SalaAulaTrat"
topicPublish = "presenca_SalaAula"
device = "servImagens"
matriculaTeste = 123456789
#-----------------------------------------------------------------------------#
#METODO UTILIZADO AO RECEBER ALGUMA NOTIFICACAO
def on_message(client, userdata, message):
    time.sleep(1)
    #FORMATA O JSON
    jData = json.loads(message.payload.decode("utf-8"))
    aux = str(jData['data'])
    aux = aux.replace('&#x2F;', '/')
    aux = aux.replace('&#x3D;', '=')
    print(aux)
    print(jData['date'])
    print(jData['location'])
    print(jData['device'])
    print(jData['id'])
    #MONTA A IMAGEM
    fh = open("cache.jpg", "wb")
    fh.write(aux.decode('base64'))
    fh.close()
    #GERA STRING JSON COM NUMERO DE MATRICULA
    data = '{"id_pres": "'+str(jData['id'])+'", "matricula": "'+str(matriculaTeste)+'"}'
    print(data)
    #PUBLICA NO TOPICO MQTT
    client.publish(topicPublish, data)
#-----------------------------------------------------------------------------#
#CONECTA AO BROKER MQTT INSCREVE NO TOPICO E FICA EM LOOP
client = paho.Client(device)
client.on_message=on_message
client.connect(broker)
client.loop_start()
client.subscribe(topicSubscribe)
while True:
    time.sleep(2)
#-----------------------------------------------------------------------------#
```

Fonte: elaborado pelo autor.

Figura 31 – Execução da aplicação



Fonte: acervo pessoal.

4.5 CONCLUSÕES SOBRE O CAPÍTULO

Com a implementação do *middleware* descrito nesta etapa do trabalho, cumpre-se uma série de requisitos, como independência e interoperabilidade entre dispositivos e aplicações, escalabilidade da rede do sistema IoT e a criação da ciência do contexto do ambiente. Por meio do MQTT e do *Node-RED,* obtém-se essa transparência entre os dispositivos e aplicações, que resulta na independência e interoperabilidade. Os fluxos do *Node-RED* e o *PostreSQL,* com sua lógica relacional, garantem a ciência de contexto do ambiente. Já a escalabilidade é de responsabilidade da infraestrutura do Google Compute Engine, que, por meio de seu gerenciador de VMs, permite a ampliação dos recursos da máquina conforme a rede de dispositivos e aplicações se expande.

O caso de teste aqui exposto evidencia que esses requisitos estão sendo atendidos. O *Raspberry Pi* atua como o dispositivo que captura dados do ambiente e os publica no *Mosquitto*. O *Node-RED* recebe os dados, armazenando-os e transmitindo-os à aplicação, que receberá as informações e retornará a identificação

do usuário a um tópico MQTT. Assim, finaliza-se o processo com o *Node-RED* armazenado e relacionando todas as informações em questão no *PostgreSQL*.

5 CONCLUSÃO

Ao longo deste trabalho, analisou-se como a Internet das Coisas se relaciona com dispositivos e aplicações e o que caracteriza essa comunicação cooperativa – mas independente. Tais questões foram associadas a conceitos referentes à computação ubíqua, que trata da mobilidade e onipresença dos dispositivos em nosso ambiente e cotidiano, e à ideia de Inteligência do Ambiente, que torna, justamente, os ambientes proativos, por meio de sensores e atuadores que utilizam do processamento de informações coletadas e fornecem serviços para otimização de tarefas cotidianas.

Uma camada de *middleware* vai ao encontro dos conceitos abordados, pois viabiliza a normalização da comunicação entre diferentes dispositivos e aplicações. O *middleware* fornece o ambiente digital necessário para que os dispositivos apenas estejam disponíveis para enviar informações referentes a seus sensores ou receber comandos para seus atuadores, e também para que as aplicações consigam receber os dados em tempo real ou posteriormente, considerando a persistência destes dados.

O *middleware* implementado neste projeto foi desenvolvido com base em eventos, ou seja, de forma que todos os dispositivos e aplicações comuniquem-se por meio de publicações e inscrições em tópicos MQTT. Ele armazena e processa essas informações por meio do *Node-RED* e do *PostgreSQL*, cumprindo os requisitos funcionais de gerência de recursos e eventos e os requisitos não funcionais de escalabilidade, *real-time*, confiabilidade, disponibilidade, segurança e privacidade.

5.1 CONTRIBUIÇÕES

O trabalho realizado evoca a importância da utilização de uma camada de *middleware* para implantação de ecossistemas IoT, em que dispositivos e aplicações funcionem de forma independente e, ao mesmo tempo, interoperável.

Uma aplicação para o reconhecimento facial de alunos em uma sala de aula foi utilizada para a implantação dessa camada e de sua configuração. Assim, recebem-se as informações dos dispositivos instalados e notifica-se a aplicação de reconhecimento facial com as informações necessárias, para que se efetue o

reconhecimento do aluno e se retorne essa identificação. Todas essas informações sendo armazenadas e relacionadas para gerar contexto ao ambiente.

A utilização de uma camada de *middleware*, portanto, agiliza a implantação de sistemas IoT, pois faz com que a comunicação entre *hardware* e *softwares* seja facilitada e menos onerosa, além de proporcionar o ambiente necessário para que dispositivos e aplicações trabalhem focados em suas especificações de tarefas.

5.2 TRABALHOS FUTUROS

Utilizando o que foi implementado neste trabalho, é possível desenvolver, futuramente, outros fluxos através do *Node-RED* e relações de contexto com o *PostgreSQL* para incrementar o ecossistema IoT com outros casos de uso.

Outras possíveis implementações são a aplicação de outras tecnologias à camada de *middleware*, para garantir a segurança e sigilo dos dados trafegados e armazenados, e a gerência remota de dispositivos e repositório de código, para que os dispositivos busquem suas configurações e atualizações no próprio *middleware*.

REFERÊNCIAS

ARAUJO, Regina Borges de. Computação Ubíqua: Princípios, Tecnologias e Desafios. **XXI Simpósio Brasileiro de Redes de Computadores**, São Carlos, p. 45-115, maio 2003.

ASHTON, Kevin. Internet das Coisas, nova revolução da conectividade. **Inovação em Pauta:** uma publicação da Finep, [s.l.], n. 18, p. 4-7, dez. 2014.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The Internet of Things: A survey. **Computer Networks**, [s.l.], p.1-19, maio 2010.

BAKKEN, David E. **Middleware.** [s.l.], mar. 2001. Disponível em: < https://www.researchgate.net/publication/2351828_Middleware>. Acesso em: 11 abr. 2018.

BARROS, Marcelo. MQTT – **Protocolos para IoT**. [s.l.], jun 2015. Disponível em: < https://www.embarcados.com.br/mqtt-protocolos-para-iot/>. Acesso em: 15 maio 2018.

CÁCERES, Ramón; FRIDAY, Adrian. Ubicomp Systems at 20: Progress, Opportunities, and Challenges. **IEEE Pervasive Computing**, [s.l.], v. 11, p. 14-21, dez. 2011.

CARISSIMI, Alexandre. **Internet das Coisas, middlewares e outras coisas**. Porto Alegre: SBC, 2016.

CASAGRAS. **Final Report:** RFID and the Inclusive Model for the Internet of Things. [S.l.], 2009.

CHAVES,Rodrigo Reuse. **Redes Neurais Deep Learning Aplicadas ao Reconhecimento Facial.** (no prelo). Trabalho de Conclusão de Curso (Graduação) – Universidade de Caxias do Sul, Campus Universitário de Caxias do Sul. Bacharelado em Ciências da Computação, 2018.

COMPUTERWORLD. **Mercado de middleware tem crescimento de 12% e chegará a US$ 30 bi em 2018.** 2018. Disponível em: <http://computerworld.com.br/mercado-de-middleware-tem-crescimento-de-12-e-chegara-us-30-bi-em-2018>. Acesso em: 11 jun. 2018.

COOK, Diane J.; AUGUSTO, Juan C.; JAKKULAA, Vikramaditya R. Ambient intelligence: Technologies, applications, and opportunities. **Pervasive and mobile computing**, [s.l.], v. 5, p. 277-298, ago. 2009.

DIAS, Renata Rampim de Freitas. **Internet das Coisas sem mistérios**: uma nova inteligência para os negócios. São Paulo: Netpress Books, 2016.

FERREIRA, Hiro Gabriel Cerqueira. **Arquitetura de Middleware para Internet das Coisas**. 2014. 125 f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade de Brasília, Brasília, 2014.

GUBBI, Jayavardhana et al. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems** , [s.l.], v. 29, p. 1645-1660, set. 2013.

ISTAG. Information Society Technologies Advisory Group. **Scenarios for Ambient Intelligence in 2010**. Sevilha: IPTS, 2001.

ITU-T. Telecommunication Standardization Sector of ITU. **Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks**: Next Generation Networks? Frameworks and functional architecture models. 2012.

LÓPEZ, Tomas Sánchez. **What the Internet of Things is NOT**. 2010. Disponível em: <http://technicaltoplus.blogspot.com/2010/03/what-internet-of-things-is-not.html>. Acesso em: 09 maio 2018.

NODE-RED: Flow-based programming for the Internet of Things. Disponível em: < https://nodered.org/>. Acesso em: 12 maio 2018.

OASIS. Advancing Open Standards for the Information Society. **Open Protocols for na Open, Interoperable Internet of Things**. Geneva, Switzerland, fev. 2014.

PIRES, Paulo F. et al. **Plataformas para a Internet das Coisas**. 2015. XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), Espiro Santo – Vitória, 2015.

RAZZAQUE, Mohammad Abdur et al. Middleware for Internet of Things: A Survey. **IEEE Internet of Things Journal**, [S.l.], v. 3, p. 70-95, fev. 2016.

SCHUSTER, Alfons. **Intelligent Computing Everywhere**. Londres: Springer, 2007.

SILVA, Everton et al. **Computação Ubíqua – Definição e Exemplos**. **Revista de Empreendedorismo, Inovação e Tecnologia**, Alegrete 23-32, 2015 - ISSN 2359-3539.

SUZIN, Matheus Zimmermann. **Ambient Intelligence na Educação: Desenvolvimento de uma Sala de Aula Inteligente utilizando IoT.** (no prelo). Trabalho de Conclusão de Curso (Graduação) – Universidade de Caxias do Sul, Campus Universitário de Caxias do Sul. Bacharelado em Engenharia de Controle e Automação, 2018.

WEISER, Mark. The Computer for the 21st Century. **Scientific American**, p. 94-104, set. 1991.

WOOTTON, George. **Internet Das Coisas**: uma visão ampla, humana e livre. [S.l.]: Clube de Autores, 2016.