UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Vitor Hugo Nascimento Pinto

Criação de interfaces de controle de IoT através do Node-RED

Juiz de Fora
2021

Vitor Hugo Nascimento Pinto

Criação de interfaces de controle de IoT através do Node-RED

Trabalho de conclusão de curso apresentado ao Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em 03 de Setembro de 2021

BANCA EXAMINADORA

Prof. Dr. Eduardo Pagani Julio - Orientador
Universidade Federal de Juiz de Fora

Prof Dr. Eduardo Barrere
Universidade Federal de Juiz de Fora

Prof Dr.a Luciana Conceição Dias Campos
Universidade Federal de Juiz de Fora

A minha família que sempre foi a base de tudo que conquistei.

Aos meus amigos e mentores, que sempre estiveram comigo nessa caminhada.

RESUMO

Com o crescimento no número de dispositivos de *Internet* das Coisas (*Internet of Things - IoT*), surgem novos desafios, tais como a entrada das conexões 5G nos novos aparelhos, a necessidade de um controle mais imediato, entre outros. Esses se somam aos já existentes no âmbito da IoT, como a quantidade de dispositivos já disponíveis ou na complexidade da implementação de uma *interface* para controle. Portanto, a necessidade de uma análise mais profunda sobre os modelos de criação de *interfaces* de controle de IoT é de extrema importância e necessidade. Dentre estas *interfaces*, este trabalho tem como objetivo a análise em específico sobre o Node-RED, que conta com diversas funcionalidades em sua implementação, que facilitam, trazem ganhos de desempenho e melhoram a interação entre aparelhos na criação de *interfaces* de controle de IoT e o como esta *interface* consegue trazer esses recursos para o usuário, buscando demonstrar seus pontos fortes, exemplificando a facilidade de sua implementação quando comparado a outras plataformas e a facilidade de utilizações de modelos complexos, tal como o uso de aprendizado de máquina nos fluxos de IoT.

Palavras-chave: Node-Red. Internet of Things. IoT. Interfaces de Controle de IoT.

ABSTRACT

With the growth in the number of Internet of Things (IoT) devices, new challenges arise, such as the introduction of 5G connections in new devices, the need for more immediate control, among others. These are added to those already existing in the IoT scope, such as the number of devices already available or the complexity of implementing a control interface. Therefore, the need for a deeper analysis on the models for creating IoT control interfaces is of utmost importance and necessity. Among these interfaces, this work will address the specific analysis of Node-RED, which has several features in its implementation, which facilitate, bring performance gains and improve the interaction between devices in the creation of IoT control interfaces and how this interface manages to bring these features to the user, seeking to demonstrate its strengths, exemplifying the ease of its implementation when compared to other platforms and the ease of using complex models, such as the use of machine learning in IoT flows.

Keywords: Node-Red. Internet of Things. IoT. IoT Control Interfaces.

RESUMEN

Con el crecimiento en la cantidad de dispositivos *Internet* de las Cosas (*Internet of Things - IoT*), surgen nuevos desafíos, como la entrada de conexiones 5G en nuevos dispositivos, la necesidad de un control más inmediato, entre otros. Estos se suman a los que ya existen en el ámbito de IoT, como la gran cantidad de dispositivos que ya están disponibles o la complejidad de implementar una *interface* para el control. Por lo tanto, la necesidad de un análisis más profundo de los modelos de creación de control *interfaces* de IoT es de extrema importancia y necesidad. Entre estas *interfaces*, este trabajo abordará el análisis específico de Node-RED, que tiene varias características en su implementación, que facilitan, aportan ganancias de rendimiento y mejoran la interacción entre dispositivos en la creación de *interfaces* de El control de IoT y cómo esta *interface* puede traer estas características al usuario, buscando demostrar sus fortalezas, ejemplificando la facilidad de implementación en comparación con otras plataformas y la facilidad de uso de modelos complejos, como el uso del aprendizaje automático. en los flujos de IoT.

Palabras clave: Node-Red. Internet of Things. IoT. Interfaces de control de IoT.

LISTA DE ILUSTRAÇÕES

Internet of Things (COMTECH, 2016) 13
Plataforma IoT (AVSystem, 2020) 15
Node-RED 16
Interface de desenvolvimento do Node-RED 17
Amazon Web Services 17
Azure IoT *Hub* 18
Fluxograma MQTT (Fernandes, 2021) 19
fluxo da implementação básica através do Node-RED 21
Código necessário para manipular os dados recebidos do Node-RED 22
Configuração para dividir os tipos de fluxo no Node-RED 22
Requisição para API do openweathermap 23
Template com as informações de temperatura 24
Envio de email através do Node-RED 24
Painel de Controle Node-RED 25
Criação da política de acesso na AWS 26
Endpoint do Topico MQTT na AWS 27
Configuração de Envio de mensagem MQTT a AWS através do Node-RED 27
Recebimento da Mensagem no tópico MQTT na AWS 28
Criação do tópico SNS na AWS 29
Criação da assinatura SNS na AWS 29
Criação da regra de envio de *e-mail* na AWS 30
Criação da ação de envio de *e-mail* via SNS na AWS 30
Criação do *Hub* IoT na Azure 31
Criação do dispositivo de IoT na Azure 32
Inserindo dados no *Device Explorer Twin* 33
Gerando chave de acesso ao dispositivo no *Device Explorer Twin* 33
Configuração de Envio de mensagem MQTT à Azure através do Node-RED 34
TensorFlow.js 36
Função de pré processamento para o TensorFlow 37
Configuração do Modelo para detecção de objetos 38
Configuração do pós-processamento para o TensorFlow 38
fluxo da implementação do detector de objetos através do Node-RED 39

LISTA DE TABELAS

Tabela 1 – Comparativo entre as implementações 35

LISTA DE ABREVIATURAS E SIGLAS

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
IoT	*Internet of Things*
IBGE	Instituto Brasileiro de Geografia e Estatística
FGVcia	Centro de Tecnologia de Informação Aplicada
IA	*Inteligência Artificial*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
TSL	*Transport Layer Security*
IOsL	*Internet of Smart Living*
IOsC	*Internet of Smart Cities*
IOsE	*Internet of Smart Environment*
IOsI	*Internet of Smart Industry*
IOsH	*Internet of Smart Health*
IOsE	*Internet of Smart Energy*
IOsA	*Internet of Smart Agriculture*
JSON	*JavaScript Object Notation*
XML	*eXtensible Markup Language*
HTTPS	*Hyper Text Transfer Protocol Secure*
MQTT	*Message Queuing Telemetry Transport*
API	*Application Programming Interface*
AWS	Amazon *Web Services*
ARN	Amazon *Resource Name*
SQL	*Structured Query Language*
SNS	Amazon *Simple Notification Service*
IDE	*Integrated Development Environment*

SUMÁRIO

1	**INTRODUÇÃO**	**10**
1.1	DESCRIÇÃO DO PROBLEMA	10
1.2	JUSTIFICATIVA	11
1.3	OBJETIVOS	12
2	**REFERENCIAL TEÓRICO**	**13**
2.1	O QUE É IOT	13
2.1.1	CONCEITO	13
2.1.2	DEFINIÇÃO	13
2.1.3	CARACTERÍSTICAS PRINCIPAIS	14
2.1.4	ÁREAS DE APLICAÇÃO	14
2.2	PLATAFORMAS DE INTERNET DAS COISAS	15
2.2.1	O QUE SÃO AS PLATAFORMAS	15
2.2.2	PROBLEMAS ATUAIS	15
2.2.3	NODE-RED	16
2.2.4	AMAZON WEB SERVICES	17
2.2.5	AZURE IOT HUB	18
2.3	PROTOCOLO MQTT	18
2.3.1	COMO FUNCIONA	18
3	**ESTUDO DE CASO**	**20**
3.1	IMPLEMENTAÇÃO BÁSICA DE UMA INTERFACE DE IOT	20
3.1.1	PADRONIZANDO AS IMPLEMENTAÇÕES POR MEIO DO MQTT	20
3.1.2	IMPLEMENTAÇÃO VIA NODE-RED	20
3.1.3	IMPLEMENTAÇÃO VIA AWS IOT	26
3.1.4	IMPLEMENTAÇÃO NA AZURE IOT HUB	31
3.1.5	CONCLUSÃO	35
3.2	IMPLEMENTAÇÃO DE INTERFACE ALIADA COM MACHINE LEARNING	36
3.2.1	O QUE É TENSORFLOW.JS	36
3.2.2	IMPLEMENTAÇÃO	36
4	**CONCLUSÃO E TRABALHOS FUTUROS**	**40**
	REFERÊNCIAS	**42**
	Referências Bibliográficas	**42**

1 INTRODUÇÃO

Com a evolução obtida nas áreas tecnológicas nas décadas finais do século XX devido à evolução na capacidade de processamento e eficiência dos equipamentos, se tornou cada vez mais viável a implementação de tecnologias em coisas[1] portáteis e não mais apenas só em computadores. Ao longos dos anos a criação e distribuição destes dispositivos portateis vem se tornando cada vez maior conforme Heuveldop (2017).

A criação e distribuição se tornou tão expressiva que ao se levar a realidade brasileira como exemplo, existem atualmente 213 milhões de habitantes (IBGE, 2021) e 440 milhões de dispositivos móveis (Meirelles, 2021) no território brasileiro. Analisando estes números superficialmente, é visto que em média, existem dois aparelhos móveis para cada habitante do país, quando trazemos outros dispositivos para essa realidade, tais como sensores, câmeras, esse número se torna cada vez mais expressivo.

Tendo em vista essa expansão da área de *Internet* das Coisas (*Internet of Things - IoT*), no qual é nitido um crescimento no número dispositivos, verifica-se a cada evolução de geração destes, um maior crescimento da tecnologia embarcada e um aumento da gama de aplicações em que são empregados.

Existem diversas aplicações que possibilitam a manipulação, configuração e controle desses dispositivos, o que resulta em uma vultosa gama de possibilidades. O objetivo deste trabalho é trazer o destaque a um desses controladores, o Node-RED, de modo a defini-lo como padronizador para essas configurações, demonstrando os benefícios de sua aplicação, sejam estes na curva de aprendizado necessário para a utilização da ferramenta, facilidade de obtenção de informações sobre possíveis implementações ou até mesmo na leitura e escrita dos dados dos sensores.

1.1 DESCRIÇÃO DO PROBLEMA

A IoT é um universo em expansão, onde segundo o relatório de mobilidade captados pela Ericsson em Heuveldop (2017) existe a previsão que entre 2019 – 2023 aconteça um crescimento anual composto de 19% no número de dispositivos IoT conectados. Quando considera uma base em torno de 30 bilhões de dispositivos (Heuveldop, 2017) esse crescimento se mostra extremamente marcante.

O número de coisas conectadas são cada vez maiores, e variam desde computadores, até carros. A IoT também traz diversos benefícios e conforme o relatório da CompTIA (2020) e os cinco principais são: economia de custos de eficiência operacional; novos e melhores fluxos de dados para melhorar a tomada de decisões; ganhos de produtividade; melhor visibilidade e monitoramento de pessoas, serviços, produto; novas e melhores

[1] É definido como tudo que pode ser utilizado como objeto no emprego da IoT

experiencias dos clientes.

GrowthEnabler (2017) estuda o crescente uso e disponibilidade de aparelhos de IoT. Com esse crescimento a configuração, coleta e processamento de informações desses aparelhos se tornou um desafio predominante no âmbito global.

Unido a esse crescimento, o impulso da tecnologia também forneceu uma base para tornar os sistemas de automação mais complexos em diferentes aspectos, tal seja no número de dispositivos e pontos de dados, interação entre estes dispositivos, requisitos de segurança, integração dos sistemas, entre outros.

Uma vez que aparelhos de IoT não se restringem apenas a sensores ou aparelhos específicos, onde câmeras, veículos, roupas, eletrodomésticos, infraestruturas e outros, também podem ser utilizados como aparelhos, é observado uma enorme gama de aparelhos que podem ser utilizados nos fluxos de IoT.

Além disso, novas técnicas de *Machine Learning* e Inteligência Artificial (IA) revelam uma gama expressiva de atuação sobre novas modalidades de interação, como reconhecimento de voz, realidade aumentada, assistentes virtuais e robótica, conforme demonstrado no estudo de Groopman (2020), corroborando para que haja um crescimento tanto em números de dispositivos, quanto em complexidade de soluções.

Portanto, ao longo dos anos, diversos modelos diferentes evoluíram, em linha com os fundamentos da tecnologia em progresso e surgiram inúmeras *interfaces* para criação de controladores de IoT conforme cita Jasperneite; Sauter; Wollschlaeger (2020).

Essas interfaces de criação devem ser avaliadas e comparadas, para que se possa ter um estudo sobre as possíveis implementações e seja viável entender os controladores atuais e buscar a melhor ferramenta de controle de IoT disponível atualmente.

1.2 JUSTIFICATIVA

É notório que atualmente é necessário considerar o alto número de formas e *interfaces* para criação de controle de IoT e também a necessidade de análise dessas para que se defina qual a melhor ferramenta para uso.

Uma implementação que se destaca entre as outras é o Node-RED, uma *interface* que se sobressai em diversas áreas, visto que permite uma criação de forma simples, isso ocorre devido à:

- Sua implementação em *JavaScript*, que é a linguagem que possui o maior ecossistema de componentes de código aberto e também uma comunidade muito madura.
- A análise de seus dados, que é feita sobre objetos *JavaScript Object Notation* (JSON) e não de *eXtensible Markup Language* (XML), o que os torna mais fáceis de serem analisados e mantidos.

- A visualização do desenvolvimento baseada em fluxo, que permite uma programação mais visual, por conseguinte, mais simples para leigos e de mais fácil entendimento conforme dito por Perry (2017).
- A possibilidade de atuar com equipamentos físicos de baixo custo e com uma implementação que pode ser executada localmente com uma solução elegante, para unir diferentes dispositivos de IoT e serviços conforme cita OpenJS Foundation (2020).
- Sua forma de desenvolvimento, sendo uma ferramenta que possibilita a integração entre dispositivos de formas novas e permite uma implementação de forma simples e direta.

Portanto, será analisado a implementação da criação de *interfaces* utilizando essa ferramenta, buscando demonstrar o funcionamento e os motivos para o uso do Node-RED como principal forma da criação de *interfaces* de controle de IoT.

1.3 OBJETIVOS

Este trabalho visa estudar como objetivo principal a forma de criação de uma *interface* de controle de IoT, utilizando o Node-RED, de modo a oferecer uma demonstração do funcionamento e dos motivos desse ter sido estudado e apontado como a melhor ferramenta para criação de *interfaces*, esses motivos serão exemplificados e comparados com outros modelos de modo a embasar as afirmações.

Além disso, tem como objetivos secundários demonstrar como ela auxilia no controle de dispositivos e também analisar a implementação do mesmo com auxílio de alguma inteligência externa e de como poderia ser conduzida essa integração.

2 REFERENCIAL TEÓRICO

2.1 O QUE É IOT

Antes que seja introduzido o conceito relativo às *interfaces* de controle de IoT, é necessário entender mais a fundo sobre o conceito de IoT para ser possível assimilar mais claramente a necessidade dessas *interfaces*.

2.1.1 CONCEITO

O Conceito de IoT foi criado por Kevin Ashton (Schultz, 2020), um pioneiro tecnológico britânico no ano de 1999 e veio se tornando cada vez mais relevante globalmente.

Pressuponha um mundo onde existam bilhões de objetos e que cada um deles possa sentir, comunicar e compartilhar informações, tudo isso de forma interconectada através de redes públicas e privadas. Esses objetos geram dados que quando analisados podem ser usados de diversas formas, fornecendo uma riqueza de detalhes e trazendo um ganho expressivo de inteligência, seja essa usada para planejamento, gerenciamento ou até mesmo nas tomadas de decisões. Esse é o mundo da IoT.

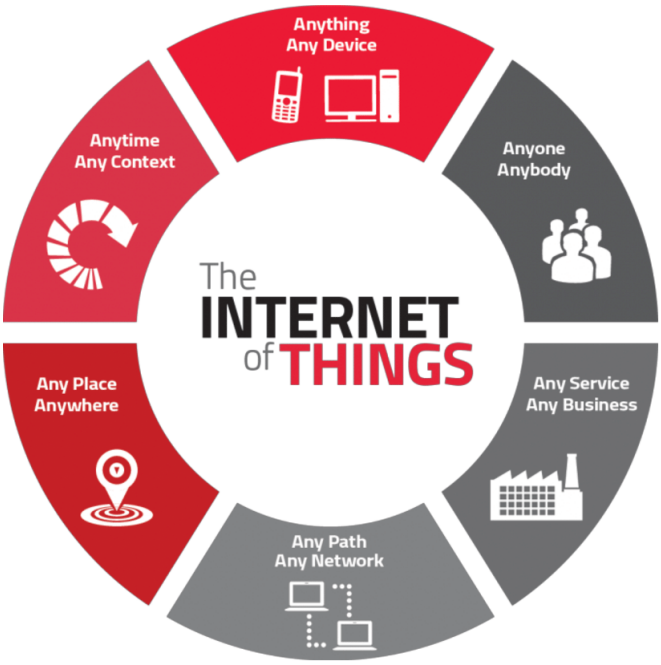

Figura 1: *Internet of Things* (COMTECH, 2016)

2.1.2 DEFINIÇÃO

IoT é uma rede de objetos físicos, não apenas uma rede de computadores. Ela evoluiu para um universo de dispositivos de todos os tipos e tamanhos, tal como, veículos, celulares, eletrodomésticos, brinquedos, câmeras, instrumentos médicos, etc. Todos estes

conectados e compartilhando informações entre si, de modo a alcançar reorganizações inteligentes, posicionamento, rastreamento, segurança e controle, controles de processo, administração e até mesmo rastreamento em tempo real vide Patel et al (2016).

2.1.3 CARACTERÍSTICAS PRINCIPAIS

As características principais em uma rede de IoT são:

- **Interconectividade:** Qualquer coisa pode ser interligada com a informação global e estrutura de comunicação.
- **Heterogeneidade:** Os dispositivos devem conseguir interagir entre si sem problemas, mesmo sendo diferentes.
- **Mudanças dinâmicas:** Os estados dos dispositivos mudam a todo instante, seja acordando ou dormindo, conectando e desconectando, localização, velocidade, etc.
- **Escalabilidade:** Cada vez existem mais dispositivos, e estes, em união com seus dados, devem ser manejados de forma eficiente.
- **Segurança:** Com os benefícios da IoT é necessário tratar de forma segura todo trafego de dados, seja esse pessoal ou não e sua manipulação.
- **Conectividade:** Deve ser possível conectar a rede e a partir de uma conexão, consumir e produzir dados.

2.1.4 ÁREAS DE APLICAÇÃO

Existem infinitas aplicações para IoT atualmente. Todavia, algumas se destacam e se mostram extremamente importantes para melhoria e evolução de áreas básicas da sociedade, que variam em camadas pessoais e econômicas, que segundo Aker (2019) são:

- Internet de Vida Inteligente - Internet of Smart Living (IOsL)
- Internet de Cidades Inteligentes - Internet of Smart Cities (IOsC)
- Internet de Ambientes Inteligentes - Internet of Smart Environment (IOsE)
- Internet de Indústrias Inteligentes - Internet of Smart Industry (IOsI)
- Internet de Saúde Inteligente - Internet of Smart Health (IOsH)
- Internet de Energia Inteligente - Internet of Smart Energy (IOsE)
- Internet de Agricultura Inteligente - Internet of Smart Agriculture (IOsA)

2.2 PLATAFORMAS DE INTERNET DAS COISAS

2.2.1 O QUE SÃO AS PLATAFORMAS

As Plataformas IoT são responsáveis pela implantação da IoT. É dever delas reunir todas as camadas de equipamento físico, conectividade, software e aplicativo para oferecer uma solução eficiente para o gerenciamento e configuração dos dispositivos, coleta e análise de dados e como será feita a conexão desses dispositivos, conforme AVSystem (2020). Notadamente, as plataformas IoT são necessárias para colocar todo o conceito de IoT em prática.

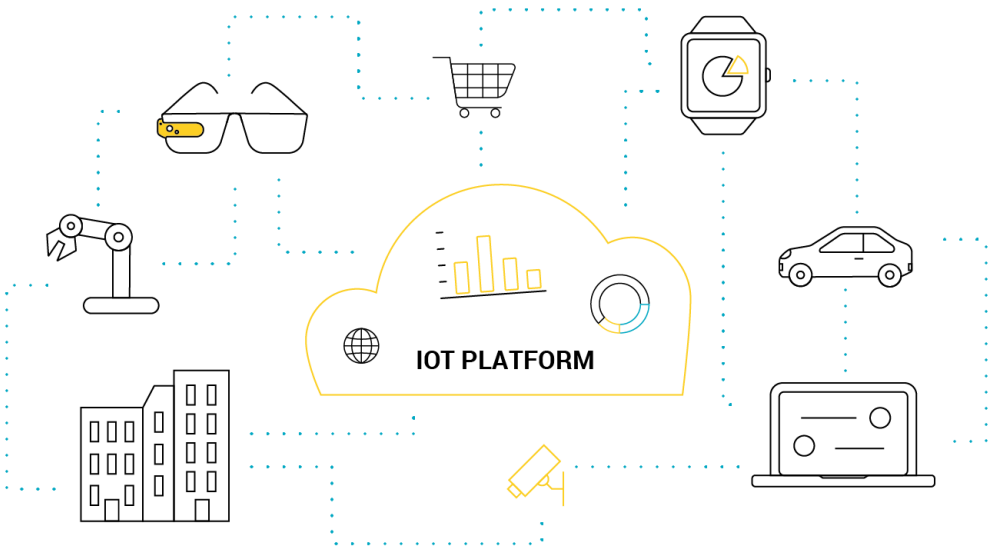

Figura 2: Plataforma IoT (AVSystem, 2020)

2.2.2 PROBLEMAS ATUAIS

Devido ao crescente interesse em IoT, o número de plataformas projetadas para oferecer suporte a IoT aumentou consideravelmente. Aliado a isso, “altas quantias de financiamento estão sendo implantadas em *startups* de IoT e devido a isso, o foco da indústria tem sido a fabricação e produção dos tipos certos de equipamentos físicos para viabilizar essas soluções”, conforme Banafa (2016).

Como resultado de diferentes abordagens, padrões e casos de uso, há uma vultosa variedade e heterogeneidade de plataformas de IoT. Isso leva a dificuldades em compreender, selecionar e usar as plataformas adequadas vide Guth et al (2018). A medida que a indústria evolui, a necessidade de um modelo padrão para realizar tarefas comuns de IoT, como processamento, armazenamento e atualizações de firmware, está se tornando cada vez mais relevante.

2.2.3 NODE-RED

Figura 3: Node-RED

Uma solução para a resolução do problema em criar uma padronização sobre as inúmeras plataformas de IoT é a utilização do Node-RED. O motivo dessa escolha é devido ao fato de esse ser uma plataforma que atua de forma mais completa em relação às outras, seja no que se refere a sua *interface* mais amigável ao usuário, no formato que seus dados atuam ou algum outro ponto de sua utilização. Isso será demonstrado mais a fundo no estudo de caso e na comparação com outras plataformas.

Simpkin et al (2020) cita que "o Node-RED oferece uma *interface* de usuário atraente, baseada na *Web* para executar fluxos de trabalho embasados em serviços de IoT". Como a maioria das aplicações, ele coordena seu fluxo de trabalho de forma central, o dificultando de ser executado em ambiente onde os nós são móveis.

Contudo, ele tem um diferencial, visto que pode ter sua utilização migrada para um ambiente de execução descentralizado para operação em redes móveis utilizando uma arquitetura para converter dinamicamente aplicativos Node-RED em representações de vetor semântico (Simpkin et al, 2020).

No artigo, Simpkin et al (2020) exemplifica como essa implementação é feita e se aprofunda mais nessa abordagem de utilização do Node-RED em ambientes descentralizados. O Node-RED também permite a execução da aplicação gerada, mesmo não estando em ambiente de produção.

Com isso, é possível identificar problemas no fluxo gerado pela abordagem, problemas de configurações e falhas na arquitetura segundo os estudos de Welter (2019).

Por se tratar de uma *interface drag-and-drop* que utiliza o conceito de nós, a Figura 4 demonstra um exemplo da *interface* do Node-RED onde cada um dos blocos é um nó e cada um deles tem uma finalidade. Ele permite que a reutilização de código seja feita de forma fácil e intuitiva e também facilita a criação de código vide Calabrez (2019).

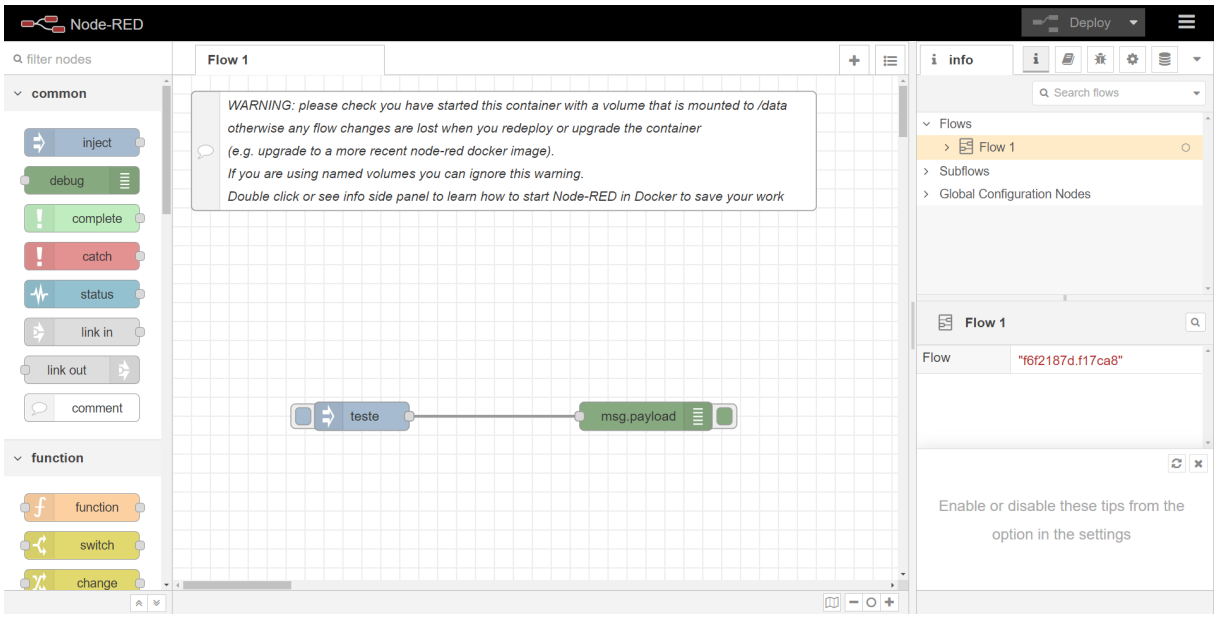

Figura 4: Interface de desenvolvimento do Node-RED

2.2.4 AMAZON WEB SERVICES

Figura 5: Amazon Web Services

É uma das plataformas de IoT mais completas do mercado atualmente. A Amazon *Web Services* (AWS) fornece serviços de nuvem para implementar soluções de IoT. O AWS IoT Core é um serviço de nuvem gerenciada que permite que dispositivos conectados interajam com segurança com aplicativos de nuvem e outros dispositivos. AWS IoT Core pode suportar muitos dispositivos e mensagens, e pode processar e encaminhar essas mensagens para endpoints e outros dispositivos conforme documentação da Amazon (2021).

2.2.5 AZURE IOT HUB

Figura 6: Azure IoT *Hub*

Uma das primeiras plataformas de IoT a surgir e uma das mais completas, o Azure IoT *Hub* possui uma enorme gama de recursos, indo de *dashboards* até sistemas complexos de inteligência artificial conforme Bertoleti (2020). Ela oferece uma solução de *back-end* hospedada na nuvem, que permite uma conexão virtual a qualquer dispositivo, habilitando uma comunicação segura e confiável entre o aplicativo IoT e os dispositivos gerenciados por ele, conforme a documentação da Microsoft (2021).

2.3 PROTOCOLO MQTT

MQTT é um protocolo de mensagens padrão para a IoT. Ele foi projetado como um transporte de mensagens de publicação e assinatura extremamente leve, ideal para conectar dispositivos remotos com uma pequena área de cobertura de código e largura de banda de rede mínima. Ele é usado em uma ampla variedade de indústrias, como automotiva, manufatura, telecomunicações, petróleo e gás, entre outras (MQTT.org, 2020).

2.3.1 COMO FUNCIONA

O Protocolo MQTT tem três tipos de entidades, que são nomeados como: *publisher*, *broker* e *subscriber*. O *Publisher* tem a responsabilidade de enviar a mensagem para o *broker*. Por conseguinte, o *broker* é responsável por receber todas as mensagem dos *publishers* e enviá-las para seus *subscribers* relevantes conforme Yuan (2017). É definido como *Subscriber* qualquer coisa que possa interagir com o *broker* e receber mensagens. Por exemplo, ele pode ser um sensor de IoT ou um aplicativo que processa dados de IoT. O fluxo funciona da seguinte forma:

1. O cliente se conecta ao *broker*. Essa conexão é realizada através da assinatura de um tópico de mensagem no *broker*. Essa conexão pode ser feita através de TCP/IP ou até mesmo via *Transport Layer Security* (TLS).

2. A partir da conexão, o cliente publica as mensagens no tópico e o tópico envia ao *broker*.

3. Por fim o *broker* encaminha a mensagem a todos os clientes que assinam o tópico.

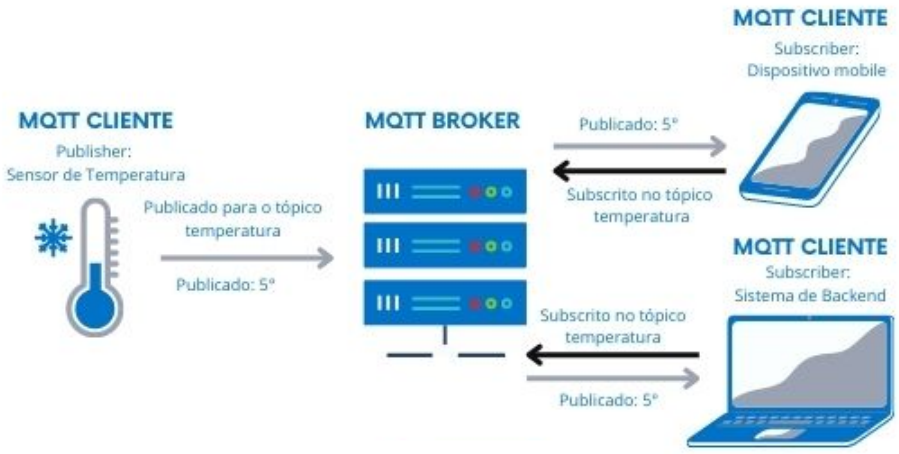

Figura 7: Fluxograma MQTT (Fernandes, 2021)

A Figura 7 é uma demonstração visual do fluxo explicado acima.

3 ESTUDO DE CASO

3.1 IMPLEMENTAÇÃO BÁSICA DE UMA INTERFACE DE IOT

Será criada uma *interface* de IoT para demonstrar o funcionamento de um recebimento de uma mensagem via protocolo MQTT. A partir da leitura dessa mensagem será realizado uma manipulação dos dados, uma busca em uma *Application Programming Interface* (API) externa com os dados manipulados e uma tratativa deste retorno. O resultado será demonstrado através do envio de um *e-mail* com os dados retornados.

O motivo da utilização desse exemplo é permear pontos que estão presentes na maioria das implementações de *interfaces*, portanto abrangendo esses pontos, são demonstradas possíveis soluções de partes do fluxo de implementação que, possivelmente, existirão na maioria dos modelos a serem desenvolvidos.

Serão utilizados além do Node-RED, o AWS IoT e o Azure IoT *Hub*, que são duas *interfaces* utilizadas de forma majoritária na criação de modelos de controle, conforme Rupareliya (2021) e Santos (2019), buscando trazer uma demonstração das diferentes dificuldades de implementação em diferentes *interfaces*.

3.1.1 PADRONIZANDO AS IMPLEMENTAÇÕES POR MEIO DO MQTT

Será utilizado MQTT para que seja possível ter uma padronização entre as implementações em diferentes *interfaces*. O propósito dessa utilização é seguir um fluxo comum de implementação em diferentes *interfaces*, deixando apenas em evidência as diferenças de codificação e desenvolvimento relativas a cada uma e não a fatores externos.

3.1.2 IMPLEMENTAÇÃO VIA NODE-RED

Para realizar a implementação do modelo básico através do Node-RED, será necessário a criação de uma *interface* de IoT aliada com o uso de bibliotecas externas, que podem ser integradas ao Node-RED devido a sua implementação em JavaScript. Isso permite a utilização de diversas bibliotecas pré prontas, sejam essas nativas, disponibilizadas por entusiastas ou empresas, assentindo em uma maior facilidade ao executar uma implementação, em virtude de possivelmente uma implementação similar já existir em forma de biblioteca podendo ser importada ao fluxo.

Para realizar a importação de bibliotecas no Node-RED existe um sistema de paleta, onde ficam disponíveis todos os módulos pré existentes na rede. Esses são criados pela comunidade para facilitar a implementação para aqueles que já desejam algo pronto ou não tenham conhecimento de como realizar tal implementação.

Portanto, basta que se procurem os módulos relacionados a necessidade da solução nas implementações a partir da paleta. Para a implementação da *interface* básica que foi

definida, será necessário a importação de três bibliotecas externas:

- **node-red-contrib-mqtt-broker:** É uma biblioteca que permite utilização do MQTT sem a necessidade de implementações via código, sendo possível executar tudo via *interface*.
- **node-red-dashboard:** É uma biblioteca que permite a criação de um painel de controle para visualização de informações em tempo real.
- **node-red-node-email:** É uma biblioteca que integra uma implementação para envios de *e-mail* de forma simples, sem a necessidade de codificação.

O fluxograma da Figura 8 demonstra o fluxo[1] completo implementado no Node-RED para realizar todas as tratativas e regras descritas que serão demonstradas a seguir. A maior parte será desenvolvida via *interface* e sem a necessidade de implementação externa, o que resulta em uma maior facilidade de aprendizado e utilização, como dito na fundamentação.

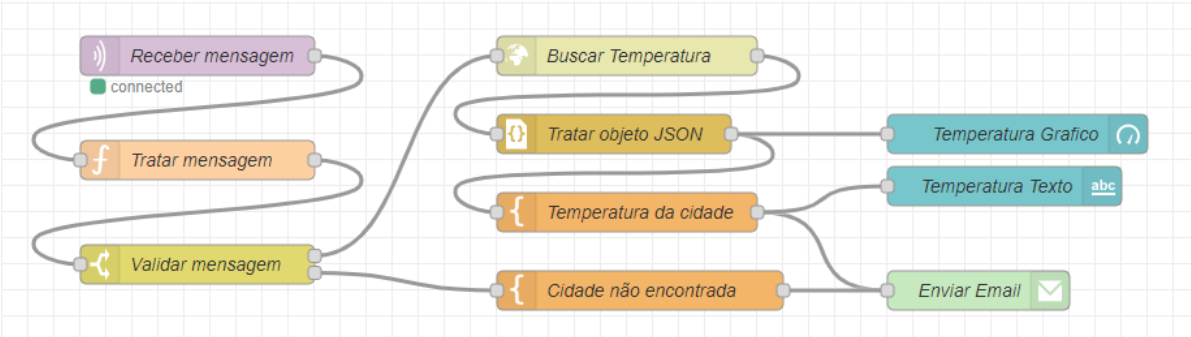

Figura 8: fluxo da implementação básica através do Node-RED

Primeiramente, é necessário receber os dados a partir da escuta do tópico do MQTT. Para isso, será utilizada a biblioteca *node-red-contrib-mqtt-broker* para criar um nó que será responsável pelo recebimento da mensagem do tópico. A biblioteca já tem por padrão todas as configurações básicas para recebimento da mensagem, sendo necessário ao usuário apenas informar qual o Servidor e o Tópico MQTT em que a mensagem será recebida, caso o usuário deseje informar um Servidor com mais informações, basta que estas sejam informadas nas configurações.

Com o recebimento da mensagem, é necessário realizar a manipulação dos dados a partir do *input* recebido, para essa manipulação não será necessária uma biblioteca externa, visto que o Node-RED dispõe de um pacote básico em que já permite a criação de funções de forma simples. Nesse momento é necessário a implementação de um pequeno trecho de código, mas como o Node-RED é implementado em *javascript*, conforme a Figura 9,

[1] `https://vitorhugo-files-upload.s3.sa-east-1.amazonaws.com/flow_basic.json`

essa implementação é bem simples e fácil de ser encontrada na rede, caso não se tenha os conhecimentos necessários.

Esse código tem a responsabilidade de receber um nome de uma cidade e definir qual o identificador relativo a este nome a aplicação externa espera em sua consulta, feita nos passos à frente.

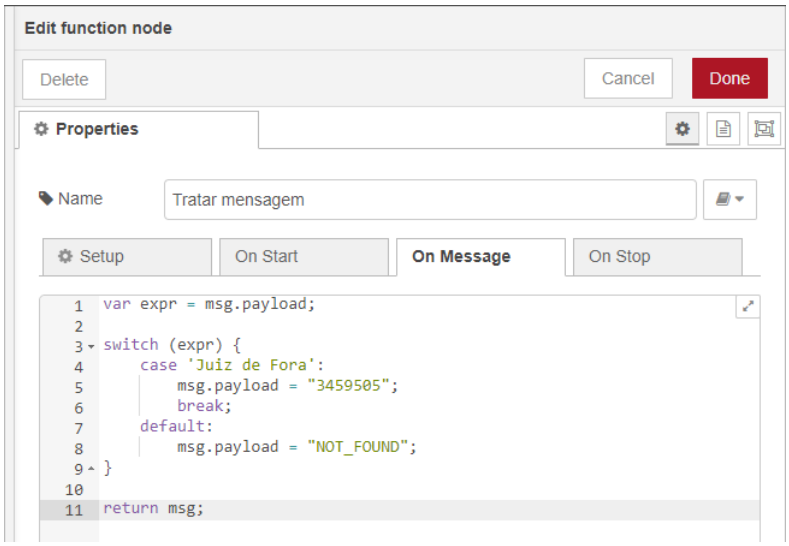

Figura 9: Código necessário para manipular os dados recebidos do Node-RED

Após a manipulação dessa mensagem, é necessário validar se a mensagem recebida realmente pôde ser identificada como um identificador na aplicação externa. Caso o nome da cidade recebido não tenha nenhum identificador, não se deve seguir o fluxo de busca na aplicação externa.

Para essa Validação também será utilizado o pacote básico do Node-RED, que permite uma validação simples, onde não necessita de nenhuma implementação de código. Tudo pode ser configurado via *interface*, como ilustrado na Figura 10.

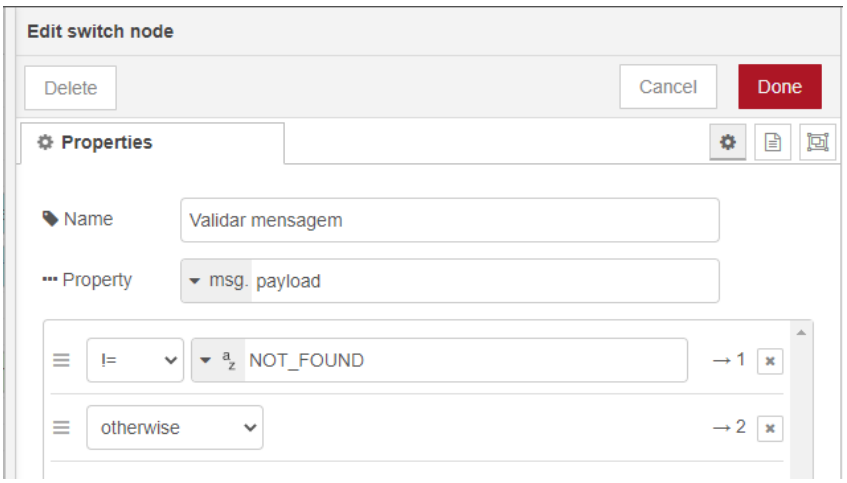

Figura 10: Configuração para dividir os tipos de fluxo no Node-RED

Nesse momento, acontece uma divisão em dois fluxos distintos, o fluxo de sucesso, onde a cidade pode ser identificada na aplicação externa e o fluxo de falha, onde não foi possível fazer essa identificação. Portanto, o fluxo de implementação será dividido em dois momentos.

- Fluxo de Sucesso

  Com o identificador da cidade mapeado, será utilizado novamente o pacote básico do Node-RED, que permite a realização de uma chamada HTTPS a uma aplicação externa.

  Esse desenvolvimento é feito via *interface* e também não necessita da implementação de nenhum trecho de código. Será realizada uma consulta à aplicação do openweathermap[2], que disponibiliza informações sobre as temperaturas de diversas cidades em tempo real e, para essa consulta deve-se apenas passar a rota da chamada de busca que a aplicação externaliza e o identificador da cidade que foi mapeado no passo anterior, conforme a Figura 11.

  O método será passado como *GET*, o identificador da cidade será informado no *Id*, a unidade será informada como métrica para que os dados sejam retornados em graus Celsius e é necessário a passagem de uma *appId*, configurada no site do openweathermap.

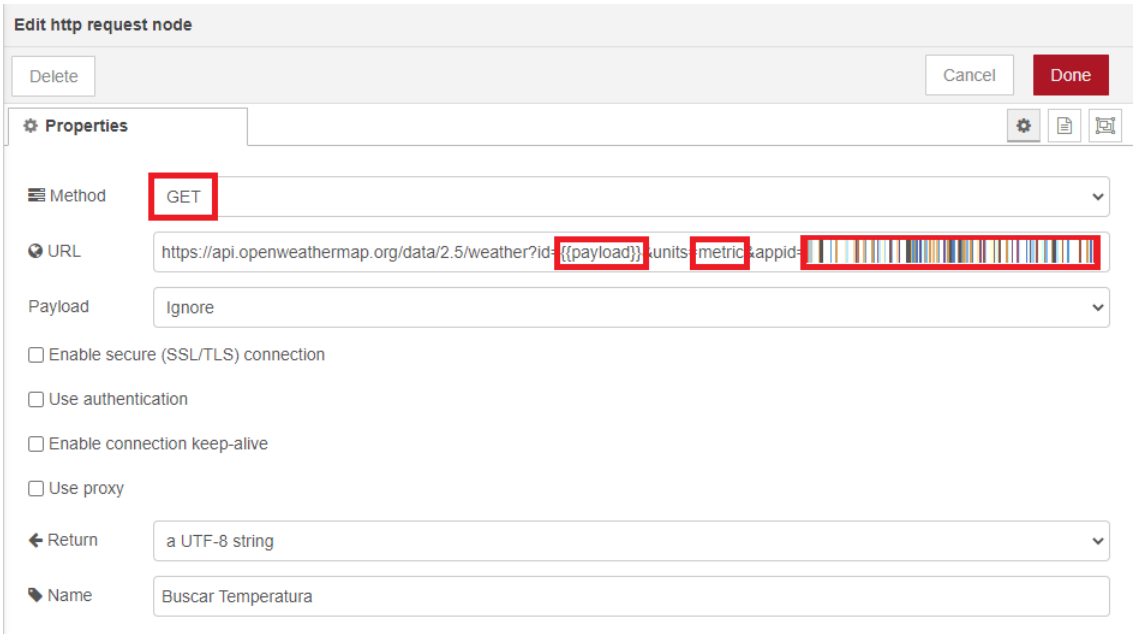

Figura 11: Requisição para API do openweathermap

Após essa chamada, é necessário utilizar o pacote básico para tratar o retorno da aplicação externa como um objeto JSON. Essa tratativa é toda conduzida via nó e não necessita de nenhuma implementação de código.

[2] `https://openweathermap.org`

Com o objeto retornado pela aplicação, deve ser utilizado a biblioteca *node-red-dashboard* para demonstrar a temperatura recebida em um painel de controle que é disponibilizado internamente no Node-RED. Para essa demonstração da temperatura será utilizado um cartão que demonstrará a temperatura em graus Celsius da cidade consultada na aplicação.

Por fim, será enviado um *e-mail* informando qual nome da cidade e sua temperatura, para isso será criado um modelo de texto a partir do objeto JSON recebido. Esse modelo é gerado a partir da biblioteca base do Node-RED, sendo necessário apenas a formatação do texto escritamente, sem a necessidade de criação de códigos de acordo com a Figura 12. Esse modelo também será enviado para o painel de controle para auxiliar na demonstração dos dados.

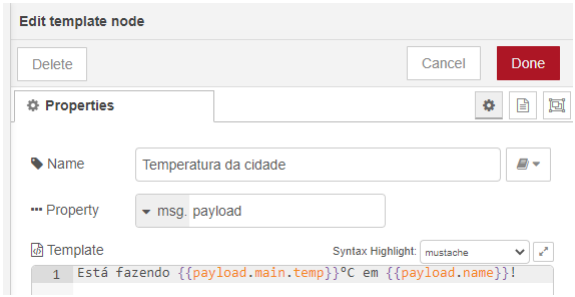

Figura 12: Template com as informações de temperatura

Com esse modelo gerado, será utilizado a biblioteca *node-red-node-email* para o envio de *e-mail* de forma simples, sem a necessidade de nenhuma configuração de SMTP ou algo do tipo, tudo é feito via *interface* do nó e não é necessária nenhuma implementação de código conforme a Figura 13. É necessário informar o *e-mail* de destino, qual o servidor e o usuário e senha da conta que será responsável pelo envio.

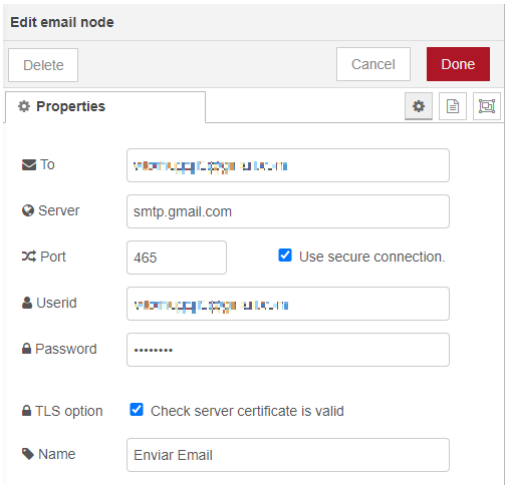

Figura 13: Envio de email através do Node-RED

O Resultado final é um painel de controle totalmente customizável que contém as informações finais. Esse painel é demonstrado na Figura 14.

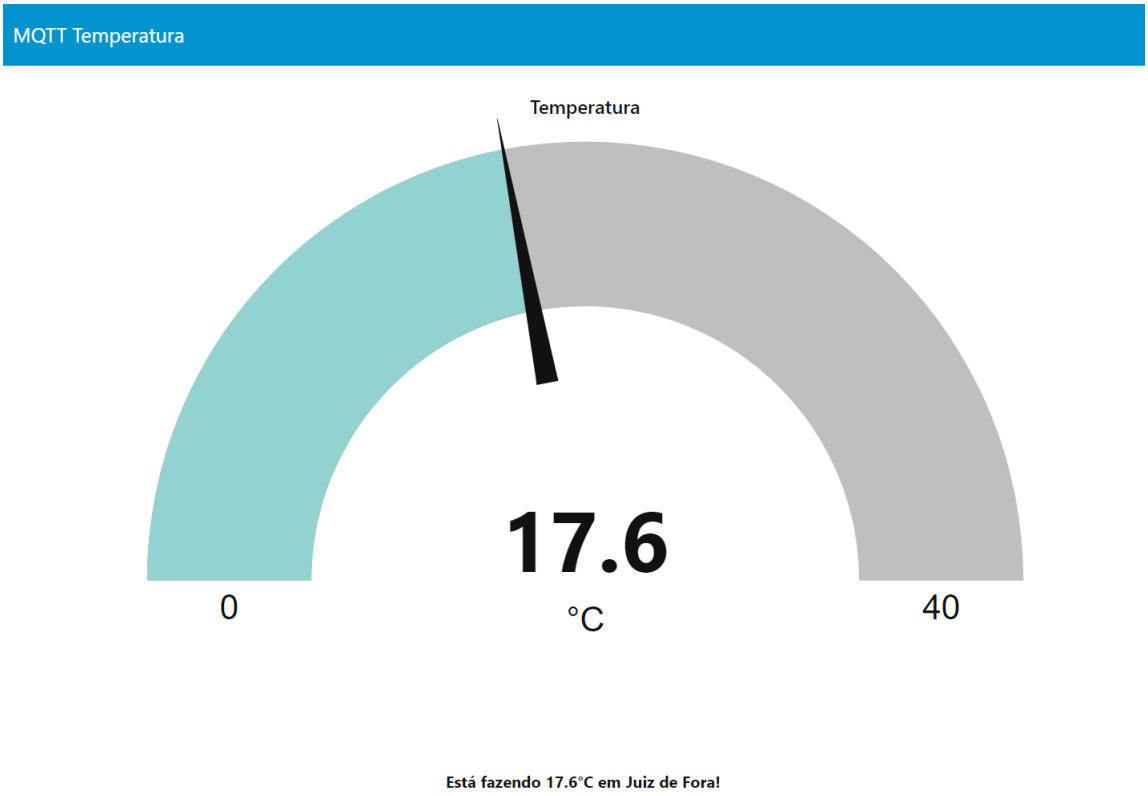

Figura 14: Painel de Controle Node-RED

- Fluxo de Falha

  Com o erro ao identificar a cidade, será formatado um texto escrito informando que a cidade não pode ser mapeada e novamente será utilizada a biblioteca *node-red-node-email* para envio de *e-mail* informando este caso.

3.1.3 IMPLEMENTAÇÃO VIA AWS IOT

Para realizar a implementação do modelo definido na AWS, é necessário inicialmente se cadastrar no ambiente da Amazon e ter acesso ao *console*. Após o cadastramento é preciso acessar o serviço IoT *Core*, que é responsável por efetuar o manejo do ambiente de desenvolvimento de IoT. Para continuar na implementação do modelo, é necessário configurar o MQTT. Essa configuração é efetuada através da *interface*, mas é composta por vários passos, diferentemente do que foi observado no Node-RED, onde essa implementação era conduzida de forma visual e simples.

Inicialmente deve ser configurado toda a parte do MQTT e para isso deve ser criado uma política na AWS, essa criação é feita a partir da *interface web* do AWS IoT. Nessa criação são necessários alguns passos:

- Definir um nome para a política.
- Definir uma ação, será utilizado o valor iot:*, este valor permite que todas as ações relativas a IoT sejam acessadas pelos objetos relacionados com a política.
- Definir nome de recurso *Amazon Resource Name* (ARN), será utilizado o valor *, este valor permite a todos os objetos terem permissão de acesso.

Esses passos são demonstrados na Figura 15.

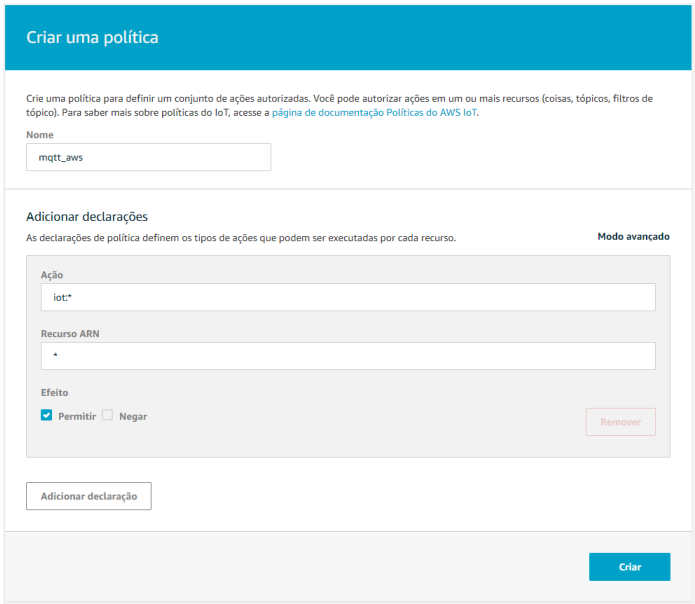

Figura 15: Criação da política de acesso na AWS

Após a criação da política é necessário baixar os certificados disponibilizados pela AWS, que são exibidos em tela. Posteriormente será necessário criar uma coisa e essa será correlacionada com a política definida anteriormente, após esse relacionamento será finalizado seu registro.

Tendo conduzido essa configuração, é necessário obter acesso ao *endpoint* do objeto que foi registrado acima para autorizar o recebimento de mensagens MQTT. Esse *endpoint* é encontrado nas configurações da coisa na parte de interação, vide Figura 16.

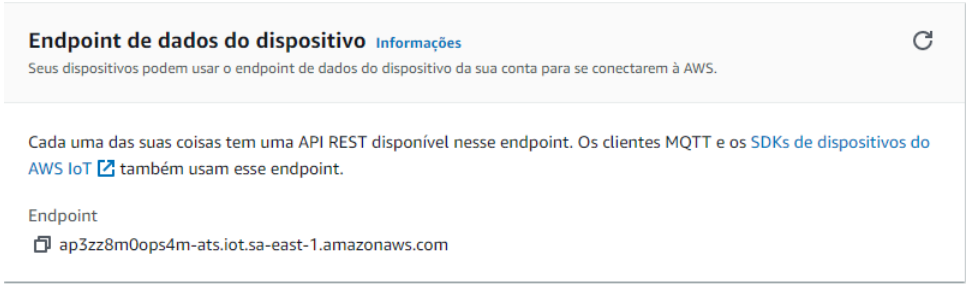

Figura 16: Endpoint do Topico MQTT na AWS

É possível perceber que comparado ao fluxo implementado no Node-RED, têm-se muitos passos e estes são complicados, algo que o Node-RED consegue proporcionar de forma simples e visual ao usuário.

Para que seja viável fazer requisições ao *endpoint* do objeto, é preciso configurar toda a parte de certificados e chaves de acesso. Após tudo isso feito, será possível realizar um envio de mensagem MQTT para a AWS e será usado o Node-RED para realizar esse envio, visto que ele já conta com uma biblioteca específica para esse momento, o que facilitará a implementação do *broker*.

De forma equivalente, como foi estruturado o recebimento de mensagem na implementação do Node-RED, é possível configurar o despacho de mensagens MQTT, permitindo o anexo dos certificados de forma visual e simples, auxiliando no procedimento de envio de mensagem para AWS. É necessário configurar um nó de envio de mensagem MQTT no modelo, conforme apresentado na Figura 17, onde é preciso definir o *server* e adicionar as políticas da AWS para que seja viável o recebimento das mensagens pelo *subscriber*.

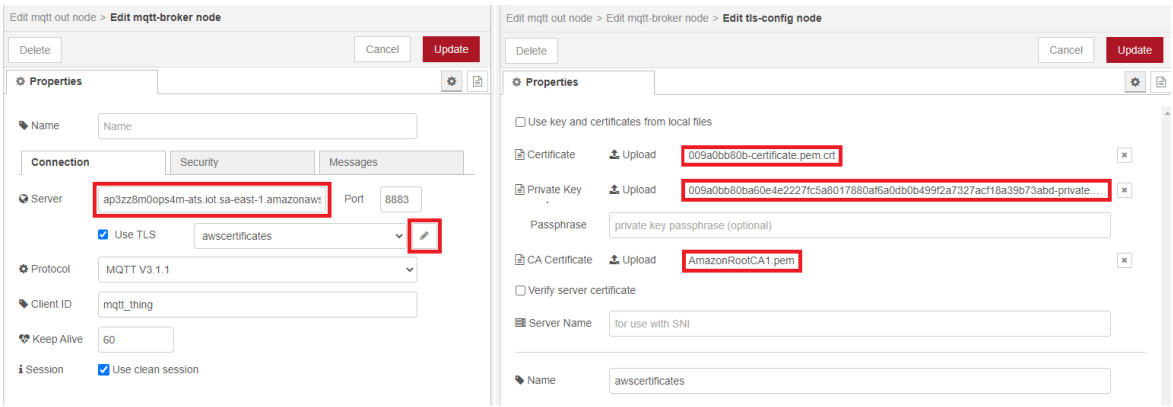

Figura 17: Configuração de Envio de mensagem MQTT a AWS através do Node-RED

Após toda a configuração, é obtido acesso ao envio de mensagens para o tópico na AWS. Esse acesso pode ser observado na parte de testes no menu do *Hub* IoT da AWS, da forma apresentada na Figura 18. Essa aba permite a adição das assinaturas nos tópicos e a possibilidade de seguir no fluxo de desenvolvimento.

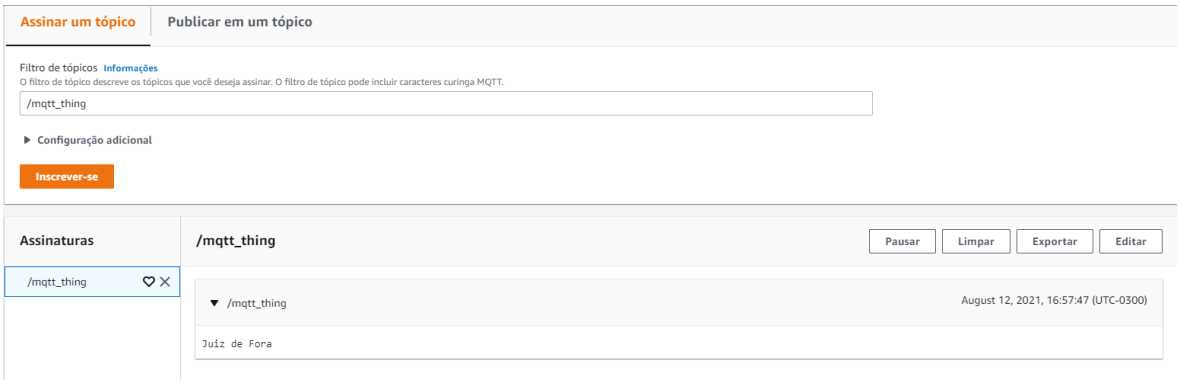

Figura 18: Recebimento da Mensagem no tópico MQTT na AWS

A *interface* da AWS permite que seja enviada uma requisição HTTPS a partir do recebimento de uma mensagem na fila, mas diferentemente do que foi implementado no Node-RED, essa *interface* não permite tratar o retorno dessa requisição.

É necessário para implementação do fluxo base que esse pedido seja feito a uma aplicação desenvolvida especificamente para o contexto do exemplo, onde esta será responsável por implementar as lógicas de envio para a aplicação do openweathermap[3]. Em seguida deverá enviar uma mensagem a um novo tópico, para que neste novo tópico seja possível adicionar a ação de enviar o *e-mail*, com as informações do clima para o usuário.

Essa implementação externa não será demonstrada aqui, considerando não ser o objetivo do trabalho demonstrar a fundo tal desenvolvimento. Apenas com a complexidade descrita é possível constatar o quão complicado pode ser uma simples implementação mostrada no Node-RED, quando executada por outras *interfaces* de controle.

Por fim, é necessário realizar o envio de *e-mail* com as informações para o usuário. A AWS conta com o serviço *Amazon Simple Notification Service* (SNS) que é responsável por realizar este processo nos desenvolvimentos pela *interface* da AWS.

Esse é um serviço que possibilita a efetuação de envios de *e-mail*, *sms* ou diversas outras funções. Para ser possível enviar um *e-mail*, é necessário configurar primeiramente um tópico no SNS, deve ser informado o tipo como padrão e dar um nome para criação do tópico conforme demonstrado na Figura 19.

[3] `https://openweathermap.org`

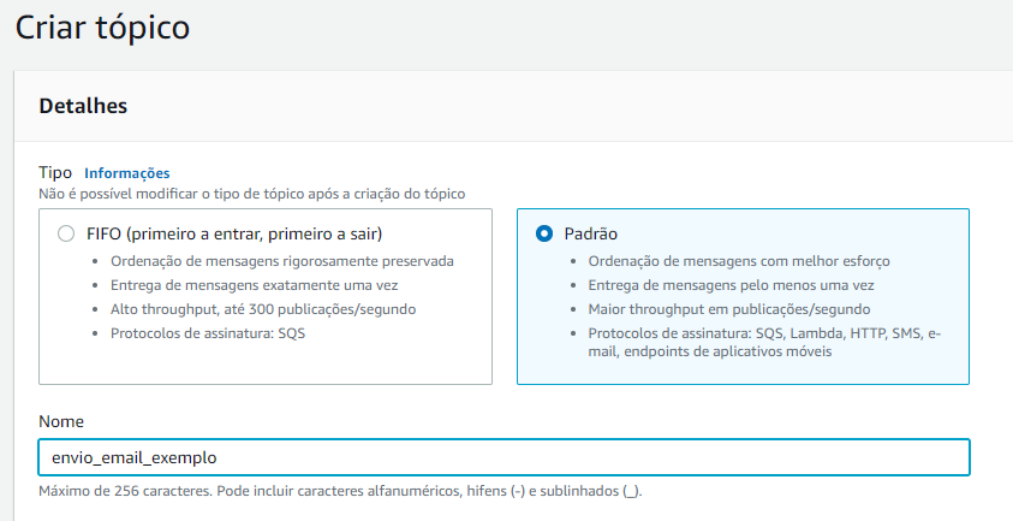

Figura 19: Criação do tópico SNS na AWS

Após a criação do tópico, é necessário criar uma assinatura. Neste momento deve ser informado o tipo do protocolo, que no modelo em questão será utilizado o protocolo de *e-mail* e após essa configuração, definir qual o endereço receberá a mensagem, de acordo com a Figura 20.

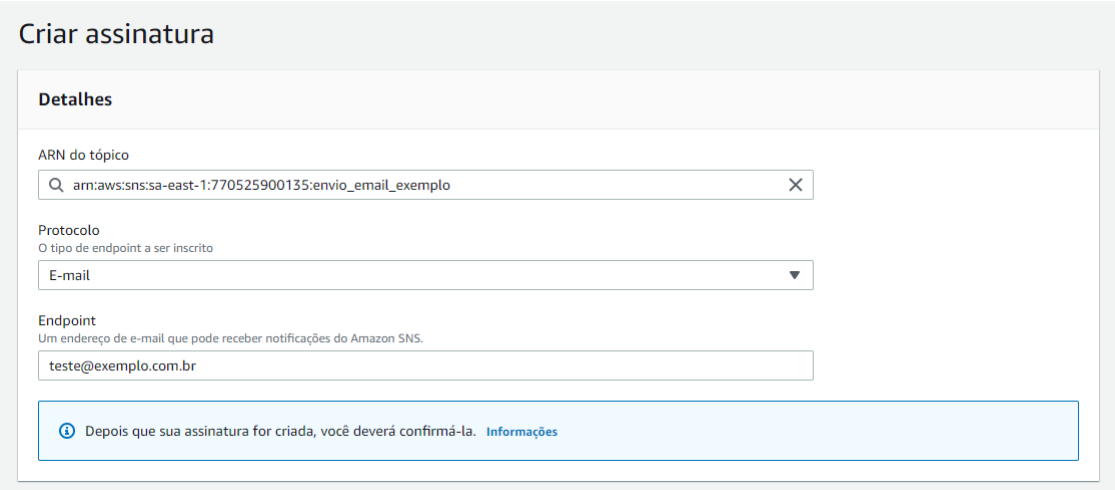

Figura 20: Criação da assinatura SNS na AWS

Feita essa configuração, é preciso estruturar uma regra no AWS IoT que será chamada no momento em que uma nova mensagem for recebida no tópico de envio de *e-mail* conforme informado na solução descrita acima. Para isso é necessário criar um *select* utilizando a linguagem SQL com o intuito de buscar tudo recebido no tópico conforme a Figura 21. Após essa definição é necessário criar uma ação para enviar uma mensagem como uma notificação por *push* SNS.

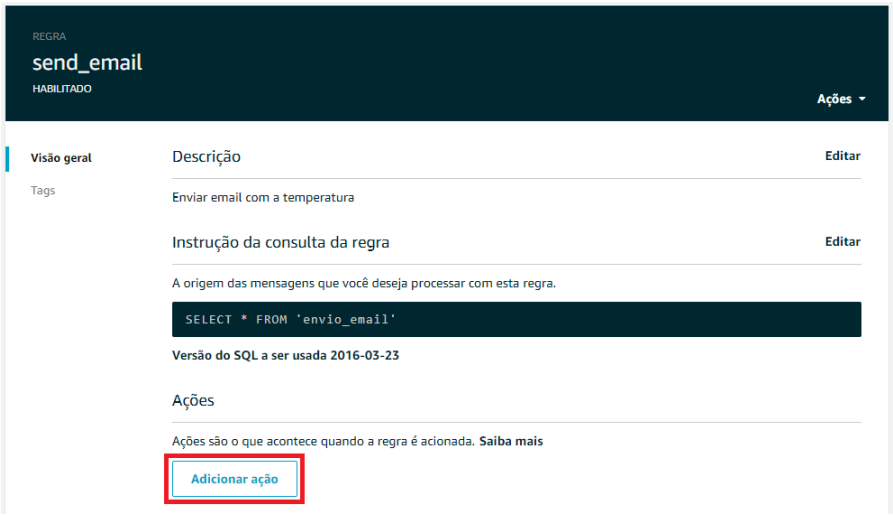

Figura 21: Criação da regra de envio de *e-mail* na AWS

Na criação da ação deve ser informado o destino de SNS que foi criado no passo anterior a criação da regra, definir o formato da mensagem como *RAW* e criar uma função para garantir ao AWS IoT invocar o serviço do SNS, o que pode ser visto na Figura 22.

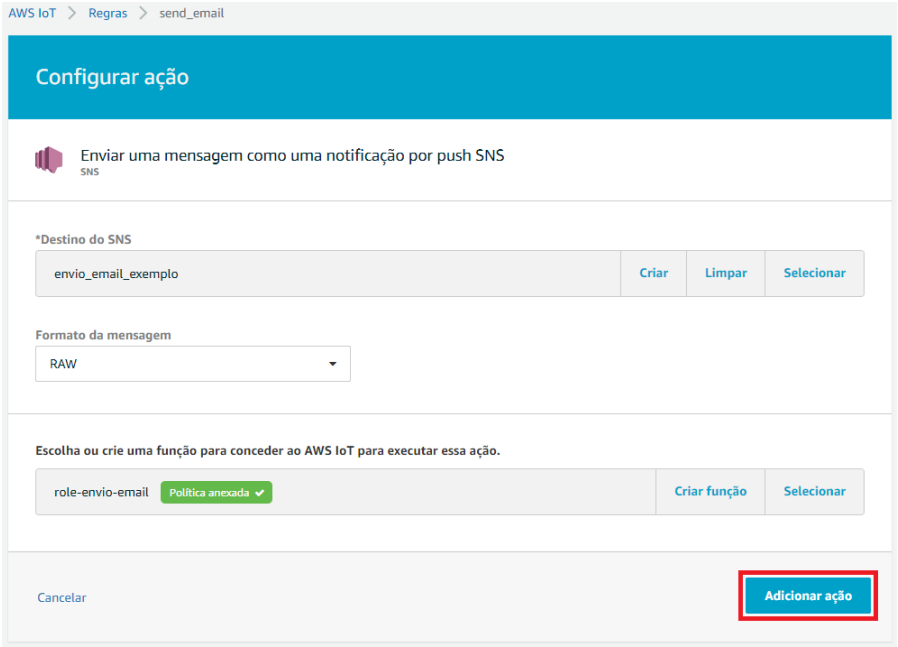

Figura 22: Criação da ação de envio de *e-mail* via SNS na AWS

Após toda configuração efetuada, todo o fluxo para envio do *e-mail* estará completo. É nítido que a implementação do mesmo fluxo feito pelo Node-RED é realizado de forma mais inteligível, podendo ser conduzida de forma mais acessível por usuários que já detenham conhecimento na área ou até mesmo por usuários leigos, visto que a configuração necessária para configurar o AWS *Core* é extremamente técnica e inviável para pessoas sem conhecimento em desenvolvimento.

3.1.4 IMPLEMENTAÇÃO NA AZURE IOT HUB

Para realizar a implementação do modelo definido na Azure IoT *Hub*, é necessário ter uma conta cadastrada para acesso a Azure. Diferentemente do Node-RED e da AWS, o Azure obriga ao usuário informar os dados de um cartão de crédito válido para realizar o cadastro e liberar o acesso ao sistema.

Após o registro é necessário procurar o recurso do *Hub* IoT para configurar uma nova solução em IoT, para que seja possível a configuração do MQTT. Essa criação é efetuada através da *interface* do Azure e necessita de um programa externo para possibilitar acesso às chaves de envio de mensagens via MQTT, o processo não é tão complexo mas necessita de ferramentas externas. Diferentemente do que foi observado na implementação pelo Node-RED, essa implementação é conduzida centralizadamente e unicamente pela *interface* gráfica.

Inicialmente deve ser criado um recurso do tipo IoT *Hub* para ser possível a configuração do MQTT. Essa criação é feita a partir da *interface web* do Azure conforme ilustrado na Figura 23, sendo necessário definir um nome e um novo grupo de recursos para a geração do *Hub*, porém é preciso atenção com a parte de gerenciamento, visto que por padrão o Azure seleciona um grupo de recursos pago no momento da criação, que deve ser substituído pelo gratuito, para que não haja cobranças monetárias na implementação.

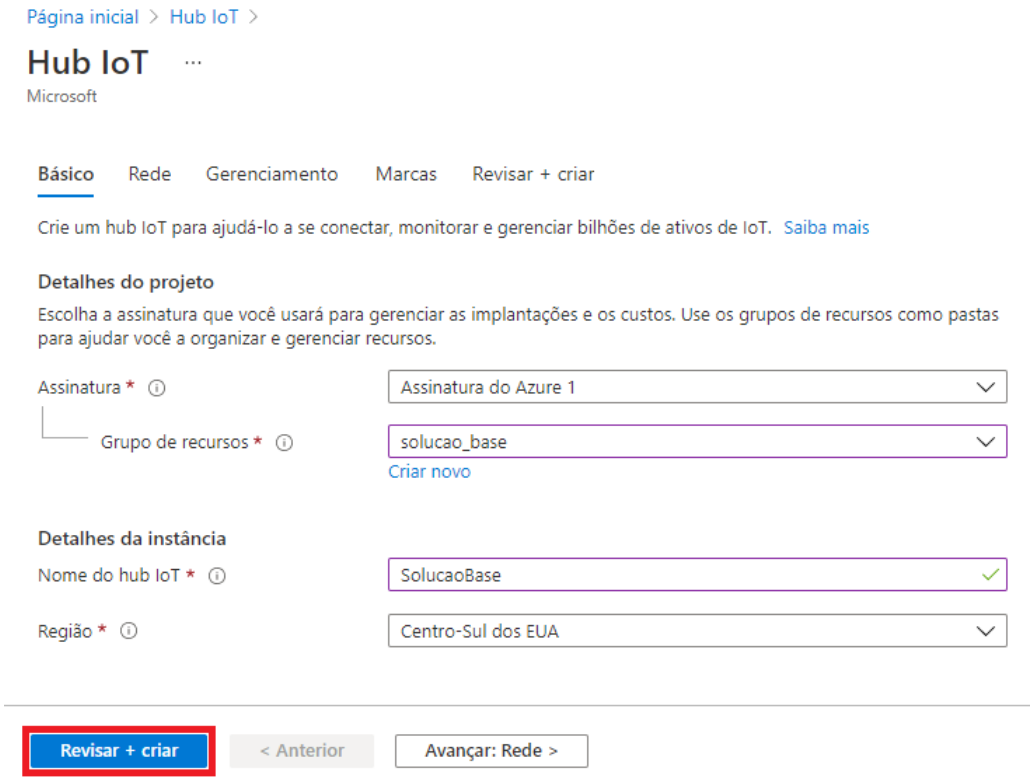

Figura 23: Criação do *Hub* IoT na Azure

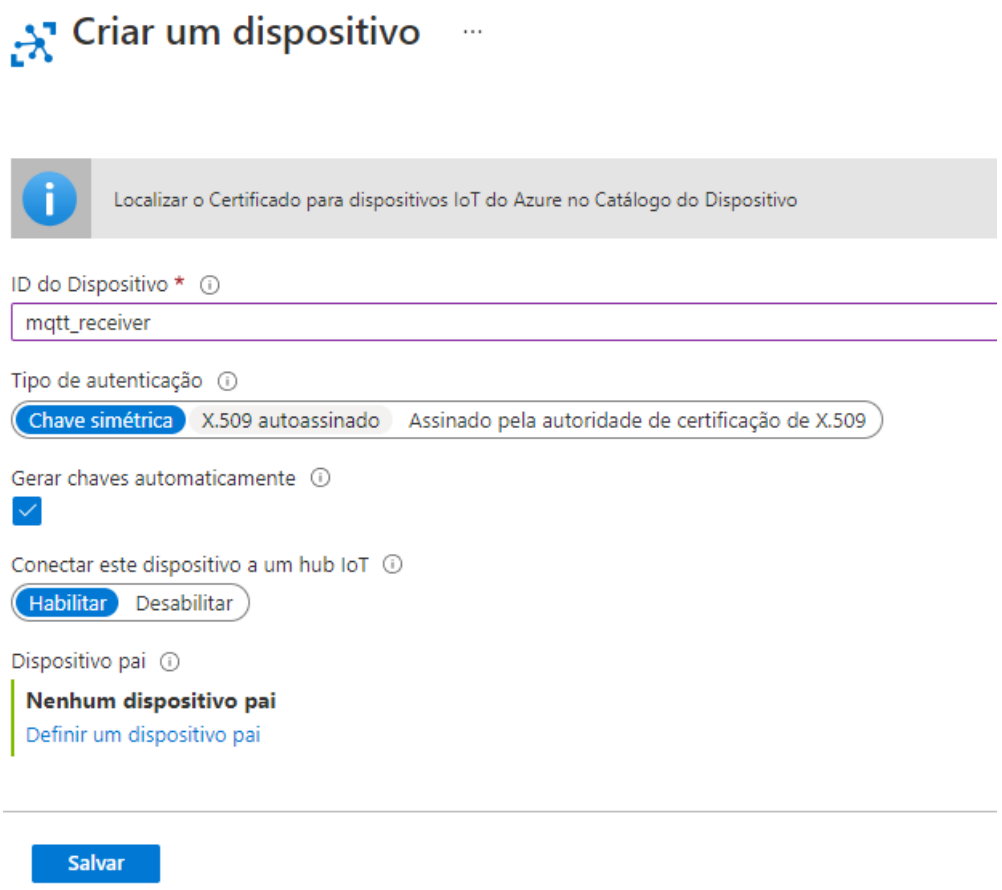

Figura 24: Criação do dispositivo de IoT na Azure

Com o dispositivo em mãos, pode se configurar o envio e recebimento das mensagens MQTT, correlato com a configuração da AWS, sendo necessário informar alguns valores na criação do *broker* para que o *subscriber* aceite receber as mensagens enviadas. Para isso serão necessários alguns dados:

- **Server do recurso**: é composto pelo seguinte modelo "ssl://[nome do recurso IoT *Hub*].azure-devices.net".
- **Identificador do cliente**: nome dado ao dispositivo IoT.
- **Usuário de segurança**: é composto pelo seguinte modelo "[nome do recurso IoT Hub].azure-devices.net/[nome do dispositivo IoT]".
- **Senha de segurança**: Para gerar uma senha de segurança é necessário na aba de configurações, acessar as políticas de acesso compartilhado em *iothubowner* e copiar o valor da cadeia de conexão primária. Em seguida é necessario abrir uma aplicação externa chamada *Device Explorer Twin*[4], que permite o manejo das configurações dos

[4] `https://github.com/Azure/azure-iot-sdks/releases/download/2016-11-17/SetupDeviceExplorer.msi`

dispositivos IoT e inserir o valor da cadeia de conexão no campo *IoT Hub Connection String* e clicar em *update*, conforme a Figura 25.

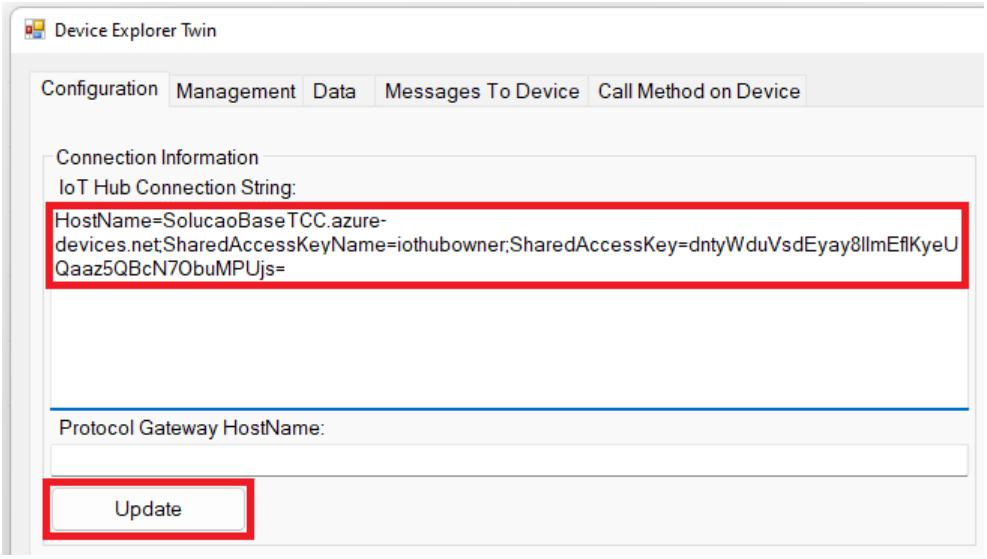

Figura 25: Inserindo dados no *Device Explorer Twin*

Após isso, em *management*, serão exibidos todos os dispositivos do recurso. Nesse momento é necessário clicar no dispositivo que será enviado a mensagem MQTT, em *SAS TOKEN* e finalmente em *generate*. Será gerado um valor de caracteres e a chave de segurança será o valor definido para o campo *SharedAccessSignature*, conforme ilustrado na Figura 26.

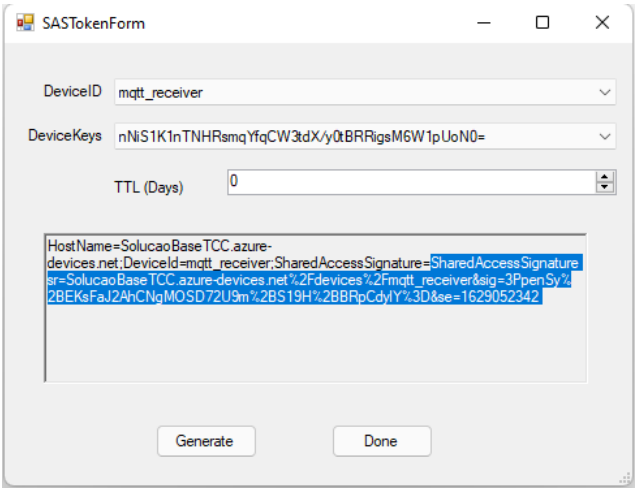

Figura 26: Gerando chave de acesso ao dispositivo no *Device Explorer Twin*

- **Tópico**: é composto pelo seguinte modelo “devices/[nome do dispositivo IoT]/ messages/events/tempData”.

Novamente é possível perceber que essa configuração é bem diferente do fluxo implementado no Node-RED, sendo mais complexa e custosa de configurar.

Com todas as informações disponíveis, é necessário configurar o envio da mensagem para o tópico da Azure para seguir no fluxo de implementação, portanto será utilizado a biblioteca de envio do Node-RED uma segunda vez para facilitar o envio da mensagem MQTT. De forma parecida com o utilizado para envio da AWS, deve ser configurado um nó de envio de mensagem MQTT onde é necessário definir o *server* e os dados de segurança como ilustrado na Figura 27.

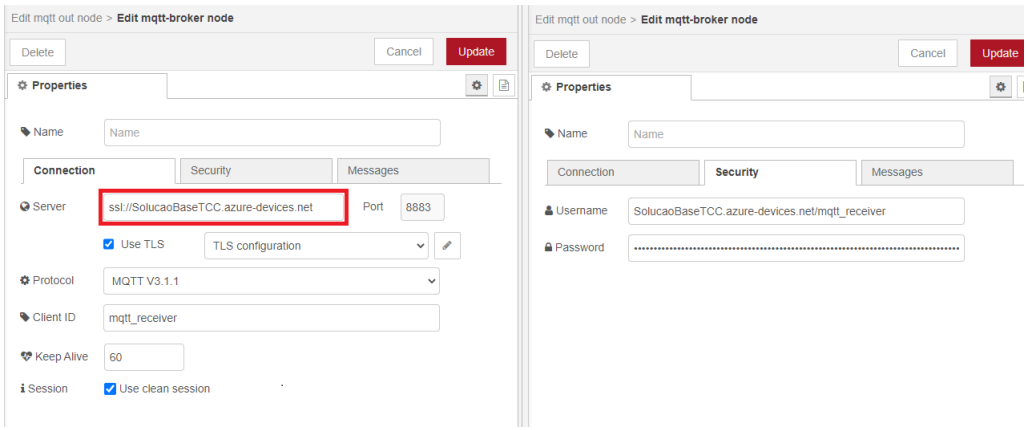

Figura 27: Configuração de Envio de mensagem MQTT à Azure através do Node-RED

Diferentemente do modelo da AWS, não são necessárias as importações de certificados e chaves, apenas o uso de usuário e senha para acesso ao tópico.

O Azure também não conta com a possibilidade do desenvolvimento por meio de *interface* visual para o envio da mensagem HTTPS a partir do recebimento da mensagem MQTT e também não é possível enviar o *e-mail* por meio da *interface web*.

É necessário para o desenvolvimento do restante do fluxo criar uma *Azure Function*, que será responsável por escutar o *subscriber* e a partir disso realizar a requisição HTTPS ao serviço de meteorologia e com o resultado dessa chamada, enviar o *e-mail* informativo ao usuário.

Esse desenvolvimento é mais fácil de ser elaborado via Visual Studio, que é um *Integrated Development Environment* (IDE) e permite um desenvolvimento de código de forma mais fácil que se feito via *console* da Azure.

De forma análoga ao ocorrido na implementação da solução pela AWS, esse desenvolvimento externo não será implementado neste trabalho, visto que apenas a complexidade descrita já demonstra a dificuldade global no escopo da implementação da Azure quando comparado ao demonstrado no Node-RED.

3.1.5 CONCLUSÃO

É possível perceber a diferença no esforço empregado na implementação do mesmo fluxo nas três *interfaces*, seja essa diferença observada na necessidade de configurações extras, na utilização de softwares externos à *interface* de criação ou mesmo na necessidade de implementações específicas para cobrir todas as necessidades básicas do desenvolvimento.

É viável também enumerar os pontos que cada uma das *interfaces* atendeu sobre os requisitos mínimos da implementação, a tabela 1 contempla essa análise.

	Leitura de Mensagens MQTT	Envio de Requisição HTTPS	Recebimento de resposta da requisição HTTPS	Envio de E-mail
Node-RED	✓	✓	✓	✓
AWS IoT Core	✓	✓	X	✓
Azure Hub IoT	✓	X	X	X

Tabela 1: Comparativo entre as implementações

Portanto, analisando as diferentes necessidades em cada implementação, é possível inferir que o Node-RED dispõe de uma maior facilidade em seu desenvolvimento, o que pode ser relacionado com:

- Sua *interface*, que permite um fluxo de criação onde agrega todas as necessidades em um mesmo lugar, deixando seu desenvolvimento mais intuitivo e simples.
- Sua configuração, onde não necessita de nenhum *software* externo para auxiliar na execução da implementação.
- Na utilização de suas bibliotecas, que permitem que todos os pontos necessários na implementação consigam ser executados e atendidos de forma simples.

Esses pontos corroboram para essa facilidade e permitem que o Node-RED seja a única das três *interfaces* que conseguiu cobrir todos os requisitos base do exemplo.

3.2 IMPLEMENTAÇÃO DE INTERFACE ALIADA COM MACHINE LEARNING

Será criada uma *interface* de IoT, aliada com a utilização de uma biblioteca de aprendizado de máquina, que implementa as chamadas ao TensorFlow. Isso evitará parte do esforço do desenvolvimento do modelo, em virtude dessa já implementar várias funções que serão necessárias para a criação de uma aplicação aliada com o aprendizado de máquina.

3.2.1 O QUE É TENSORFLOW.JS

Figura 28: TensorFlow.js

A Biblioteca que será responsável pela parte do Aprendizado de Máquina na aplicação é o TensorFlow.js, ela é gratuita e criada em JavaScript, permitindo que seja possível utilizar suas implementações dentro de um fluxo do Node-RED. Essa facilidade elimina a necessidade da implementação a partir do início, algo que é de extrema importância para o usuário, visto que permite que este não tenha a necessidade de ter um conhecimento aprofundado na área, tornando possível a usuários mais leigos a utilização de Aprendizado de Máquina em aplicações mais básicas.

3.2.2 IMPLEMENTAÇÃO

O Objetivo é utilizar aprendizado de máquina para descobrir, a partir de uma imagem recebida de um sensor, quais são os objetos que estão presentes naquela imagem. Será utilizado o TensorFlow para facilitar o uso de aprendizado de máquina. Utilizando o sistema de paletas, serão importados módulos relacionados ao TensorFlow.

Para a implementação do identificador de objetos, será necessário a importação de cinco bibliotecas:

- **node-red-contrib-browser-utils:** É uma biblioteca que permite a leitura de dados a partir de arquivos, câmeras e microfones.

- **node-red-contrib-tf-function:** É uma biblioteca que sobrescreve a função base do node-RED e a partir dela é possível acessar diretamente o TensorFlow e utilizar suas funções.
- **node-red-contrib-tf-model:** É uma biblioteca usada para carregar os modelos do TensorFlow e executar as inferências e gerar um modelo de detecção de objeto.
- **node-red-contrib-post-object-detection:** É uma biblioteca que lida com os resultados de predição a partir de um modelo de detecção de objeto.
- **node-red-contrib-image-tools:** É uma biblioteca utilizada para gerar imagens e código de barras diretamente no fluxo do Node-RED.

Primeiramente, é necessário capturar os dados do sensor, para isso será utilizada a biblioteca *node-red-contrib-browser-utils* para captar imagens de uma câmera conectada ao sistema. Essa forma de implementação foi escolhida para o estudo, mas existem inúmeras bibliotecas na paleta do Node-RED que possibilitam diversas formas de realizar a importação de uma imagem de um sensor para o fluxo, sem a necessidade de um conhecimento profundo da linguagem de desenvolvimento. Tudo pode ser realizado através da *interface drag-and-drop* disponível ao usuário.

A partir da captura da imagem, é preciso realizar um pré-processamento desse dado, será necessário utilizar o *node-red-contrib-tf-function* para que seja possível refinar as informações recebidas e deixá-las no formato que o modelo do TensorFlow espera, para que este possa atuar corretamente. Nesse momento é necessário a escrita de um código em *javascript* para realizar esse pré processamento da imagem conforme a Figura 29, porém as funções utilizadas neste código já estão disponíveis para utilização no objeto do TensorFlow, então não é necessária nenhuma implementação nova, apenas a utilização de funções já definidas na biblioteca.

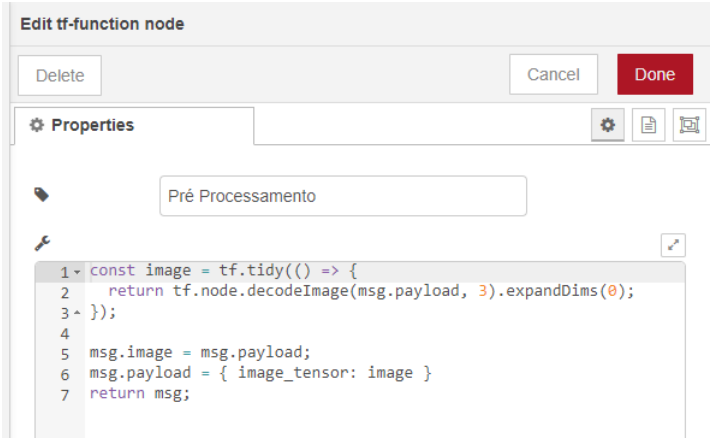

Figura 29: Função de pré processamento para o TensorFlow

Com a imagem modelada no formato correto, ela é passada ao *node-red-contrib-tf-model*, que identifica os objetos presentes na imagem a partir dos dados do COCO SSD, que é uma base de detecção de objetos utilizada no TensorFlow. Nesse momento, é necessário informar qual o endereço do modelo que pretende-se usar, novamente, tudo é configurável via *interface* do Node-RED, conforme visto na Figura 30, sem ser necessário a escrita de nenhum código adicional.

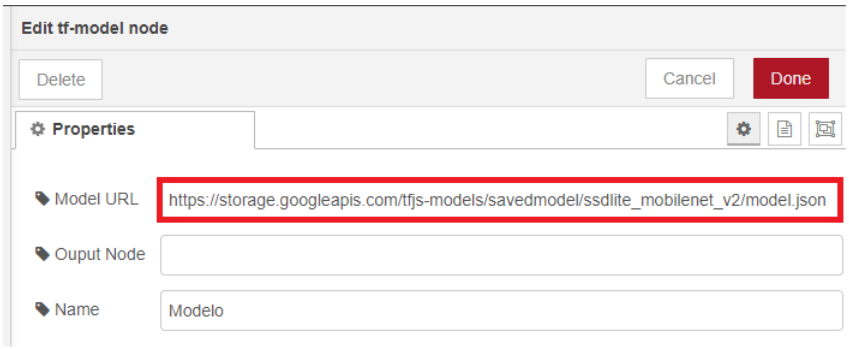

Figura 30: Configuração do Modelo para detecção de objetos

Esse mapeamento é passado para um pós-processamento que está presente na biblioteca *node-red-contrib-post-object-detection*, que a partir do modelo recebido, identifica as posições e nomes dos objetos presentes na imagem utilizando um *match* com um arquivo de classes relativo ao mapeamento do modelo, conforme a Figura 31.

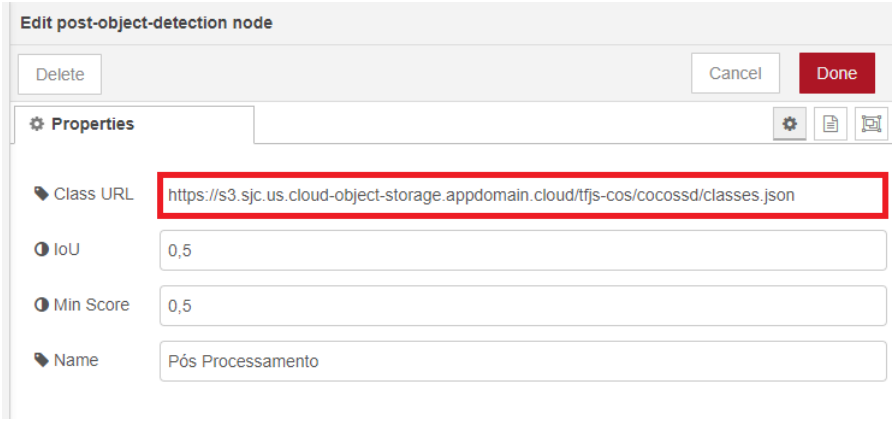

Figura 31: Configuração do pós-processamento para o TensorFlow

O resultado desse processamento é um objeto que contém as definições de cada objeto mapeado na imagem. Para demonstração do resultado, foi utilizado novamente o *node-red-contrib-post-object-detection* para gerar a imagem com o mapeamento dos objetos e o *node-red-contrib-image-tools* para exibir a imagem dentro do próprio fluxo de desenvolvimento. O fluxograma da Figura 32 demonstra o fluxo[5] completo implementado no Node-RED.

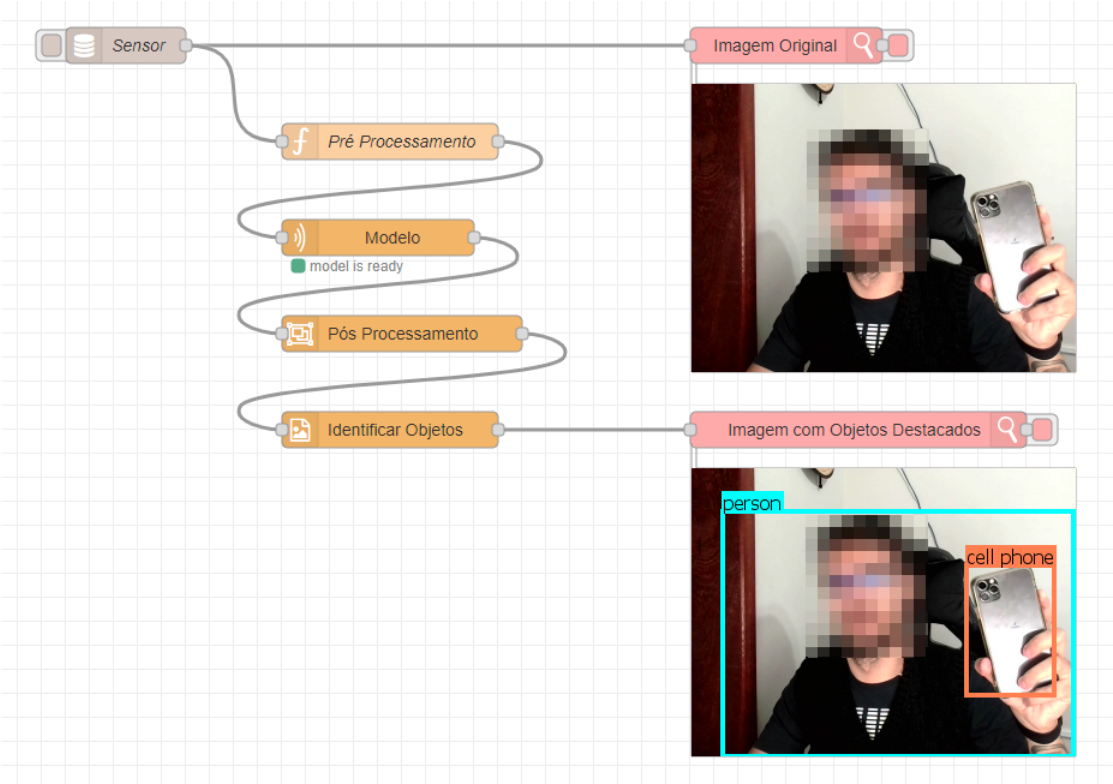

Figura 32: fluxo da implementação do detector de objetos através do Node-RED

[5] `https://vitorhugo-files-upload.s3.sa-east-1.amazonaws.com/flow.json`

4 CONCLUSÃO E TRABALHOS FUTUROS

A necessidade de um padronizador para implementação de *interfaces* IoT é de extrema importância devido ao crescimento da IoT e seu uso em diversas áreas de aplicação, tais como na agricultura, indústrias, ou até mesmo na vida em geral da população, áreas que são extremamente importantes para o crescimento global, onde a IoT consegue tornar a vida muito mais dinâmica e as interações muito mais simples. Essa expansão corrobora em um crescimento no número de dispositivos e de usuários que almejam ingressar nessa área, sejam esses usuários leigos ou com conhecimento prévio.

A pesquisa apresentada teve como propósito definir o Node-RED como *interface* IoT norteadora para as implementações de IoT. Estudar como ocorre a criação de uma *interface* IoT através do Node-RED, evidenciando em um primeiro momento quais são as facilidades que essa *interface* traz ao usuário quando comparada aos modelos *default* de *interfaces* existentes, seja pela sua facilidade de implementação, seu ambiente de desenvolvimento mais amigável ao usuário, sua linguagem base de desenvolvimento, seu sistema de bibliotecas externas ou outros.

Demonstrar essa diferença e facilidade ao comparar uma implementação de um fluxo básico em diferentes criadores de *interfaces* de controle, para que se possa entender na prática qual a real diferença em realizar uma implementação de uma *interface* através do Node-RED e de plataformas amplamente utilizadas no mercado, confirmando assim o motivo deste ter sido definido como um modelo a ser utilizado como padronizador.

Por fim, implementar uma solução mais complexa, utilizando uma integração com aprendizado de máquina de forma simples para qualquer usuário que busque implementar uma solução desse tipo, sem a necessidade de um conhecimento profundo sobre linguagens de programação ou mesmo sobre aprendizado de maquina, tudo isso sendo possível através do uso do Node-RED e seu sistema de bibliotecas externas, que permite a importação de implementações já prontas sobre fluxos de desenvolvimentos complexos, permitindo a qualquer tipo de usuário essa utilização de forma fácil.

Em trabalhos futuros, podem ser exploradas mais comparações com outras plataformas de implementação, levando mais a fundo a comparação caso a caso, trazendo assim uma maior fundamentação sobre as diferenças entre as plataformas padrões do mercado para o Node-RED. Ademais é possível explorar os aspectos mais técnicos do Node-RED, demonstrando todo seu potencial para aplicações mais complexas e como este reage em ambientes desse tipo, trazendo maior riqueza no detalhamento do comportamento dessa *interface* para todas as formas de implementação.

Por fim, podem ser construídas análises a partir das diferenças de custos sobre implementações em diferentes plataformas, demonstrando as diferenças em questões de valores pagos por serviço, capacitação de mão de obra necessária ou até mesmo do custo com dispositivos necessários na atuação das plataformas, trazendo uma maior completude a análise global, gerando assim uma maior certeza no motivo da utilização do Node-RED como padronizador.

REFERÊNCIAS

AVSYSTEM. **What is an internet of things platform?**, 2020. Disponível em https://www.avsystem.com/blog/what-is-internet-of-things-platform/.

AKER, A. P. Cloud computing, internet of things and the courts: Innovative tools or dangerous fad? **workshop for organised by the National Judicial Institute**, p. 28–36, 2019.

AMAZON. **How aws iot works**, 2021. Disponível em https://docs.aws.amazon.com/en_us/iot/latest/developerguide/aws-iot-how-it-works.html.

BANAFA, A. **Iot standardization and implementation challenges**, 2016. Disponível em https://iot.ieee.org/newsletter/july-2016/iot-standardization-and-implementation-challenges.html.

BERTOLETI, P. **O que são as plataformas iot e quais são as maiores do mercado (mic435)**, 2020. Disponível em https://www.newtoncbraga.com.br/index.php/iot/17606-o-que-sao-as-plataformas-iot-e-quais-sao-as-maiores-do-mercado-mic435.html.

AHADI, A. **Internet of things**, 2016. Disponível em http://comtech2.com/internet-of-things/.

CALABREZ, G. T. M. **Implementação de uma arquitetura iot com a ferramenta node-red**. In: Implementation of IOT architecture with Node-RED, Paraná, 2019. Disponível em http://repositorio.roca.utfpr.edu.br/jspui/handle/1/12033.

COMPTIA. **Internet of things insights and opportunities**, 2020. Disponível em https://connect.comptia.org/content/research/internet-of-things-insights-and-opportunities.

FERNANDES, N. **O que é o protocolo mqtt?**, 2021. Disponível em https://www.hitecnologia.com.br/blog/o-que-e-protocolo-mqtt/.

GROWTHENABLER. Discover key trends & insights on disruptive technologies & iot innovations. **GrowthEnabler**, p. 6–8, 2017.

GROOPMAN, J. **Design iot user interfaces beyond the screen**, 2020. Disponível em https://internetofthingsagenda.techtarget.com/feature/The-IoT-user-interface-designs-Thinking-beyond-the-screen.

GUTH, J.; BREITENBÜCHER, U.; FALKENTHAL, M.; FREMANTLE, P.; KOPP, O.; LEYMANN, F. ; REINFURT, L. **A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences.**, p. 81–101. Springer Singapore, Singapore, 2018.

HEUVELDOP, N. **Ericsson mobility report**. In: Ericsson Mobility Report, 2017.

IBGE. **Projeção da população do brasil e das unidades da federação**, 2021. Disponível em https://www.ibge.gov.br/apps/populacao/projecao/index.html.

JASPERNEITE, J.; SAUTER, T. ; WOLLSCHLAEGER, M. Why we need automation models: Handling complexity in industry 4.0 and the internet of things. **IEEE Industrial Electronics Magazine**, v.14, n.1, p. 29–40, 2020.

MQTT.ORG. **Mqtt: The standard for iot messaging**, 2020. Disponível em `https://mqtt.org/`.

MEIRELLES, F. S. 32ª pesquisa anual do uso de ti nas empresas, 2021. **FGV EAESP**, 2021.

MICROSOFT. **Hub iot do azure**, 2021. Disponível em `https://azure.microsoft.com/pt-br/services/iot-hub/#overview`.

OPENJS FOUNDATION. **Node-red, low-code programming for event-driven applications**, 2020. Disponível em `https://nodered.org/`.

PATEL, K. K.; PATEL, S. M.; Scholar, P. ; Salazar, C. Internet of things-iot: definition, characteristics, architecture, enabling technologies, application and future challenges. **International journal of engineering science and computing**, v.6, n.5, 2016.

PERRY, S. **Top 5 reasons to use node-red right now**, 2017. Disponível em `https://developer.ibm.com/blogs/top-5-reasons-to-use-node-red-right-now`.

RUPARELIYA, P. **Top iot development tools & platforms with comparison [2021]**, 2021. Disponível em `https://www.intuz.com/blog/top-iot-development-platforms-and-tools`.

SANTOS, D. d. Iot cloud framework. **Institute of Informatics - PPGC/UFGRS**, p. 1–4, 2019.

SCHULTZ, F. **O que é internet das coisas (iot) e como funciona?**, 2020. Disponível em `https://milvus.com.br/internet-das-coisas-iot/`.

SIMPKIN, C.; TAYLOR, I.; HARBORNE, D.; BENT, G.; PREECE, A. ; GANTI, R. K. Efficient orchestration of node-red iot workflows using a vector symbolic architecture. **Future Generation Computer Systems**, v.111, p. 117 – 131, 2020. Disponível em `http://www.sciencedirect.com/science/article/pii/S0167739X19317467`.

WELTER, C. **Model of things: Uma abordagem de desenvolvimento de software dirigida por modelos para aplicações cloud of things**. In: Model of Things: Uma Abordagem de Desenvolvimento de Software Dirigida por Modelos para Aplicações Cloud of Things, São Leopoldo, 2019.

YUAN, M. **Conhecendo o mqtt**, 2017. Disponível em `https://developer.ibm.com/br/articles/iot-mqtt-why-good-for-iot/`.