PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ARQUITETURA PARA MONITORAMENTO DE TEMPERATURA EM DATA CENTERS UTILIZANDO COMPUTAÇÃO EM NUVEM

RUAN FLESCH PEREIRA

Trabalho de Conclusão II apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Tiago Ferreto

Porto Alegre
2023

ARQUITETURA PARA MONITORAMENTO DE TEMPERATURA EM DATA CENTERS UTILIZANDO COMPUTAÇÃO EM NUVEM

RESUMO

O monitoramento de temperatura é uma atividade fundamental em Data Centers para garantir o funcionamento adequado da infraestrutura e assegurar a alta disponibilidade dos serviços hospedados. Este trabalho tem como objetivo analisar teoricamente e na prática diferentes modelos de sistemas de monitoramento de temperatura em Data Centers, utilizando uma das técnicas mais avançadas do mercado: a integração da Computação em Nuvem com sensores de temperatura inteligentes, monitorando diferentes pontos do Data Center. A adoção dessas duas tecnologias forma uma poderosa ferramenta para o controle de ambientes que exigem alta disponibilidade.

Inicialmente, serão apresentados conceitos teóricos sobre Data Centers e a importância de seus componentes, destacando os motivos pelos quais é essencial monitorá-los continuamente. Além disso, serão abordados temas como Internet das Coisas, Computação em Nuvem e TI Verde. Com base em trabalhos relacionados e estudos de ferramentas de mercado, optamos pela utilização dos serviços da *Amazon Web Services* para a aplicação prática deste projeto. Realizamos diversos experimentos para analisar e validar o projeto proposto, utilizando diferentes cargas de trabalho e gerando uma estimativa de custo de infraestrutura na nuvem. Essa análise permitiu obter dados valiosos sobre o desempenho do sistema proposto, bem como identificar os custos associados à sua implementação e operação na nuvem.

Palavras-Chave: Data Center, Monitoramento de Temperatura, Computação em Nuvem, Internet das Coisas.

ARCHITECTURE FOR TEMPERATURE MONITORING IN DATA CENTERS USING CLOUD COMPUTING

ABSTRACT

Temperature monitoring is a fundamental activity in Data Centers to ensure proper functioning of the infrastructure and ensure high availability of hosted services. This work aims to theoretically and practically analyze different models of temperature monitoring systems in Data Centers, using one of the most advanced techniques in the market: the integration of Cloud Computing with intelligent temperature sensors, monitoring different points of the Data Center. The adoption of these two technologies forms a powerful tool for controlling environments that require high availability.

Initially, theoretical concepts about Data Centers and the importance of their components will be presented, highlighting the reasons why continuous monitoring is essential. In addition, topics such as Internet of Things, Cloud Computing, and Green IT will be addressed. Based on related work and market tool studies, we have chosen to use the services of Amazon Web Services for the practical application of this project. We conducted several experiments to analyze and validate the proposed project, using different workloads and generating an estimate of the cloud infrastructure cost. This analysis allowed us to obtain valuable data on the performance of the proposed system, as well as identify the costs associated with its implementation and operation in the cloud.

Keywords: Data Center, Temperature Monitoring, Cloud Computing, Internet of Things.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

SUMÁRIO

1. INTRODUÇÃO

1.1 Motivação

O Data Center é considerado o elemento central da infraestrutura de TI. Neste local, ocorre o processamento de dados de uma ou diversas organizações. Com o avanço da tecnologia, juntamente das facilidades que ela proporciona no nosso cotidiano, grande parte das empresas possuem um Data Center próprio ou utilizam um de forma terceirizada. Um Data Center pode ser utilizado para os mais diversos fins, desde o armazenamento de dados, até a hospedagem de aplicativos e serviços que podem ser disponibilizados para uso interno ou fornecidos para terceiros.

Por conta da criticidade do ambiente, é de suma importância manter os níveis de temperatura em padrões aceitáveis durante sua operação. Encontrar maneiras de monitorar e armazenar estes dados, ajuda o time de gerência de infraestrutura a realizar estudos referentes aos horários de maior aquecimento de seus equipamentos, podendo realizar intervenções físicas e/ou virtuais a fim de melhorar o desempenho e a durabilidade dos ativos de TI.

1.2 Descrição do problema

Os equipamentos que compõem um Data Center fazem a conexão física e lógica dos mais diversos serviços. Estas aplicações atendem todos os tipos de negócios e incumbências que uma empresa pode possuir. Podemos dizer que os dispositivos encontrados nestes ambientes são os responsáveis por manter a integridade e disponibilidade de uma organização. Os principais componentes encontrados nesta infraestrutura física incluem: roteadores, switches, firewalls, sistemas de armazenamento, refrigeração, servidores e controladores de disponibilização de aplicativos[21].

Esta infraestrutura pode ser encontrada em diversos tamanhos. Podemos dizer que quanto maior a empresa, maior será o espaço necessário para processar e manter seus dados de maneira íntegra, pois está constantemente recebendo um grande volume de informações e milhares de acessos simultâneos de seus clientes. Portanto, o ambiente físico pode ser do tamanho de um armário, uma sala dedicada ou um armazém. Algumas empresas podem precisar de mais de uma instalação física, o que pode se tornar muito custoso. Para diminuir estes gastos, também é possível alugar espaço num determinado servidor e fazer com que um terceiro mantenha a operação e gerenciamento dos ativos de

TI, dispensando os custos com *hardware* e mão de obra qualificada para gerir uma central própria.

Os Data Centers que processam grandes aplicações ou trabalham com um volume elevado de dados estão propensos a operar em altas temperaturas no seu ambiente físico, tornando-os dependentes de sistemas de resfriamento e controle de umidade. O aquecimento dos equipamentos hospedados nestes locais, além de comprometer a sua efetividade, pode dificultar o funcionamento de seus componentes, levando a diminuição de vida útil. Para certificar-se que os centros de processamento de dados estejam operando conforme as normas e exigências, surgiu a ASHRAE *(American Society of Heating, Refrigerating and Air Conditioning Engineers)*, associação responsável pela elaboração das normas de controle para climatização em Data Centers.

Segundo a ASHRAE, a temperatura que devemos encontrar na entrada de ar dos dispositivos que trabalham em temperaturas elevadas deve encontrar-se entre 18ºC e 27ºC, com a umidade do ar entre 40% e 55%. Preservar estes valores é um desafio para os responsáveis pelo gerenciamento destas instalações, já que o sistema deve funcionar de forma ininterrupta. Por conta da necessidade de manter os níveis num grau aceitável, em torno de 40% a 45% do custo de energia na operação destes ambientes concentra-se em climatização [4].

1.3 Descrição do trabalho

Este trabalho apresenta uma arquitetura de monitoramento de temperatura em uma infraestrutura de Data Center hospedada na nuvem. Um estudo teórico e prático é realizado, focando na implementação de um conjunto de dispositivos e ferramentas que, juntos, criam um sistema capaz de monitorar independentemente as condições do ambiente físico, com possíveis ajustes e configurações realizados remotamente pelo administrador do local. Serão apresentados testes e resultados que validam a utilização da tecnologia abordada.

A parte prática desse estudo foi desenvolvida na plataforma Amazon Web Services (AWS), utilizando técnicas de ponta como o Terraform e o Ansible. Além disso, foi desenvolvido um simulador de sensor de temperatura em Python para simular leituras de temperatura em tempo real. Essas técnicas e ferramentas avançadas permitiram uma avaliação abrangente da funcionalidade e desempenho do sistema proposto.

Através da infraestrutura de nuvem da AWS, o sistema demonstrou escalabilidade, confiabilidade e flexibilidade no gerenciamento das tarefas de monitoramento de temperatura. O uso do Terraform facilitou o provisionamento dos recursos necessários, garantindo a implantação eficiente do sistema de monitoramento. O Ansible desempenhou um pa-

pel crucial na automação da configuração e gerenciamento dos componentes do sistema, simplificando o processo de administração.

O simulador de sensor de temperatura em Python permitiu a criação de cenários de teste realistas. Ao integrar esse simulador ao sistema, foi possível simular e analisar diversas condições de temperatura e suas respostas correspondentes, garantindo estabilidade do sistema em diferentes situações durante os testes de carga.

1.4 Organização do documento

O trabalho está estruturado da seguinte forma:

No Capítulo 2, é apresentada a fundamentação teórica do tema, abordando o funcionamento de uma infraestrutura de Data Center, o conceito de Green IT, a Internet das Coisas e a Computação em Nuvem. São explorados os principais aspectos e características dessas áreas para embasar o desenvolvimento do projeto.

No Capítulo 3, são discutidos os trabalhos relacionados, enfatizando sua relevância, formas de implementação, comparações e adoções que podem ser integradas ao projeto em questão, permitindo contextualizar e identificar abordagens semelhantes, fornecendo uma base para o desenvolvimento da solução.

No Capítulo 4, são apresentados o projeto e o desenvolvimento do trabalho. São detalhados os objetivos, a metodologia adotada, as atividades realizadas e outros componentes necessários para a conclusão do projeto. Nesse capítulo, são definidos os passos e a estrutura do trabalho, fornecendo uma visão geral do processo de desenvolvimento.

No Capítulo 5, é descrito o desenvolvimento prático do Trabalho. São apresentados os detalhes da implementação, incluindo a utilização das técnicas mais recentes do mercado, como o Terraform, Ansible e um simulador de sensor de temperatura desenvolvido em Python. Esse capítulo demonstra como as teorias e conceitos discutidos nos capítulos anteriores foram aplicados na prática.

No Capítulo 6, é realizada a validação do projeto. São apresentados os testes e os resultados obtidos, demonstrando a eficácia e a confiabilidade da arquitetura de monitoramento de temperatura proposta. Essa validação permite verificar se o sistema atende aos requisitos e objetivos estabelecidos, garantindo a sua funcionalidade e desempenho.

No Capítulo 7, é apresentada a conclusão do estudo realizado. São destacados os principais resultados obtidos, as contribuições do projeto. Esse capítulo encerra o trabalho, resumindo as principais descobertas e conclusões derivadas do estudo.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta brevemente conceitos fundamentais relacionados ao desenvolvimento deste trabalho, abordaremos a importância e utilização da Infraestrutura de Data Centers, Green IT, Computação em Nuvem e Internet das coisas (IoT).

Como proposta, o trabalho apresentará uma arquitetura para o monitoramento da temperatura através da nuvem. Os Data Centers operam 24x7, portanto, acompanhar os níveis de temperatura e umidade são de suma importância para garantir a eficiência e durabilidade dos equipamentos existentes nestas instalações. Com a utilização dos sensores IoT, podemos receber em tempo real avisos de possíveis falhas em determinadas zonas. Através da Computação em Nuvem, é possível hospedar diversas funcionalidades de forma escalável e, ao mesmo tempo, proporcionar uma TI mais ecológica, dispensando a compra de novos recursos computacionais de maneira física.

2.1 Infraestrutura de Data Center

Os componentes existentes numa infraestrutura de Data Center incluem servidores, equipamentos de rede conhecidos como roteadores e switches, sistema de segurança, armazenamento, sistemas de gerenciamento para software e aplicativos, além de equipamentos de gerenciamento de energia e refrigeração, itens que necessitam operar de forma ininterrupta[21]. Conforme a Figura 2.1 a arquitetura de Data Center é composta por:

- *Entrance Room* (ER): é o espaço de interconexão do cabeamento estruturado do Data Center, além de ser a porta de entrada para o cabeamento vindo das operadoras de telecomunicações.
- *Main Distribution Area* (MDA): é a área onde se encontra a conexão central do Data Center, onde também ocorre a distribuição do cabeamento estruturado, local que conta com roteadores e o backbone da estrutura.
- *Horizontal Distribution Area* (HDA): é utilizado para realizar a conexão com a área de equipamentos, onde costumam ser encontrados switches LAN, SAN e KVM no local.
- *Equipment Distribution Area* (EDA): local onde se encontram equipamentos terminais, como servidores, storage e unidades de fita. Também possuem equipamentos de rede, como racks e gabinetes.
- *Air Conditioning* (AC): é onde se encontram os equipamentos responsáveis por manter o Data Center em temperaturas aceitáveis, evitando alterações de temperatura e umidade.

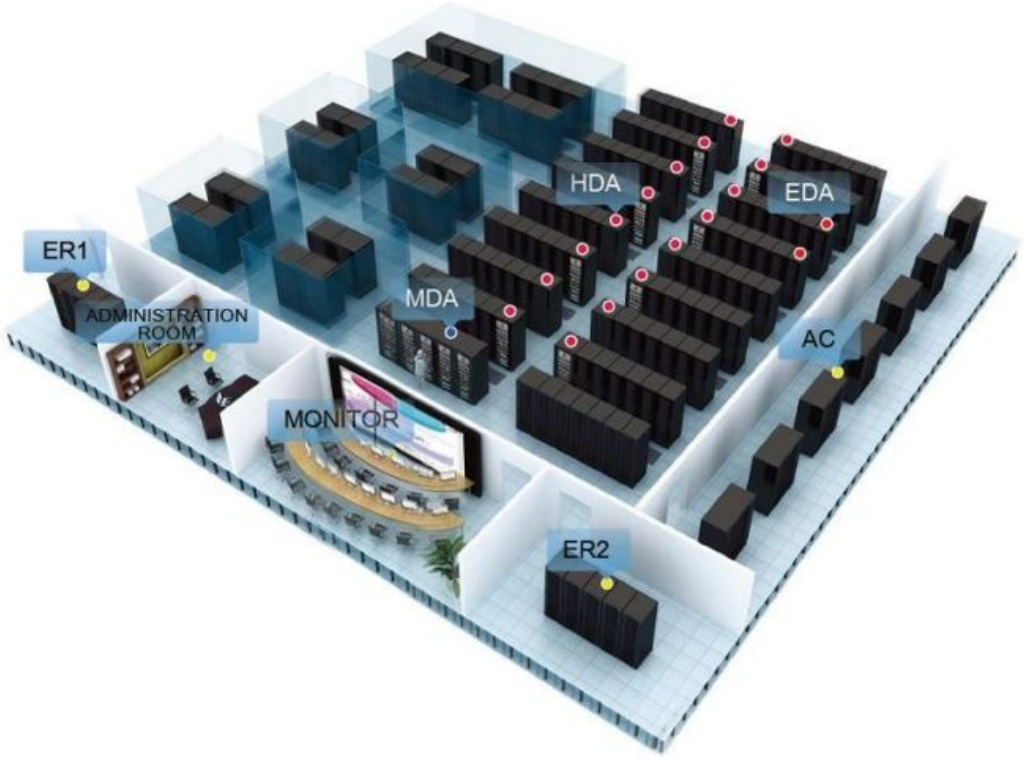

Figura 2.1 – Arquitetura de Data Center (Fonte: [18])

Desenvolver, gerenciar e operar a infraestrutura de um Data Center requer avaliações aprofundadas de desempenho e risco. A confiabilidade destas instalações pode ser entendida com a distribuição do tempo entre o ocorrer de uma falha, tanto em nível de sistema ou componente físico, até a devida correção e restabelecimento do serviço afetado. Garantir uma infraestrutura 100% imune a erros e desastres é uma tarefa impossível quando se fala em tecnologia, mas é possível preveni-las detectando as ameaças com antecedência, impedindo possíveis intermitências no sistema.

Para garantir a eficiência dos Data Centers, o *Uptime Institute* criou o Sistema de Classificação de Tiers para avaliar instalações de centros de dados em termos de desempenho da sua infraestrutura de sites e/ou tempo de atividade de seus ativos em nível de *hardware* e *software*. O Tier Standard é conhecido e reconhecido globalmente na comunidade de tecnologia, sua missão é instruir e ajudar Data Centers a atingirem suas metas de disponibilidade e, consequentemente, reduzir os riscos de suas operações. [20]

A Tier Certification é dada pelo *Uptime Institute* aos centros de processamento de dados que atendem as condições impostas pelo grupo, variando de modelo, modo de operação e criticalidade existente em determinado Data Center. Existem 4 certificações Tier [20]:

- Tier 1: Data Center básico
- Tier 2: Data Center redundante

- Tier 3: Data Center que possibilita a manutenção sem interrupções
- Tier 4: Data Center tolerante a falhas

2.2 Refrigeração em Data Center

Os ambientes de Data Center, em sua maioria, utilizam um modelo de climatização chamado de *Computer Room Air Conditioner* (CRAC). Esse sistema serve para monitorar e manter a temperatura da sala de equipamentos, distribuição de ar e umidade da sala de rede em níveis adequados. Através do piso elevado perfurado, o ar frio é injetado e conduzido para dentro dos racks de servidores, local onde encontramos a maior fonte geradora de calor dentro do ambiente. Este ar quente é expelido pela parte superior do rack. Apesar deste modelo ser uma fonte eficiente para controlar a energia e temperatura, o seu custo elevado impede a utilização em todos os Data Centers, sendo mais adotado em estruturas de grande porte.

Para manter a temperatura e umidade em níveis aceitáveis e garantir a eficiência dos dispositivos, existem diversos investimentos na área de refrigeração e atualização dos equipamentos de ar-condicionado. Durante a construção de um Data Center são levados em conta os tipos de condicionadores de ar e ainda a disposição do sistema de refrigeração no ambiente.

Visando o correto funcionamento de seus climatizadores, é indispensável a utilização de sistemas de redundância destes equipamentos, para que em caso de falha no sistema principal, um secundário assuma e garanta a temperatura dos ambientes em níveis aceitáveis para assegurar a continuidade do serviço. Existem diferentes níveis de redundância, cabe ao responsável entender qual o mais adequado para a sua infraestrutura[8]:

- O nível de redundância mais básico é o N, onde a redundância praticamente não existe, pois considera que a infraestrutura atua sempre em condições aceitáveis. Apesar de ser um cenário arriscado, é muito comum entre pequenas empresas.
- A redundância N+1 requer que exista ao menos um equipamento extra disponível para ser realocado em falhas, como exemplo, podemos considerar um servidor refrigerado por apenas um ar-condicionado, mas que possui um segundo para cobrir possíveis falhas.
- A redundância N+2 prevê dois equipamentos extras em caso de falhas, pode ser considerado como backup do backup, sendo característico nos Data Centers N+2.
- No modelo de redundância Nível 2N toda infraestrutura existente é duplicada, levando em conta hardware e software, fontes elétricas, segundo caminho para acesso, além do backup de dados.

- O nível de redundância mais alto, chamado de Nível 2(N+1), adota cuidados extras nos sistemas críticos, passando a ter o dobro da quantidade de equipamentos e um módulo extra para cada N.

2.3 Internet das Coisas (IoT)

O conceito de IoT é que praticamente todas as coisas físicas podem se tornar um computador conectado à Internet. Estes dispositivos não se tornam computadores na essência, mas conteriam pequenos computadores dentro de seus componentes. Quando isso ocorre, surge o nome de coisas inteligentes, pois os objetos começam agir de maneira independente, atendendo comandos e realizando simulações internas para verificar qual tarefa deve ser executada, diferente dos outros equipamentos que não possuem um microcontrolador que permita realizar tais atividades.[16].

Para o surgimento desta tecnologia, foi necessário o suporte e avanço de outras áreas de inovação. O desenvolvimento destas áreas correlacionadas, foi e ainda é de suma importância para a evolução dos dispositivos inteligentes, já que existe a dependência entre diversas áreas para o seu pleno funcionamento. A evolução de algumas destas áreas que incluem dispositivos IoT, já nos trazem diversos benefícios. Dentre estas inovações e benefícios, podemos citar avanços na domótica (tecnologia que realiza a gestão de todos os recursos habitacionais), automação industrial para construção de diversos tipos de equipamentos, além da logística e automação de residências [14].

Com a utilização de sensores e atuadores conectados às redes de computadores, é possível detectar o que está acontecendo em determinado ambiente. Para tratar esses dados, utiliza-se um algoritmo de software que permite realizar cálculos com base nos dados recebidos. Com estas leituras, o sistema busca métricas impostas e direciona os autuadores para modificar o local físico conforme as regras estabelecidas pelo usuário [16]. A maioria dos sensores e atuadores incluem modelos de transdutores. Estes dispositivos são responsáveis por converter uma forma de energia em outra. Os sensores existentes nos sistemas IoT utilizam alguns parâmetros físicos e os transformam em sinais elétricos. Da mesma forma, quase todos os atuadores nos sistemas de IoT captam sinais elétricos e os convertem em algum tipo de saída física.

Com o uso da IoT, é possível monitorar de forma ininterrupta a temperatura, umidade e os possíveis vazamentos de água em um Data Center através da adoção de sensores. Ao menor sinal de instabilidade ou mudança de estado, os sensores podem enviar alertas para uma determinada fonte de controle (aplicação), sendo possível transmitir estes dados para celular, e-mail ou painel de monitoramento da equipe responsável pelo controle do local, podendo visualizar qual a ocorrência está em andamento e seu status atual. Atra-

vés destas informações, o administrador pode disparar comandos de maneira remota para solucionar ou mitigar alguma intermitência[19].

Para padronizar a comunicação destes dispositivos é normalmente adotado o protocolo MQTT *(Message Queuing Telemetry Transport)*. A utilização deste protocolo de comunicação é mais rápida e segura, pois os protocolos que também poderiam ser adotados (HTTP e HTTPS) costumam ser pesados e utilizam uma grande quantidade de memória no envio de mensagens, o que torna a comunicação com os demais serviços mais lenta. Além disso, o MQTT impõe uma criptografia no disparo de suas mensagens, acrescentando uma camada de segurança durante a transferência das informações.

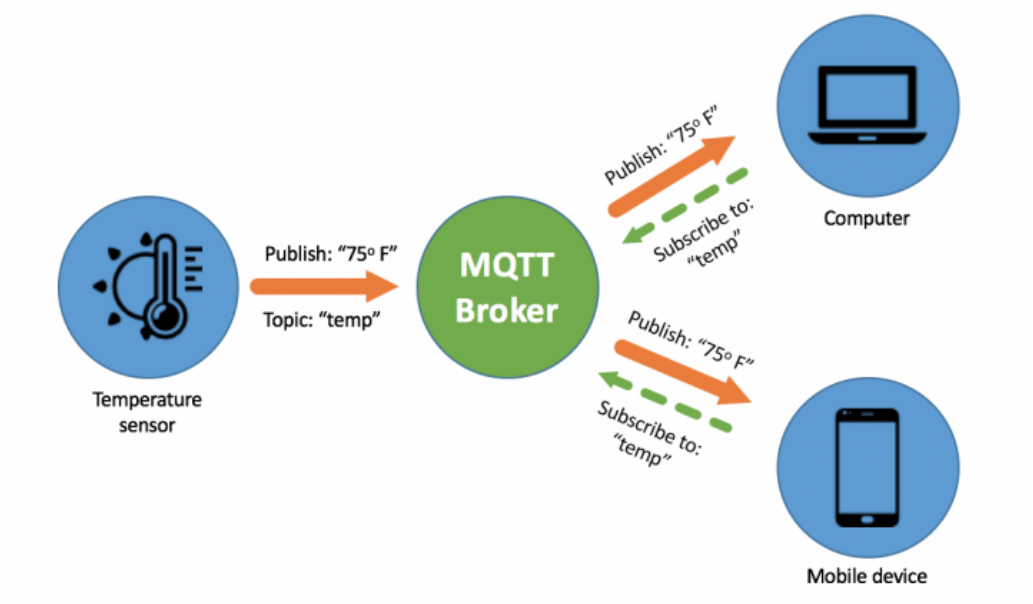

Figura 2.2 – Monitoramento e Controle por Aplicativo - MQTT (Fonte: [13])

Em cenários de Internet das Coisas, o funcionamento do protocolo acontece através da captação de uma variável através do sensor e o envio deste dado para um microcontrolador *(publisher)*. Esta informação é repassada para um *broker* (servidor) que ao recebê-la, compara os dados recebidos com métricas estabelecidas pelo usuário. Caso o dado esteja diferente ou igual ao da regra estabelecida, será realiza a transferência destas informações para os chamados *subscribers* que podem ser desde outros serviços até dispositivos que trabalham em cima do recebimento destes elementos. Seu funcionamento está ilustrado na Figura 2.2.

2.4 Computação em Nuvem

Segundo o NIST *(National Institute of Standards and Technology)*, "Computação em Nuvem é um modelo pay-per-use para permitir o acesso pela rede sob demanda, com disponibilidade e conveniência a um conjunto compartilhado de recursos computacionais

configuráveis (e.g., redes, servidores, armazenamento, aplicações, serviços) que podem ser rapidamente provisionados e liberados através de um custo de gerência, mas com miníma interação com o provedor de serviços.[9]"

A arquitetura através da Computação em Nuvem muda o modelo de operação da TI, deixando de operar através da compra de ativos, para operar através da aquisição de serviços. Com a utilização dos serviços oferecidos pela Computação em Nuvem, a TI obtém a otimização dos recursos e oferece a flexibilidade para seus usuários. Atualmente, os serviços em nuvem são na sua grande maioria oferecidos por grandes organizações como Google, Microsoft e Amazon.

A centralização e também consolidação de Data Centers em nuvem só foi capaz pela grande velocidade e estabilidade da Internet mundial, junto de tecnologias que permitem um alto poder de processamento e armazenamento em estruturas que reduzem o custo da infraestrutura de TI. Por outro lado, estes grandes Data Centers demandam uma grande quantidade de energia e consequentemente refrigeração, tornando grande parte dos processos mais complexos.

Dentro da proposta da terceirização da infraestrutura de TI, existem três modelos de serviços (Figura 2.3): Infraestrutura como Serviço (IaaS), Software como Serviço (Saas) e Plataforma como Serviço (PaaS) . A ideia destes serviços é oferecer um modelo "pago por tempo de uso e/ou execução"*(Pay-Per-Use)*, dispondo de escalabilidade e redundância, ou seja, aumentar ou diminuir os recursos conforme a necessidade e mitigar a interrupção do serviço por falhas de seus componentes[22].

A infraestrutura como serviço funciona através das grandes empresas de Computação em Nuvem que fornecem a sua infraestrutura para a criação de soluções. O cliente aluga o hardware, sejam eles, servidores, balanceadores de carga, firewall e cabos, porém, dispensa a necessidade de provisionar estes equipamentos. A configuração é feita remotamente pelo próprio usuário que ajusta e utiliza os itens conforme sua necessidade. Os itens criados podem ser reconfigurados ou escalados a qualquer momento. Este modelo é conhecido como IaaS[3].

Na utilização de Plataforma como Serviço (PaaS), o cliente aluga uma plataforma onde ele próprio implementa seus aplicativos sem a necessidade de configurar a infraestrutura. Alguns provedores do serviço também disponibilizam ambientes de desenvolvimento, homologação e testes, além de ferramentas de automação e de entrega de versões[3].

No Software como Serviço (SaaS), o usuário pode alugar determinado serviço oferecido pelo fornecedor e apenas realizar a sua configuração, sendo que neste modelo o pagamento é realizado conforme a utilização dos serviços que o cliente escolher, dispensando a necessidade de controle sobre a infraestrutura ou atualização do produto oferecido[3].

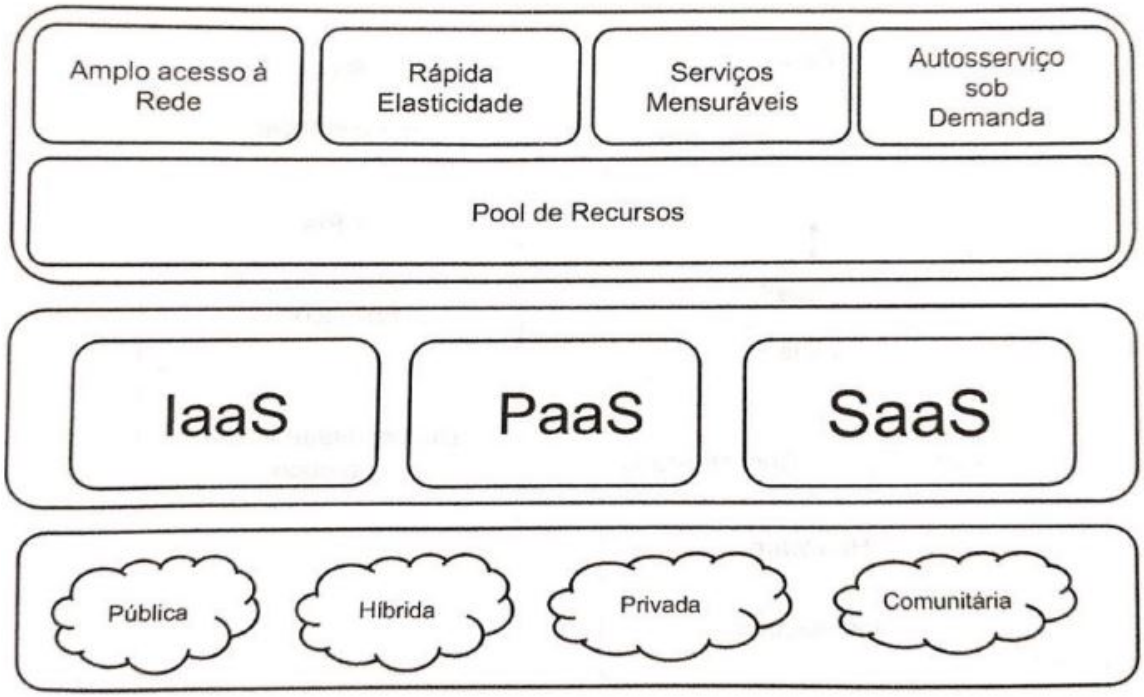

Figura 2.3 – Definição Computação em Nuvem (Fonte: [22])

2.5 Green IT / TI Verde

O Green IT ou TI Verde, surgiu pela necessidade de controlar os problemas ambientais causados pelos recursos não naturais e não renováveis que vieram com a evolução da tecnologia. Com uma demanda cada vez maior de processamento, qualidade e segurança trazidos pelo mercado competitivo, acaba sendo exigida uma constante adequação às novas tecnologias. Buscando redução de custos e melhoria de desempenho, a Tecnologia da Informação (TI) Verde trabalha questões relacionadas à eficiência no consumo de energia, provisionando e dimensionando recursos, efetuando controles sobre a operação e também cuidado com o meio ambiente[12].

Os Data Centers trabalham com equipamentos de diversos fornecedores, e com o passar dos anos, surgem novas tecnologias e os equipamentos precisam ser substituídos devido ao fim de seu ciclo de vida ou pela necessidade de entrada de novas tecnologias. Para sustentar as mudanças que ocorrem nos projetos de Data Center, é necessário o equilibro de três fatores: conhecimento das tendências tecnológicas, padronização e modularidade.

Não existe um conceito bem definido para a TI Verde, sendo tratada como um conjunto de ações aceitas pela comunidade global. Muitos gestores de tecnologia vem associando o conceito às novidades tecnológicas e também através das iniciativas para reduzir os custos energéticos e de refrigeração, além do estado vigente das operações de TI. Portanto, a TI Verde pode ser considerada um estudo e a prática de projetar, produzir, uti-

lizar e descartar computadores, servidores e subsistemas associados de maneira eficiente e eficaz, trazendo o mínimo ou sem impacto algum ao meio-ambiente[12].

Algumas empresas já abordam este conceito, pois além da preocupação com a preservação dos recursos naturais, saem ganhando com melhor eficiência das suas operações. A preocupação com o meio ambiente não está apenas relacionada ao descarte e mau uso. Outro ponto de interesse dos gestores de grandes Data Centers é a questão econômica e energética, pois os gastos necessários para manter a infraestrutura de TI com servidores, computadores, monitores e demais periféricos funcionando já representa a terceira maior fonte de consumo de energia dentro das grandes empresas[12].

3. TRABALHOS RELACIONADOS

Neste capitulo de trabalhos relacionados apresentaremos diversos modelos e arquiteturas de monitoramento térmico. Abordaremos Data Centers, Casas Inteligentes e consumo energético. Serão apresentadas similaridades e diferenças entre os projetos e possíveis metodologias que podem ser adotadas na continuidade do trabalho que será proposto.

3.1 Análise de Zonas Térmicas em Data Center Não-CRAC

Em [10] é apresentada a arquitetura MonTerDC. O sistema de monitoramento é voltado para Data Centers de pequeno e médio porte que adotam o modelo não-CRAC. Nestes ambientes os sistemas de refrigeração são formados por condicionadores de ar com poucos pontos de injeção de ar, diferentemente do modelo CRAC, onde existem dezenas de entradas/saídas direcionadas para o controle de temperatura. O CRAC trabalha através de um sistema de condução pressurizada de suas correntes de ar, enquanto o modelo de refrigeração não-CRAC (existente na maioria dos Data Centers espalhados pelo Brasil), possui uma má distribuição destas correntes de ar frio e quente, o que leva a formação de zonas térmicas indesejadas, que vão contra as normas de padronização de controle e temperatura em Data Centers.

A modelagem das zonas térmicas através do MonTerDC, acontece através de sensores de temperatura, juntamente do mapeamento da estrutura estática, localização dos equipamentos e com o fluxo de refrigeração formado pelas correntes de ar frio e quente. O sistema identifica as zonas onde se encontram as altas temperaturas e, mediante um simulador CFD (*Computational Fluid Dynamics*), é gerada uma imagem 3D das zonas térmicas em tempo real [10].

Dentro da sua arquitetura, o MonTerDC possui três módulos principais: Central de Coleta (C), Central de Processamento (CP) e o 3D Viewer. O funcionamento da arquitetura está ilustrado na Figura 3.1. Na Central de Coleta, encontram-se os sensores responsáveis pelo gerenciamento das zonas térmicas de temperatura, umidade e pressão atmosférica. A sua configuração é realizada de acordo com as especificações do administrador do ambiente [10].

A Central de Processamento trata os dados recebidos em tempo real, realizando uma associação dos mesmos à infraestrutura física do Data Center. Os dados tratados pelos sensores recebem uma tag e são armazenados em um banco de dados. Este módulo necessita da estrutura física do Data Center, juntamente da localização dos servidores e das saídas do sistema de refrigeração e seus sensores. Através das informações geradas,

o submódulo CFX gera o cenário atual do Data Center demonstrando as correntes de ar quente/frio, juntamente da intensidade destas correntes no espaço físico, mostrando as zonas térmicas com suas respectivas temperaturas [10].

O módulo 3D View transforma os dados em imagem. A visualização das zonas térmicas ocorre em tempo real e a sua arquitetura pode ser integrada tanto na nuvem quanto num Data Center local. Através da arquitetura MonTerDC o administrador de Data Center consegue monitorar e tratar zonas térmicas indesejáveis, evitando possíveis falhas nos equipamentos por conta de superaquecimento ou umidade do ar [10].

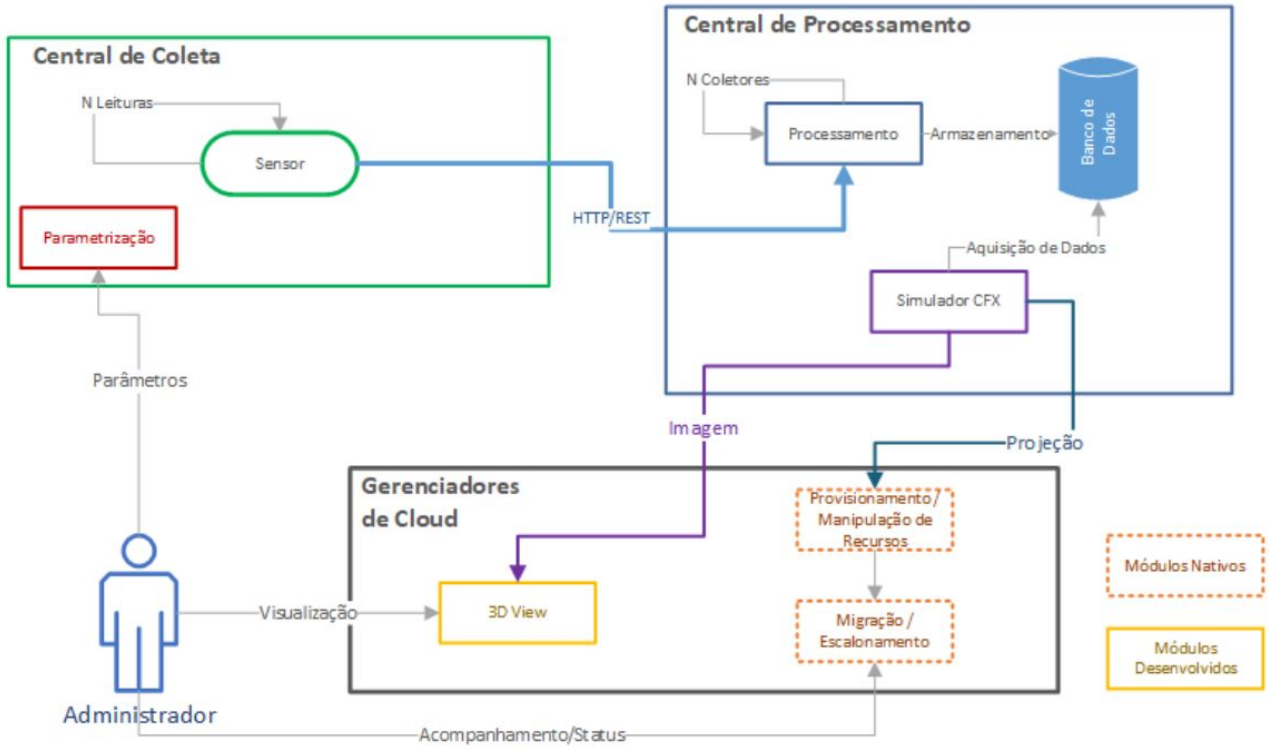

Figura 3.1 – Arquitetura MonTerDC (Fonte: [10])

3.2 Monitoramento e controle da climatização de Data Center via IOT

O artigo [5] apresenta um sistema de monitoramento remoto de temperatura e umidade utilizando uma central de processamento de dados em IoT, mas com seus atuadores controlando apenas a temperatura. Utilizando uma placa Arduino MKR WiFi 1010 responsável por receber a temperatura e umidade do ambiente, é construída uma rede de sensores conectados ao roteador local.

Através do controle da temperatura, é possível evitar possíveis intermitências no ambiente físico do Data Center. O processo de monitoramento e controle dos ativos de climatização de ar e dissipação de calor evitam a ocorrência de falhas no ambiente físico do

Data Center, tendo em vista que os servidores podem atingir altas temperaturas em questão de segundos, podendo causar a perda parcial ou total do equipamento caso ultrapasse o limite definido por seus fabricantes.

O processador existente na placa Arduino MKR WiFi 1010 é um Arm Cortex M0 SAM21 de 32 bits de baixo consumo. A conexão WiFi e Bluetooth ocorre através do módulo da u-blox. O chipset que opera na faixa de 2,4GHz é o NINA-W10. A criptografia dos dados acontece com a utilização do Microship ECC508. Para medir a temperatura e umidade do ambiente, foi utilizado um sensor modelo DHT22/AM2302, com poder de coletar temperaturas entre -40 e +80 graus Celsius, também realizando a coleta da umidade do ar entre 0 e 100%.

Como demonstração da arquitetura de utilização e aplicação IoT Cloud da placa MKR WiFi 1010 podemos verificar a Figura 3.2, sendo considerado um modelo genérico, pois a arquitetura necessita de ajustes para o seu correto funcionamento.

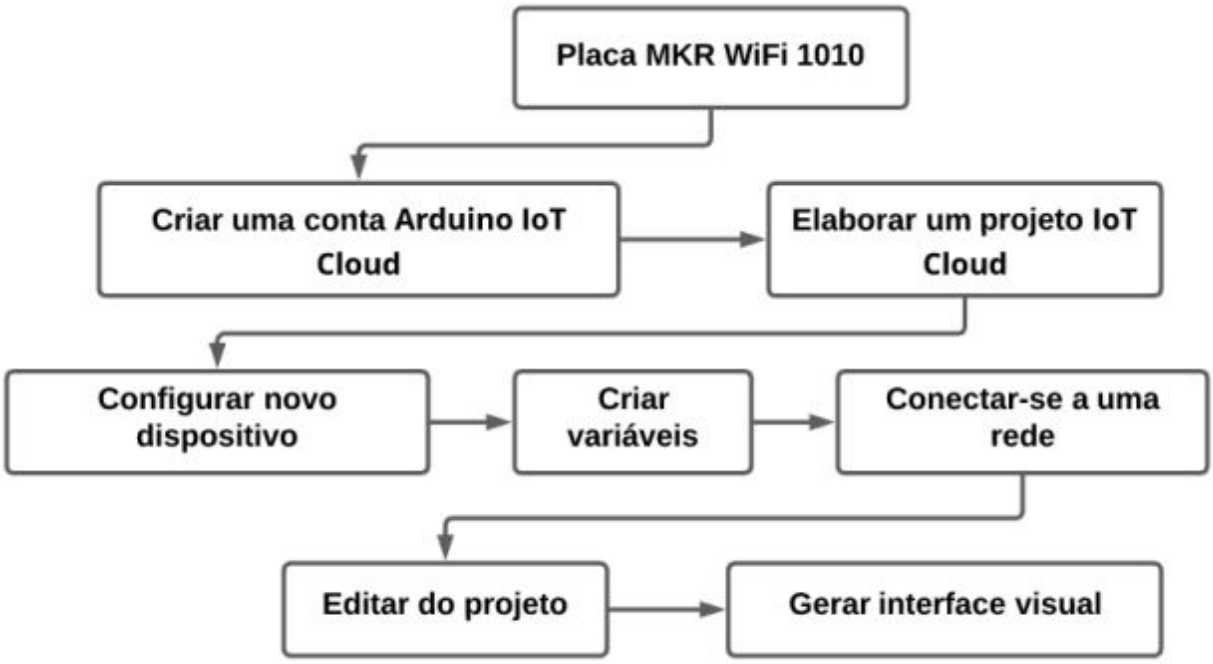

Figura 3.2 – Arquitetura Cloud - MKR WiFi 1010 (Fonte: [5])

As verificações de *status* acontecem a cada dois segundos. Se as temperaturas estão dentro da normalidade (até 27º), o atuador do Condicionador 1 mantém-se ligado, exibindo uma mensagem de temperatura normal. Essa mensagem é transmitida no *Dashboard* do Arduino IoT Cloud, juntamente com os dados coletados, que estão em constante atualização, de forma automática. Caso alguma intermitência na temperatura seja identificada (acima de 27º), o sistema assume que o Condicionador 1 está em falha e não está resfriando o ambiente. Com isso, aciona os módulos de atuadores para desligá-lo e, em paralelo, o Condicionador 2 é ativado, gerando uma mensagem de alerta para o usuário,

informando que Condicionador 1 foi desligado e encontra-se com defeito e que o Condicionador 2 está ativado e em pleno funcionamento.

3.3 Sistema de monitoramento de baixo custo para Data Center baseado em Internet das Coisas

Em [7] é apresentado um sistema de controle de temperatura em Data Center com um custo dezoito vezes menor que seus concorrentes. Para o desenvolvimento da ferramenta de baixo custo, utilizou-se um microcontrolador modelo *Orange Pi - Lite*, um módulo para captura de temperatura e um sistema de alerta de e-mails. Os itens citados tornaram viável a construção de um sistema eficiente e econômico para Data Centers de pequeno e médio porte.

O microcontrolador *Orange Pi* foi escolhido devida a possibilidade de incluir um sistema operacional de rede, o que aumenta a segurança e também a sua estabilidade em relação ao acesso remoto ao dispositivo. Mesmo sendo possível adicionar um módulo de rede nos modelos Arduíno, a necessidade de estar constantemente atualizando os aplicativos de rede disponíveis para garantir a segurança, fez com que o dispositivo não fosse escolhido.

O modelo *Orange Pi - Lite* possui memória RAM de 512 MB. O processador é Quad-core ARM 1.2GHz e também possui interface de rede via WiFi, o que permite a integração com a rede local sem a necessidade de um cabo de rede. No dispositivo foi instalado o sistema operacional Armbian, baseado na distribuição Ubuntu Server do sistema operacional Linux.

Para controle de umidade e temperatura foi utilizado o módulo DHT11. O modelo é composto por um sistema de medição de umidade e um termistor para temperatura. O termistor trata-se de um semicondutor sensível à temperatura, ou seja, quando ocorre o aumento da temperatura a sua resistência diminui. O dispositivo também possui um microcontrolador de 8 bits para fornecer dados ao DHT11. Sua arquitetura está apresentada na Figura 3.3.

A comunicação dos módulos com o microcontrolador é feita através do conjunto de pinos de entrada e saída que podem ser configurados através de softwares instalados dentro do sistema operacional instalado no dispositivo. Os dados são coletados através do pino "data". O algoritmo, implementado na linguagem C, envia uma requisição e recebe os dados em formato de pacotes de 40 bits. Estes dados são enviados para o microcontrolador e possibilita realizar o envio e recebimento de dados.

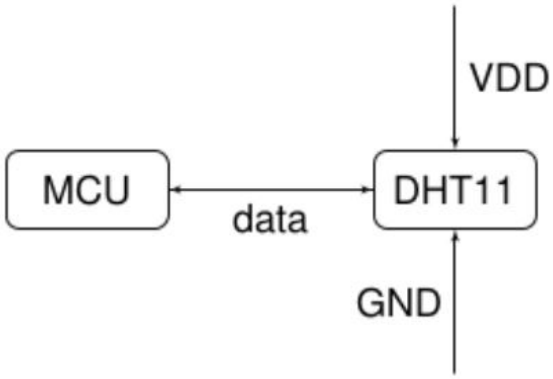

Figura 3.3 – Funcionamento do módulo DHT11 (Fonte: [7])

3.4 Monitoramento Inteligente do Consumo de Energia Elétrica em Residências Utilizando Recursos de IoT

Em [17] foi desenvolvido um estudo teórico e prático de um sistema de monitoramento de consumo energético residencial, recebendo as informações através do microcontrolador ESP32 que está conectado ao sensor SCT-013 usado na aquisição da corrente elétrica, mais um sensor transformador de potência ZMPT101B, que monitora a tensão elétrica. O ESP32 está integrado com a rede WiFi para ser capaz de fornecer históricos de consumo, tempo real de utilização, consumo monetário, qualidade da energia recebida e diversos outros fatores, tudo realizado de forma automatizada através das ferramentas de IoT e controlada remotamente de qualquer dispositivo móvel.

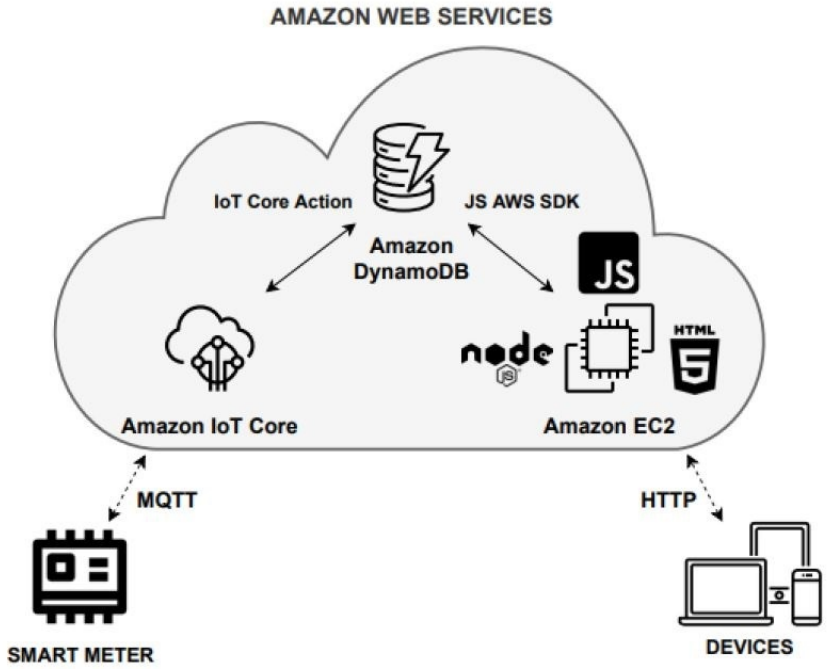

Figura 3.4 – Arquitetura de comunicação dos dispositivos locais com a nuvem(Fonte: [17])

Os dados adquiridos são armazenados na nuvem através da provedora de serviços AWS. O seu funcionamento baseia-se com o ESP32 enviando as variáveis através do protocolo MQTT para o serviço da IoT Core. As métricas definidas pelo usuário dentro do broker (IoT Core) são armazenadas no banco de dados Amazon DynamoDB. As informações contidas no banco de dados são enviadas para uma aplicação Web desenvolvida em Javascript e HTML, ferramenta que se encontra hospedada numa máquina virtual no serviço da Amazon EC2, tendo um servidor Node.js para gerenciamento. Sua arquitetura está apresentada na Figura 3.4.

Figura 3.5 – Tela das funcionalidades Smart Meter Energy v1.0 (Fonte: [17])

A aplicação desenvolvida recebeu o nome de Smart Meter Energy v1.0, disponível para todos os dispositivos que possuem acesso à Internet. A ferramenta na sua tela principal dispõe de um menu de opções para visualizar os registros dos dados recebidos pelos sensores, sendo estas informações de: tempo real, dados do dia atual, mês corrente e meses anteriores. Com a integração destes equipamentos, foi obtido com sucesso referências reais e precisas sobre a utilização da energia elétrica de determinada residência, possibili-

tando um maior controle do ambiente e promovendo a diminuição do consumo energético do local.

3.5 Room Temperature Control and Fire Alarm/Suppression IoT Service Using MQTT on AWS

Em [11], foi utilizado o software Mosquitto instalado numa máquina virtual hospedada na *Amazon Web Services* como MQTT broker. A aplicação desempenha a função de uma central que recebe os dados enviados por Arduinos que se encontram conectados à rede WiFi do ambiente monitorado. Estes dispostivos estão constantemente recebendo informações dos diversos sensores que monitoram e controlam a temperatura e também o ambiente em caso de incêndio. O local onde ocorreram as simulações foi desenvolvido pelo autores (Figura 3.6). Por mais que o monitoramento ocorra em um ambiente simulado, sua metodologia atenderia tranquilamente a realidade de diversos Data Centers.

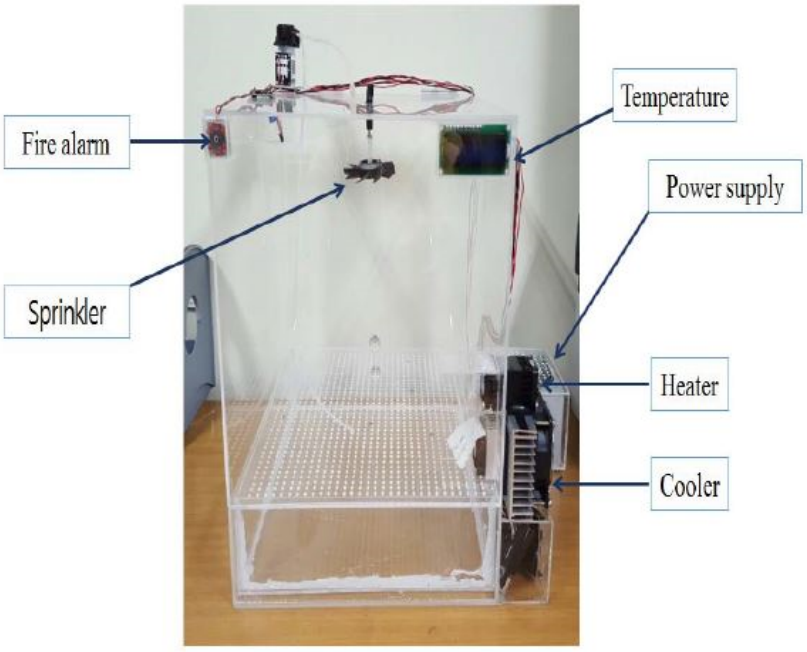

Figura 3.6 – Ambiente de simulação dos eventos (Fonte: [11])

O Mosquitto é uma ferramenta gratuita, motivo pelo qual foi o escolhido pelos autores que desenvolveram o estudo. O software utiliza o protocolo MQTT para sua comunicação com os demais aparelhos. Com a junção destas tecnologias, foi possível criar um cenário de *smart home* de baixo custo sem perder a sua qualidade de monitoramento e funcionando de seus sistemas, mostrando a eficiência da junção entre o MQTT e os serviços oferecidos pela AWS. O broker MQTT trata as informações recebidas e é capaz de transmiti-las para outros diversos dispositivos, como celulares ou computadores.

A conexão entre a aplicação e os Arduinos ocorre através do IP público existente na EC2 *(Amazon Elastic Compute Cloud)*. Este IP é cadastrado em um servidor DNS para ser transformado numa URL nominada de iotwarrior.mynimbus.xyz. Para manter um custo baixo, não foi atrelado um endereço publico fixo, ou seja, toda vez que a máquina é reiniciada, o endereço antigo é perdido, devendo ser realizado novamente o direcionamento para o DNS existente.

Conforme descrito na Figura 3.7, dois Arduinos possuem a função de controlar os sensores de temperatura e de ar condicionado. O outro dispositivo fica responsável pelo controle e disparo de alarme em caso de incêndio. Estes dados são recebidos pelo *broker* que, através das métricas estabelecidas pelo usuário, pode disparar ou não uma mensagem para os dispositivos inscritos no recebimento de alertas. Através dos dispositivos móveis, o administrador do ambiente pode monitorar e realizar ajustes determinando novas métricas de temperatura, quanto nos chuveiros automáticos utilizados em caso de incêndio.

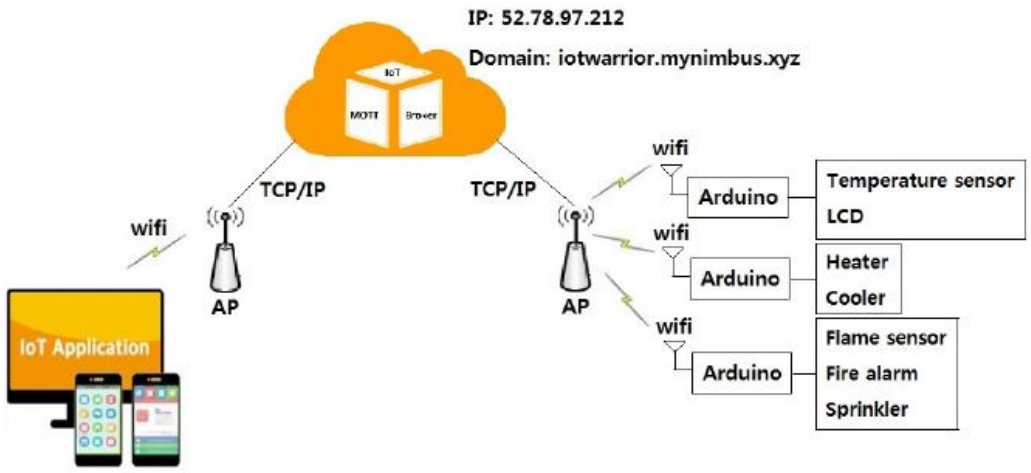

Figura 3.7 – Funcionamento do módulo DHT11 (Fonte: [11])

3.6 Análise dos trabalhos relacionados

A arquitetura apresentada pelo [10] dispõe de uma estrutura modelo CRAC. Os dados coletados por sensores IoT irão auxiliar o administrador de Data Center a tomar as melhores decisões com base nas métricas parametrizadas pelo próprio usuário. Através de uma aplicação hospedada na nuvem, em conjunto de um banco de dados, o usuário poderá buscar gráficos com históricos de temperaturas, realizar estudos e analisar possíveis picos de temperaturas, receber alertas de mudanças ou instabilidade em tempo real, através de e-mail, tablet ou smartphone.

No artigo [5], o sistema de monitoramento desenvolvido mostra com clareza o modo de comunicação entre os dispositivos Arduinos com os sensores inteligentes. Porém,

mantém o controle dos dados na arquitetura local, o que acaba consumindo recursos computacionais que poderiam estar sendo melhor utilizados. Caso o projeto fosse hospedado numa infraestrutura de nuvem, teríamos uma maior segurança do histórico dos dados, além de uma maior escalabilidade em caso de acréscimos de outros dispositivos de controle de ambiente, possibilitando também uma maior segurança do Data Center, pois em caso de violação da estrutura, dificilmente as invasões teriam acesso às informações contidas no centro de dados local.

O trabalho desenvolvido em [17] apresenta o controle de energia através dos dispositivos IoT, mesmo sendo um monitoramento diferente do trabalho proposto, os métodos adotados para seu avanço utilizam toda sua arquitetura com hospedagem na nuvem, utilizando recursos computacionais importantes e modernos como os serviços da IoT Core e de banco de dados DynamoDB. Como exemplo da importância deste artigo, pode-se citar a ferramenta desenvolvida para visualização dos dados energéticos que também poderia ser desenvolvida para realizar o monitoramento de temperatura dos Data Centers.

O controle de temperatura apresentado em [11] adota uma metodologia baseada no envio dos dados para o serviço de máquinas virtuais da Amazon Web Services, ou seja, no servidor criado encontra-se o programa Mosquitto. A adoção do serviço hospedado no EC2, implica na necessidade de monitorar o funcionamento da máquina virtual, pois o *broker* que trata os dados recebidos encontra-se nesta máquina. Em caso de falha em seu funcionamento, todo o monitoramento será afetado, causando a indisponibilidade do serviço.

Com os dados processados pelo *software* é possível armazenar as informações em diversos bancos de dados da própria AWS, Uma das desvantagens do sistema apresentado em [11] é o uso de um IP público variável, o que pode impactar na disponibilidade do serviço. Realizando uma conta rápida de valores na calculadora da provedora de nuvem, seu custo seria de 3,75 USD mensais, com seu custo anual em 43,80 USD. Com este investimento, seria dispensada a necessidade de realizar ajustes toda vez que a EC2 é reinicializada.

A comparação e benefícios dos trabalhos relacionados está descrita na Tabela 3.1 abaixo:

Artigo	Contribuição	Monitoramento	Alertas	Hospedagem	Sensor	Dispositivo	Armazenamento	Visualização
[10]	Monitora pequenos e médios Data Centers, modelos não-CRAC	- Correntes de ar - Zonas térmicas indesejadas - Consumo energético	- Variação de temperatura - Carga de energia	Open Stack	LM35DZ	-	Banco de dados não específicado	3D Viewer
[5]	Monitora e controla a temperatura em pequenos Data Centers	- Temperatura - Umidade do ar	- Temperatura fora do padrão - Umidade do ar	Arduino Cloud	DHT22	MKR-1010	Arduino Cloud	Arduino Cloud
[7]	Monitora a temperatura de data centers, utilizando ferramentas de baixo custo	- Temperatura	- Temperatura fora do padrão	Local Sistema Operacional da placa	DHT11	Orange Pi Lite	Disco local	Aplicação PHP Própria
[17]	Monitoramento de consumo energético residencial com sistema IoT	- Corrente elétrica - Valores estimados de consumo	- Consumo energético	AWS	SCT-013 ZMPT101B	ESP32	DynamoDB	Aplicação JS Própria
[11]	Monitoramento de temperatura, controle/supressão de incêndio	- Temperatura - Sprinkler - Incêndio	- Temperaturas fora do padrão - Falha dos Sprinkler - Alarme de incêncio	AWS	Não especificado	Não especificado	AWS - EC2	Não especificado

Tabela 3.1 – Tabela dos benefícios e funcionalidades dos trabalhos relacionados.

4. PROJETO E DESENVOLVIMENTO

Neste capítulo apresentaremos algumas ferramentas utilizadas para realizar o monitoramento de temperatura em Data Centers. Através de um estudo mais profundo, foi desenvolvido um protótipo visando simular um sensor que realiza a medição e o envio de temperatura para um microcontrolador também simulado, com os dados obtidos sendo exibidos numa aplicação web.

4.1 Formulação do problema

Todos os ambientes físicos de Data Center são de suma importância para suas organizações. Devido a criticidade deste ambiente, sua temperatura deve ser monitorada 24 horas por dia. O armazenamento destas informações permite a equipe de gerenciamento encontrar padrões e realizar estudos em casos de aumento desproporcional de calor ou processamento. A fim de realizar esta tarefa, podemos optar pela instalação de sensores de captação das temperaturas do local onde os dispositivos se encontram, interligando este dispositivo com um microcontrolador, podemos tratar estes dados e transmiti-los para outras diversas aplicações de controle e gerenciamento.

4.2 Definição do objetivo geral e específicos

O objetivo deste trabalho é desenvolver uma arquitetura de monitoramento de temperatura em ambientes de Data Center, utilizando a Computação em Nuvem. Para alcançar esse objetivo, foram empregadas as técnicas atuais de mercado, como o Terraform e o Ansible, para provisionar e configurar o ambiente na nuvem de forma eficiente.

Como objetivo específico buscou-se a implementação de um ambiente escalável e de baixo custo, focando em uma fácil integração com dispositivos IoT e sensores de temperatura. Para garantir a viabilidade financeira do projeto, optou-se pelo uso do protocolo MQTT, que requer menos recursos computacionais em comparação a outras opções.

Dessa forma, o trabalho visou criar uma arquitetura de monitoramento que seja eficaz, escalável e de baixo custo para ambientes de Data Center. A integração com dispositivos IoT e sensores de temperatura é um aspecto importante, garantindo que o sistema seja capaz de coletar dados precisos e relevantes. O uso do protocolo MQTT contribui para otimizar o consumo de recursos, tornando o projeto mais acessível em termos de custo e desempenho.

4.3 Metodologia

A metodologia deste trabalho se baseou no estudo de artigos científicos e casos de sucesso para o monitoramento térmico de Data Centers. Com base nessas informações, uma arquitetura simulada de servidores foi desenvolvida no ambiente da Amazon Web Services (AWS) para realizar o monitoramento das máquinas virtuais. Além disso, um sensor de temperatura emulado foi utilizado, desenvolvido em Python, para simular o envio de dados de temperatura, como se fossem sensores reais de servidores.

A arquitetura foi implementada utilizando os serviços disponibilizados pela AWS, eliminando a necessidade de provisionar uma infraestrutura local para gerenciar as informações enviadas e recebidas pelos dispositivos simulados. O objetivo era manter as funcionalidades existentes e adicionar novas funções conforme necessário.

Ao utilizar os serviços de nuvem, foram obtidas vantagens como escalabilidade, confiabilidade e flexibilidade. Recursos como servidores virtuais, armazenamento, serviços de banco de dados e ferramentas de monitoramento disponibilizadas pela AWS foram aproveitados para construir uma solução eficiente e eficaz de monitoramento térmico.

O sensor emulado de temperatura, desenvolvido em Python, foi responsável por simular a coleta e o envio de dados de temperatura dos servidores. Esses dados foram processados e armazenados na infraestrutura da AWS, permitindo a análise e o monitoramento em tempo real.

Com essa abordagem, foi possível criar um ambiente virtualizado e escalável de monitoramento térmico de Data Centers, aproveitando as capacidades da AWS. Isso resultou em uma solução de monitoramento eficiente e econômica, eliminando a necessidade de investimentos em infraestrutura local e permitindo a incorporação de novas funcionalidades conforme identificadas nos estudos de referência.

4.4 Estudo de caso utilizando um sensor simulado

Após a análise dos trabalhos relacionados (apresentados no Capítulo 3), em busca de um melhor entendimento sobre o funcionamento do sistema de monitoramento, foi desenvolvido um protótipo para realizar à análise do problema. Este protótipo visa emular um sensor que capta e realiza o envio de temperaturas para um microcontrolador, cujo objetivo é tratar e transmitir as informações para um banco de dados hospedado em uma máquina virtual através da ferramenta VirtualBox. As informações salvas são consultadas por uma aplicação também virtualizada que pode exibir os dados em forma de gráficos ou listando-as de maneira automatizada, facilitando a visualização para o usuário final. A arquitetura do sistema está apresentada na Figura 4.1.

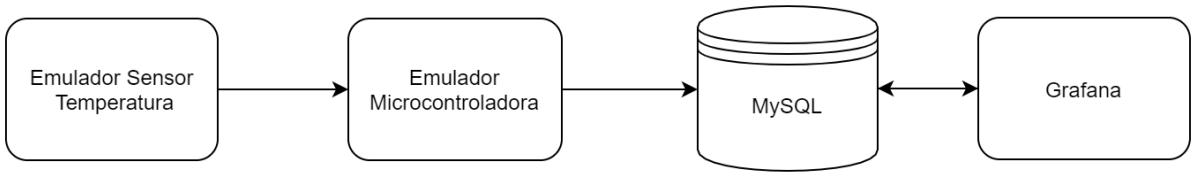

Figura 4.1 – Arquitetura do protótipo (Fonte: Autor)

Para o desenvolvimento dos emuladores foi utilizado um micro-framework web chamado *flask* que oferece um modelo simples para o desenvolvimento de aplicações web. A linguagem Python foi utilizada para a criação dos simuladores. Sua utilização é de código aberto e é bastante utilizada em diversas áreas, como: Data Science, Machine Learning, desenvolvimento web, desenvolvimento de aplicativos, automação de scripts, entre outros.

O VirtualBox é um *software* de virtualização da empresa Oracle. Seu funcionamento ocorre sobre um sistema operacional já instalado. Trata-se de uma ferramenta multiplataforma, portanto, pode ser utilizado no Windows, distribuições Linux ou MacOS X. Através da utilização do hardware de seu hospedeiro, o VirtualBox permite o provisionamento de memória, *clusters* e outros vários recursos que atendem as mais diversas demandas de virtualização.

O MySQL é um sistema de gerenciamento de banco de dados (SGBD) relacional. Ele surgiu com o intuito de atender a demanda de pequenas e médias aplicações. Devido a sua popularização e implantações de melhorias, passou a atender grandes aplicações, dispondo de uma gama maior de recursos que muitos concorrentes. Essa adoção em massa lhe concedeu o reconhecimento por muitos especialistas como o banco de dados de código aberto com maior capacidade para concorrer com outros programas de mesma finalidade de código fechado, como o SQL Server da Microsoft e Oracle [15].

O Grafana é uma aplicação de código aberto utilizada para análise e monitoramento de sistemas online. Através das métricas obtidas, pode-se gerar diversos tipos de painéis para avaliar o desempenho de um servidor, banco de dados ou programa web. Por ser uma ferramenta de código aberto, é possível desenvolver *plugins* para integrá-lo com outras fontes de dados. Atualmente, é possível interligar o Grafana com o Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL e outros diversos serviços.

Para o desenvolvimento do protótipo, foi criado um código na linguagem Python utilizando a biblioteca *"flask"*. Após importá-la, podemos enviar requisições HTTP trabalhando com métodos GET e POST, ou seja, o microcontrolador recebe as requisições do sensor. Ao receber os dados, o microcontrolador realiza o devido tratamento e encaminha as informações para o banco de dados onde são armazenadas. O banco de dados encontra-se numa máquina virtual criada através do *software* de virtualização VirtualBox.

O servidor de banco de dados foi configurado no sistema operacional Ubuntu/Bionic, utilizando a versão 5.7 do MySQL. Após a instalação, foi criado um banco de dados chamado "temperature". Dentro desse banco de dados, foi criada uma tabela denominada "server_temperature"com quatro colunas distintas.

A primeira coluna, chamada "codigo", é configurada para auto-incrementar à medida que recebe dados de temperatura da microcontroladora. Essa coluna serve como um identificador único para cada registro de temperatura.

A segunda coluna, denominada "temperature", armazena os valores de temperatura em formato inteiro. Ela registra as temperaturas capturadas pelos sensores.

A terceira coluna, chamada "date-timestamp", armazena a data e hora exatas em que as informações de temperatura são recebidas e registradas no banco de dados.

Essa estrutura de banco de dados permite armazenar e acessar as informações de temperatura de maneira organizada, facilitando a análise e o monitoramento das condições térmicas do ambiente do Data Center.

Para uma melhor visualização dos dados, criamos outra máquina virtual, também utilizando o VirtualBox com o sistema operacional Ubuntu/Bionic, com a função de executar o Grafana. A ferramenta nos permite enviar alertas quando conectado a alguma fonte de dados. Para realizar a comunicação entre o MySQL e o Grafana, utilizamos o plugin disponível na ferramenta, onde se faz necessário a criação de um novo usuário dentro do banco de dados com a função de leitura das informações nele contidas.

Buscando uma simulação mais real, foi realizada a separação física do banco de dados, da aplicação do Grafana, pois muitas empresas utilizam a ferramenta com acesso externo através da Internet, pois possibilita à visualização das métricas de qualquer local. Como existe uma porta aberta para Internet, existe a possibilidade de invasores buscarem brechas e falhas no sistema, o que acarretaria um acesso não autorizado às informações contidas nesta máquina, e caso tivéssemos a união dos dois serviços na mesma máquina virtual, acarretaria no acesso às informações contidas no banco de dados MySQL. A versão final do protótipo está ilustrada na Figura 4.2

4.5 Modelagem utilizando Computação em Nuvem

Após o desenvolvimento do protótipo local, foram criados modelos de software utilizando Computação em Nuvem, mais especificamente, utilizando os serviços do provedor Amazon Web Services. Como a solução pode ser desenvolvida usando diferentes serviços da AWS, foram criados 3 modelos. Para cada um são apresentados os serviços e ferramentas utilizadas, forma de implementação, possíveis alertas e atuações em determinada ocorrência, além de uma tabela (3.1), para demonstrar com clareza as técnicas abordadas.

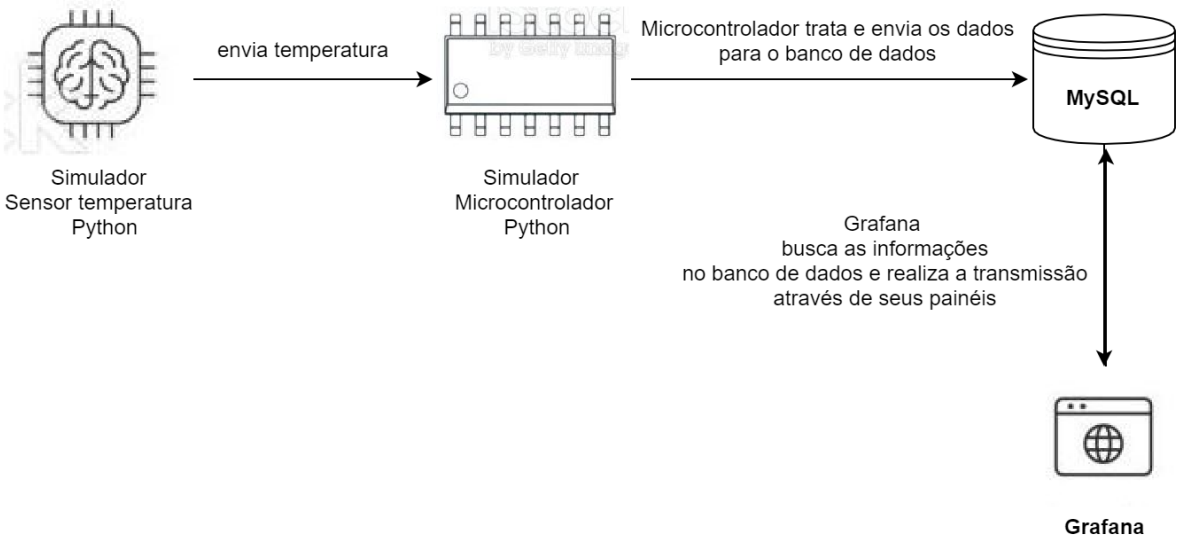

Figura 4.2 – Arquitetura final do protótipo (Fonte: Autor)

4.5.1 Modelagem na nuvem M1 - Utilizando IoT Core, Funções Lambda e DynamoDB

Após estudos de trabalhos relacionados, encontrou-se a possibilidade de interligar os sensores no serviço da IoT Core (Figura 4.3). Este serviço disponibilizado pela Amazon Web Services possibilita a comunicação de dispositivos IoT existentes no ambiente local com a nuvem. Para estabelecer esta conexão, é necessário cadastrar o seu dispositivo dentro do serviço referido. No final de sua configuração, é recebido um certificado gerado pela AWS juntamente das chaves pública e privada para estabelecer a conexão. Em posse destes três itens deve-se realizar a configuração dentro do microcontrolador, acrescentando as chaves no código existente para termos a confiança entre o dispositivo e o serviço do provedor de nuvem. [1]

Com o dispositivo conectado, é necessário estabelecer a ligação entre a AWS IoT Core com o AWS Lambda. Este serviço é conhecido como computação sem servidor *(serverless)* orientado a eventos, sendo compatível com qualquer tipo de aplicação, dispensando a necessidade de provisionar ou gerenciar servidores. Seguindo a documentação do próprio provedor do serviço, para realizar esta conexão, deve-se criar uma *rule* dentro do serviço da IoT Core para que seja possível invocar uma função Lambda quando determinada ação acontecer. Basicamente estas *rules* permitem que os dispositivos interajam com os demais serviços disponíveis pela Amazon Web Services. A comunicação interna ocorre através do protocolo MQTT.

A partir de uma função Lambda, podemos enviar as informações recebidas pelo sensor de temperatura, para o serviço de banco de dados NoSQL chamado de DynamoDB,

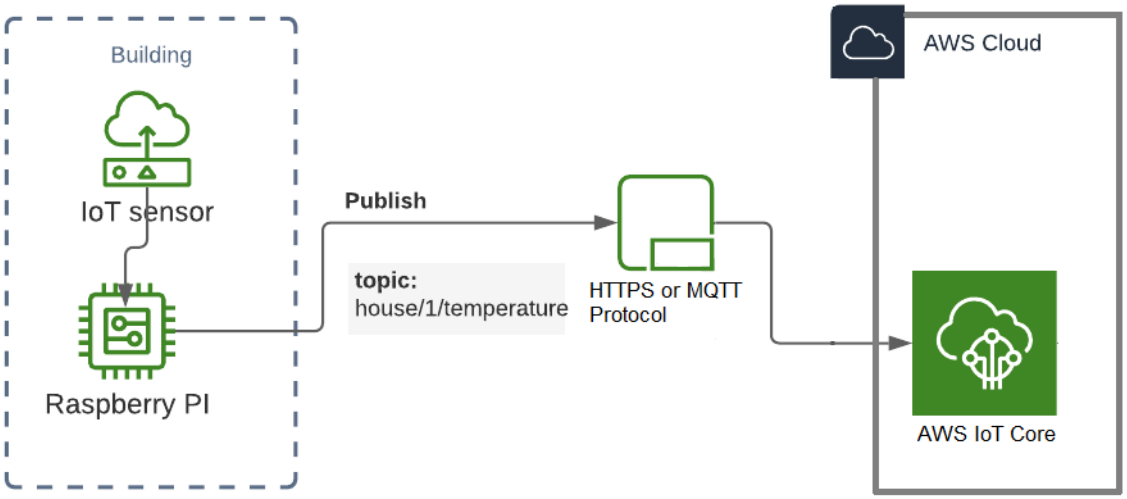

Figura 4.3 – Exemplo de funcionamento - Arquitetura IoT Core(Fonte: Autor)

gerenciado pela *Amazon Web Services*. Como todos os outros serviços de banco de dados oferecidos, o DynamoDB não necessita realizar o provisionamento de uma infraestrutura para utilizá-lo. Seu funcionamento é orientado a documentos ou chave-valor, onde você pode armazenar um arquivo JSON ou CSV, por exemplo.

Para o recebimento de alertas via SMS ou e-mail, pode-se utilizar o SNS *(Amazon Simple Notification Service)*. Este serviço de mensagens pode ser utilizado tanto no modelo aplicação para aplicação (A2A) quanto de aplicação para pessoa (A2P). Através da plataforma, o administrador de Data Center tem a possibilidade de receber notificações referente a falhas no sistema de refrigeração em diversos dispositivos, facilitando o gerenciamento e pronto atendimento em caso de uma eventual necessidade de intervenção humana no sistema.

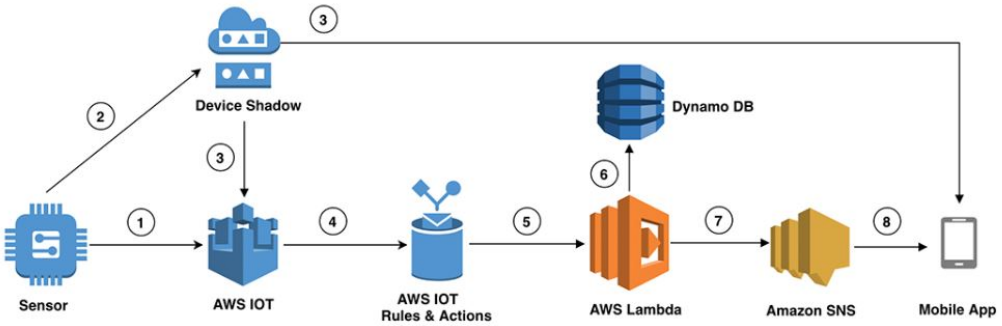

Figura 4.4 – Exemplo de computação *serverless* com IoT (Fonte: [6])

A arquitetura está ilustrada na Figura 4.4, devendo ser considerada apenas como um modelo genérico utilizado para um melhor entendimento do sistema aqui descrito, pois

o seu desenvolvimento real pode necessitar de alterações no seu modo de operação, visando atender as particularidades que cada estrutura física de Data Center dispõe, visando alcançar um gerenciamento mais completo e seguro do ambiente.

O levantamento de custos na utilização dos serviços mencionados foi realizada na própria calculadora de preços da AWS e podem ser visualizados na Tabela 4.1. Os valores são estimados, podendo ocorrer variações conforme a sua utilização e escalabilidade

Serviço	Backup	Região	Quantidade	Custo Mensal	Custo Anual
IoT Core	não aplica	Norte da Virgínia	1 dispositivo	1,00 USD	12,00 USD
Lambda	não aplica	Norte da Virgínia	2 funções	4,74 USD	56,88 USD
DynamoDB	Sim	Norte da Virgínia	1 tabela	46,89 USD	562,68 USD
SNS	não aplica	Norte da Virgínia	1 dispositivo móvel - 2 emails	0,50 USD	6,00 USD
				53,13 USD	**637,56 USD**

Tabela 4.1 – Tabela de custos estimados para utilização da Modelagem M1.

4.5.2 Modelagem na nuvem M2 - Utilizando o Software Mosquitto, Amazon Aurora e Software Grafana

Outra estratégia de arquitetura de monitoramento de temperatura utilizou-se dos serviços da Amazon Web Services (AWS) para permitir a comunicação dos dispositivos IoT com a nuvem, utilizando o software Mosquitto. Os dados recebidos e processados pela aplicação foram enviados para o serviço de banco de dados Amazon Aurora *(Relational Database Service)*, e o software Grafana foi utilizado para visualizar o histórico das temperaturas em painéis informativos.

Para essa implementação, foram utilizadas duas máquinas virtuais hospedadas na EC2 (Elastic Compute Cloud) e uma instância RDS de banco de dados MySQL. Essa escolha dos serviços da AWS permitiu alcançar um modelo de arquitetura de baixo custo, mantendo a qualidade e a segurança operacional do sistema.

O Eclipse Mosquitto é uma ferramenta de código aberto sendo um intermediário de mensagens que utiliza o protocolo MQTT para comunicação. O sistema trabalha como *broker*, realizando o direcionamento das requisições recebidas para outras ferramentas ou serviços. Por não exigir muito poder computacional e cumprir bem a sua tarefa, ele é bastante utilizado e adequado para todos os tipos de infraestrutura, sejam elas grandes Data Centers ou pequenos laboratórios. Esta arquitetura também dispõe de uma biblioteca na linguagem de programação C para implementação dos MQTTT clients, conhecidos como *mosquitto_pub* e *mosquitto_sub* sendo os publicadores e subscritos para recebimento das informações (Figura 2.2).

O Amazon Aurora é um serviço de banco de dados relacional, escalável e de fácil configuração oferecido pela AWS. A ferramenta suporta uma vasta variedade de mecanis-

mos existentes nos bancos de dados que ajudam o administrador a armazenar e organizar os dados, além de praticamente automatizar as tarefas de gerenciamento de dados, como migrações, backups, recuperação e aplicações de plugins, dispensando a necessidade de provisionar uma máquina física ou virtual para o seu funcionamento. A arquitetura do modelo de atuação está presente na Figura 4.5.

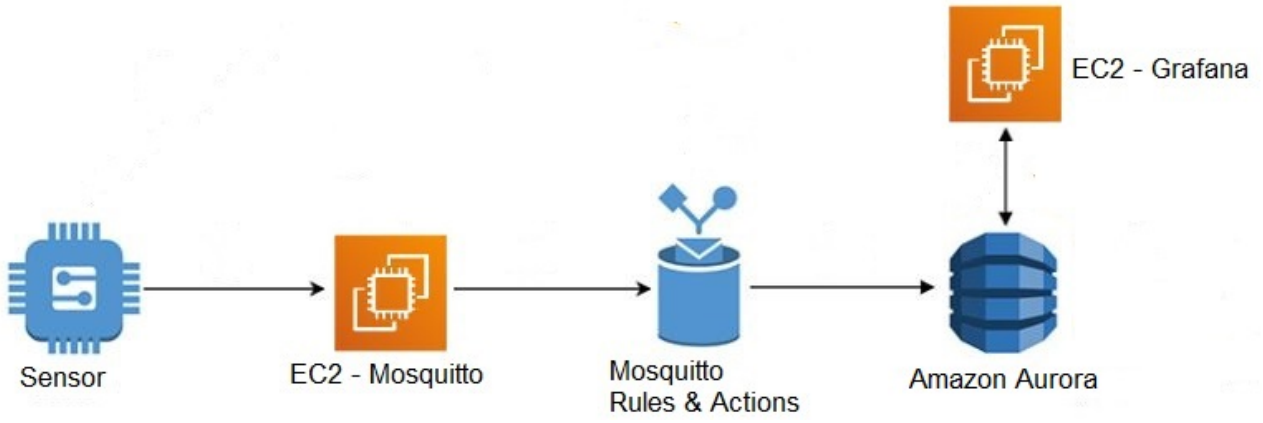

Figura 4.5 – Modelagem Arquitetura M2 (Fonte: Autor)

A conexão entre a aplicação Mosquitto e os dispositivos que recebem os dados dos sensores é realizada através do IP público existente na EC2. Este IP pode ser cadastrado em um serviço de DNS para ser acessado via uma URL. Para manter um custo baixo, não se faz necessário atrelar um endereço público fixo, ou seja, toda vez que a máquina for reiniciada, o endereço antigo será perdido, devendo ser realizado novamente o direcionamento para o DNS existente.

As estimativas de custo para o funcionamento das duas instâncias de máquina virtual e o serviço de banco de dados RDS foram criadas utilizando a caluladora da própria AWS, baseando-se num modelo de operação 24x7, durante o período de um ano. Os resultados podem ser vistos na Tabela 4.2.

Serviço	Modelo	Disco	Backup	Região	Quantidade	Custo Mensal	Custo Anual
Amazon EC2	t2.micro	10 GB - Un	Não	Norte da Virgínia	2 instâncias	12,51 USD	150,12 USD
Amazon Aurora	db.t2.micro	30 GB	Sim	Norte da Virgínia	1 instância	18,71 USD	374,64 USD
						31, 22 USD	**524,76 USD**

Tabela 4.2 – Tabela de custos estimados para utilização da Modelagem M2.

4.5.3 Modelagem na nuvem M3 - Utilizando IoT Core, Amazon Timestream e Amazon Managed Service for Grafana

Para o desenvolvimento desta arquitetura, é utilizado o banco de dados de séries temporais chamado de Amazon Timestream. Seguindo o modelo dos outros bancos de dados disponibilizados pela AWS, a sua arquitetura é escalável e sem servidor, ideal para diversos tipos de aplicações, como Internet das Coisas. Sua velocidade pode ser até mil vezes mais rápida que a de um banco de dados relacional e seu funcionamento ocorre via funções analíticas de séries temporais que ajudam o usuário a identificar tendências e padrões nas temperaturas recebidas.

Os sensores de temperatura enviam seus dados para o dispositivo que está conectado à rede WiFi do ambiente monitorado. Após a emissão do certificado na IoT Core e realizada a devida configuração no código-fonte do microcontrolador, ocorre a conexão entre os dispositivos. Os dados são transmitidos para o provedor de nuvem através do protocolo MQTT. O tratamento da informação é com base nas métricas estabelecidas dentro das *rules* do serviço. Uma destas regras é o envio dos dados para armazenamento no banco de dados, e caso exista alguma alteração nas variáveis de temperatura, como um aumento excessivo de calor, uma mensagem é disparada para os dispositivos inscritos alertando os responsáveis pelo local.

O *Amazon Managed Service for Grafana* é a disponibilização da aplicação já conhecida, porém, dentro dos serviços da AWS. A vantagem de utilizá-la é que não é necessário realizar o provisionamento de uma instância de máquina virtual, nem realizar a instalação do programa, sendo necessário apenas configurar as suas métricas e painéis. A sua vantagem é evitar a necessidade de manutenções na máquina e atualizações do produto, pois estas funções ficam a cargo da empresa que disponibiliza a sua utilização. A demonstração do funcionamento completo da arquitetura pode ser analisada na Figura 4.6.

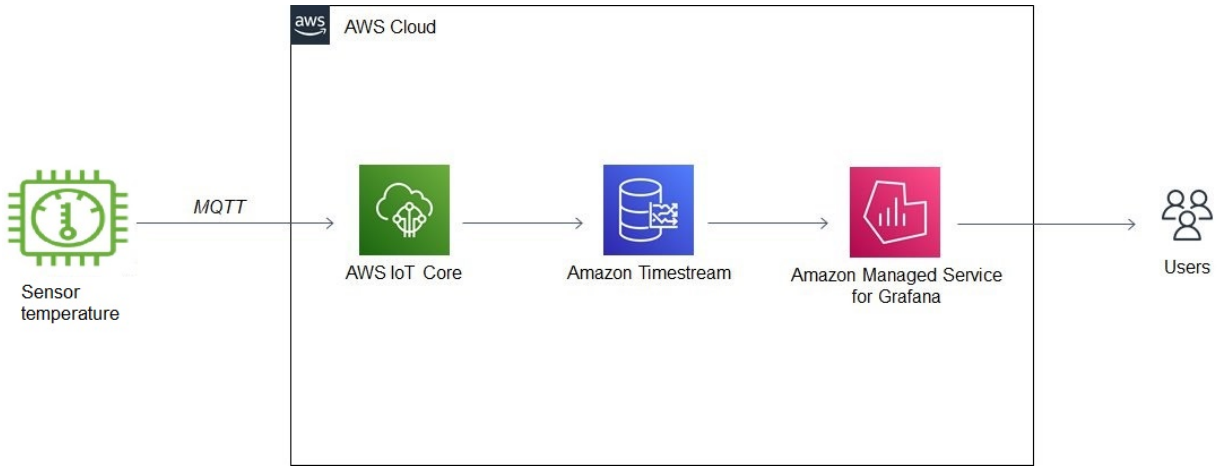

Figura 4.6 – Modelagem Arquitetura M3 (Fonte: Autor)

O levantamento de custos na utilização dos serviços mencionados utilizando a calculadora de preços da AWS pode ser visualizado na Tabela 4.3. Os valores são estimados, podendo ocorrer variações conforme a sua utilização e escalabilidade.

Serviço	Backup	Região	Quantidade	Custo Mensal	Custo Anual
IoT Core	não aplica	Norte da Virgínia	1 dispositivo	1,00 USD	12,00 USD
Timestream	sim	Norte da Virgínia	1 tabela	147,56 USD	1.770,72 USD
Grafana	não aplica	Norte da Virgínia	1 usuário - 3 dispositivos	24,00 USD	288 USD
				172,56 USD	**2.070,72 USD**

Tabela 4.3 – Tabela de custos estimados para utilização da Modelagem M3.

4.6 Proposta de Solução

Analisando a complexidade e custo dos modelos de solução usando Computação em Nuvem apresentados, foi escolhido o modelo 4.5.2. Este modelo utiliza duas máquinas virtuais providas pelo serviço EC2, uma instância de banco de dados RDS-MySQL e o serviço de notificações SNS. Através destes serviços, será realizada uma simulação de diversos sensores enviando informações referentes a um ambiente simulado. A arquitetura do modelo está escrita na Figura 4.7.

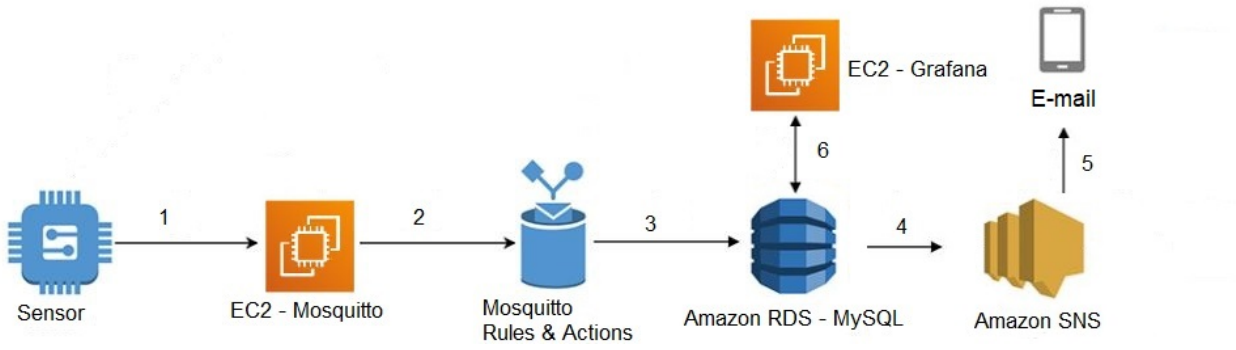

Figura 4.7 – Arquitetura proposta de solução (Fonte: Autor)

O software utilizado como broker para tratamento das regras e ações a serem tomadas será o Mosquitto, hospedado em uma das máquinas virtuais com o sistema operacional Ubuntu/Bionic, que, através do protocolo MQTT, receberá as temperaturas dos sensores. A temperatura será armazenamento no banco de dados, que em tempo real, está sendo transmitido no *dashboard* do Grafana. O serviço de mensagem SNS, que estará encarregado de avisar o administrador caso venha ocorrer alguma instabilidade na instância de banco de dados RDS.

A instância de banco de dados será chamado de "temperatura" e será utilizada apenas uma tabela com o nome de "temperatura_servidor" que possuirá seis colunas, uma

chamada "codigo", que auto-incrementa conforme recebe a temperatura através das ações impostas no Mosquitto, outra chamada "temperatura", que armazena as temperaturas dentro de um int, outra coluna "dispositivo", que será uma string, para armazenamento do nome do dispositivo, "descricao", referente ao local onde ele se encontra instalado e possíveis coordenadas e por fim, um "date-timestamp", que armazena a data e hora do recebimento das informações.

Para gerir, estudar e avaliar os dados, utilizaremos uma segunda máquina virtual também através do serviço de EC2 da AWS. Nesta instância será inicializado o sistema operacional Ubuntu/Bionic, local onde hospedaremos o Grafana. Sua escolha ocorre devido à expectativa de uma baixa demanda. Essa solução se torna mais eficiente em termos de custo, do que o uso do Amazon Managed Service for Grafana. Através da ferramenta, realizaremos o monitoramento em tempo real das temperaturas, para realizar a comunicação entre o banco de dados RDS-MySQL e o Grafana. Será utilizado o plugin disponível no próprio programa, onde será necessária a criação de um novo usuário dentro do banco de dados com a função de leitura das informações nele contidas. Este usuário será o intermediário entre o banco de dados e o Grafana.

As estimativas de custos para o funcionamento das duas instâncias de máquina virtual, o serviço de banco de dados RDS, e o sistema de envio de mensagens SNS, foram estimadas dentro da calculadora da própria provedora dos serviços, baseando-se num modelo de operação 24x7, durante o período de um ano. Os resultados podem ser vistos na Tabela 4.4.

Serviço	Modelo	Disco	Backup	Região	Quantidade	Custo Mensal	Custo Anual
Amazon EC2	t2.micro	10 GB - Un	Não	Norte da Virgínia	2 instâncias	12,51 USD	150,12 USD
Amazon RDS-MySQL	db.t2.micro	30 GB	Sim	Norte da Virgínia	1 instância	18,71 USD	374,64 USD
Amazon SNS	SMS-E-mail	Não aplica	Não aplica	Norte da Virgínia	1 dispositivo móvel - 2 E-mails	0,50 USD	6,00 USD
						31,72 USD	**530,76 USD**

Tabela 4.4 – Tabela de custos estimados para realização da proposta.

5. IMPLEMENTAÇÃO DA ARQUITETURA DE MONITORAMENTO

Neste capítulo apresentaremos a implementação da arquitetura de monitoramento, abordaremos o uso de ferramentas de automação de infraestrutura e configuração de servidores e aplicativos na plataforma de nuvem da Amazon Web Services (AWS). Exploraremos como essas ferramentas podem oferecer uma abordagem eficiente para gerenciar a arquitetura em nuvem, agilizando o processo de configuração e atualização dos recursos. Veremos como a automação pode trazer benefícios como agilidade, eficiência, padronização e escalabilidade, além de possibilitar a prática de infraestrutura como código (IaC). Utilizaremos os exemplos do Terraform e do Ansible para ilustrar como essas ferramentas podem ser aplicadas para criar uma infraestrutura confiável e segura na AWS.

Para o desenvolvimento do projeto na AWS, utilizamos ferramentas de automação de infraestrutura e configuração de servidores e aplicativos. Essas ferramentas permitem gerenciar a arquitetura em nuvem de maneira eficiente, facilitando a implementação de mudanças com poucas linhas de código.

Ao adotar essas ferramentas, ganhamos agilidade e eficiência na configuração e atualização dos recursos em nuvem. Por meio da automação, conseguimos provisionar e configurar servidores, redes, bancos de dados e outros componentes com facilidade e precisão.

Além disso, as ferramentas de automação permitem a aplicação de práticas de infraestrutura como código (*Infrastructure as Code* - IaC), o que significa que podemos definir a configuração da infraestrutura utilizando arquivos de código aberto. Esses arquivos são versionados, podem ser revisados e compartilhados facilmente, proporcionando um controle mais efetivo sobre as mudanças realizadas na infraestrutura.

Com a automação de infraestrutura e configuração, podemos alcançar uma maior consistência e padronização em nossa arquitetura em nuvem. Além disso, a escalabilidade e a capacidade de resposta aos requisitos de carga de trabalho são aprimoradas, uma vez que podemos ajustar rapidamente os recursos de acordo com as necessidades do projeto.

Em resumo, as ferramentas de automação utilizadas no projeto AWS permitem uma gestão eficiente da infraestrutura em nuvem, agilizando o processo de configuração e garantindo maior consistência e escalabilidade. Ao adotar uma abordagem baseada em código aberto, ganhamos flexibilidade e controle sobre a infraestrutura, facilitando a colaboração e o gerenciamento de mudanças.

Utilizamos o Terraform para criar uma rede virtual privada (VPC) com duas sub-redes, garantindo a redundância da infraestrutura. Além disso, foram provisionadas três instâncias no serviço de máquinas virtuais (EC2) da AWS. Para cada instância, foi criado um grupo de segurança (security group) com restrições de portas, garantindo a segurança e o acesso restrito às máquinas. Também foi provisionada uma instância do banco de dados

RDS, com uma única liberação de porta (3306) para o servidor do Mosquitto no grupo de segurança, utilizada pelo banco de dados MySQL.

Após a execução do Terraform, o próximo passo é executar os playbooks do Ansible. Temos três playbooks distintos para cumprir diferentes tarefas. O primeiro playbook instala e configura o servidor Prometheus com Grafana. O segundo playbook instala o Node Exporter para monitorar as instâncias do através Prometheus. Por fim, temos um playbook para instalação e configuração do servidor Mosquitto, além de outro para instalar e configurar a instância que executa o sensor, criado por meio de código em Python.

Essa abordagem com o uso do Terraform e Ansible ofereceu uma maneira automatizada e escalável de provisionar e configurar a infraestrutura na AWS. O Terraform nos ajuda a criar e gerenciar recursos de infraestrutura, enquanto o Ansible facilita a automação da configuração e implantação de aplicativos e serviços. Dessa forma, obtemos uma infraestrutura confiável, segura e pronta para executar as aplicações necessárias no ambiente em nuvem da AWS, um esboço da arquitetura pode ser visto na Figura 5.1.

5.1 Infraestrutura como código com Terraform

Ao usar o Terraform, é possível criar, modificar e destruir recursos de infraestrutura de forma programática, o que ajuda a garantir a consistência e a previsibilidade do ambiente. Com o Terraform, também é possível controlar as dependências entre recursos, criar módulos reutilizáveis e colaborar facilmente em equipe[2].

Em resumo, o Terraform é uma ferramenta essencial para gerenciar infraestrutura em nuvem, permitindo a automação completa da criação e gerenciamento de recursos em nuvens públicas e privadas, SaaS e outras infraestruturas de TI, uma parte do código utilizado pode ser visto no trecho Código 5.1.

Nas primeiras 11 linhas do código, é gerada a chave de acesso para as máquinas criadas na AWS. Para facilitar a gestão e utilização de informações relevantes, é comum a criação de variáveis, como podemos observar nas linhas 4, 9 e 14. Na linha 4 e 9, é definido o nome da chave que será utilizada, denominada "tcc.key". Já na linha 14, são buscadas duas referências: o código do sistema operacional que será criado e a região em que será lançado, no caso, Ubuntu 20.04 na região us-east-1. Essa organização de variáveis permite uma maior agilidade e facilidade na administração das máquinas criadas na AWS.

Na seção do código entre as linhas 13 e 23, são definidos os requisitos da máquina, como o hardware a ser utilizado, o sistema operacional, sua versão e região, bem como a associação de subnet e grupo de segurança. Além disso, é possível adicionar tags que permitem nomear as instâncias de acordo com as necessidades do usuário. Essas configurações são fundamentais para garantir que a máquina criada atenda aos requisi-

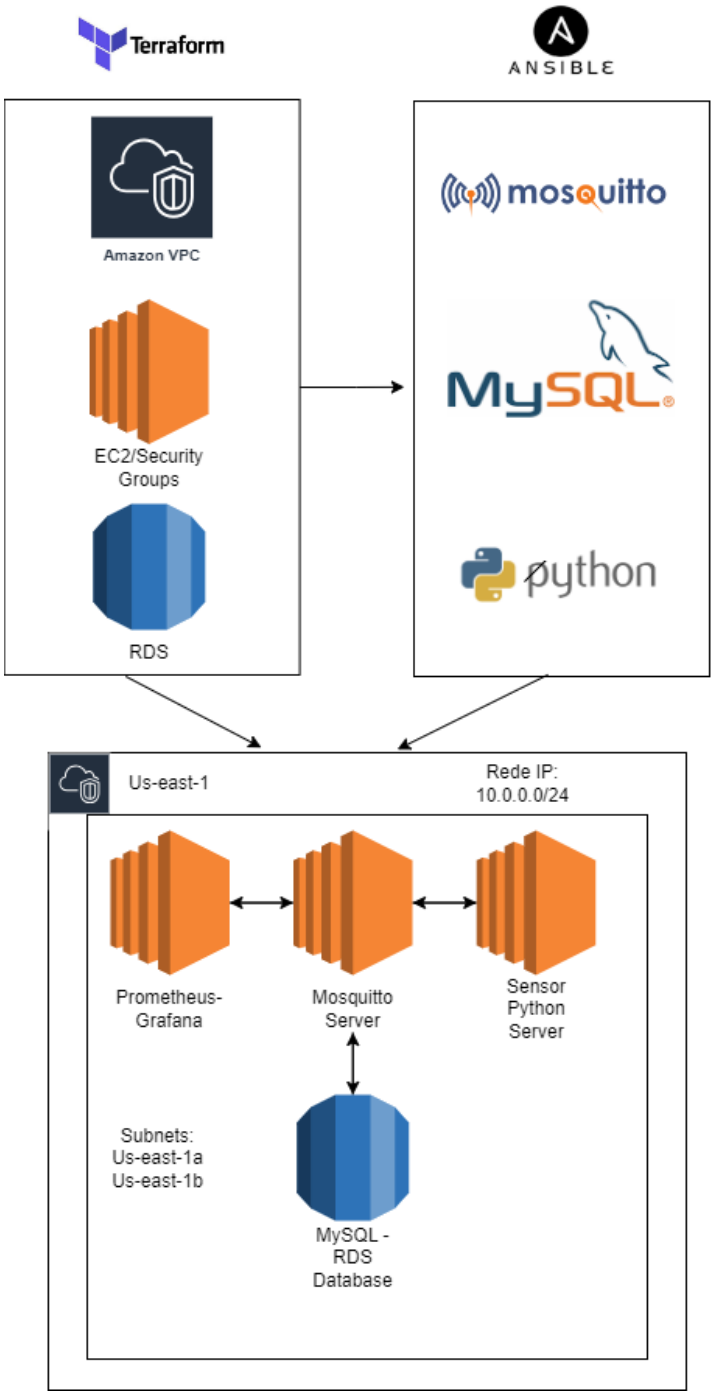

Figura 5.1 – Esboço de funcionamento Terraform e Ansible, juntamente do seu resultado. (Fonte: Autor)

tos específicos de cada projeto ou aplicação. A definição de um hardware adequado e uma região estratégica podem impactar diretamente no desempenho e na disponibilidade do sistema. Já a associação de subnet e grupo de segurança proporciona uma camada adicional de segurança, limitando o acesso de terceiros à instância. Por fim, as tags permitem que o usuário identifique facilmente a instância criada, auxiliando na sua gestão e organização.

Código 5.1 – Trecho de código com Terraform.

```
resource "local_sensitive_file" "private_key" {
  content          = tls_private_key.key.private_key_pem
  filename         = var.KEY_NAME
  file_permission = "0400"
}

resource "aws_key_pair" "key_pair" {
  key_name   = var.KEY_NAME
  public_key = tls_private_key.key.public_key_openssh
}

resource "aws_instance" "server-mosquitto-project" {
  ami                    = var.AMIS[var.REGION]
  instance_type          = "t2.micro"
  subnet_id              = aws_subnet.tcc-pub-1.id
  key_name               = aws_key_pair.key_pair.key_name
  vpc_security_group_ids = [aws_security_group.mosquitto-project_sg.id]

  tags = {
    Name = "server-mosquitto-project"
  }
}
```

5.2 Automação e configuração de servidores com Ansible

Na etapa de instalação dos programas, foi utilizada a ferramenta Ansible, uma plataforma de código aberto que possibilita a gestão, automação e configuração de servidores e aplicativos a partir de uma máquina central, utilizando uma linguagem própria.

Com o Ansible, foi possível configurar facilmente o cliente MQTT, o MySQL, o Python e outros aplicativos fundamentais para o pleno funcionamento do projeto. Essa ferramenta se mostrou altamente eficiente na implantação de softwares, permitindo a execução de tarefas complexas de maneira simples e rápida.

Dessa forma, o uso do Ansible garantiu uma instalação rápida e confiável dos aplicativos necessários, além de possibilitar uma fácil manutenção e atualização dos mesmos. Em resumo, o Ansible é uma ferramenta poderosa e versátil que se mostrou fundamental para a realização dessa etapa do projeto. Parte do código utilizado está apresentado no Código 5.2.

Nas primeiras linhas, 2, 3 e 4, definimos a tarefa principal que o Ansible executará. Na linha de "hosts", especificamos o grupo de máquinas que serão modificadas pelo playbook, enquanto a linha 4, executa as tarefas como usuário administrador do sistema Linux.

Código 5.2 – Trecho de código com Ansible.

```
1  ---
2  - name: Install Mosquitto and Mosquitto client
3    hosts: server-mosquitto-project
4    become: true
5    tasks:
6      - name: Update package index
7        apt:
8            update_cache: yes
9            cache_valid_time: 3600
10
11     - name: Install Mosquitto package
12       apt:
13         name: mosquitto
14         state: present
15
16     - name: Install Mosquitto client package
17       apt:
18         name: mosquitto-clients
19         state: present
20
21     - name: Start and enable Mosquitto service
22       systemd:
23         name: mosquitto
24         state: started
25         enabled: true
```

A partir da linha 5 até a 25, são apresentadas as tarefas executadas pelo sistema, incluindo instalações, cópias e pequenas execuções. Na linha 6, é realizada uma atualização do kernel do sistema operacional. Já nas linhas 11 até 19, são instalados os componentes necessários do Mosquitto, um cliente MQTT utilizado no projeto.

Por fim, na linha 21, o serviço é ativado e iniciado após a instalação. Essa sequência de tarefas é fundamental para garantir o pleno funcionamento do projeto, além de demonstrar a eficiência do Ansible na automação de processos complexos.

5.3 Arquitetura e suas especificações na AWS

Foram criadas duas instâncias t2.micro com especificações de uma CPU e 1 GB de memória RAM cada, todas executando o sistema operacional Ubuntu 20.04. Em uma das máquinas, o servidor Mosquitto está em execução para receber dados de temperatura, enquanto em outra máquina, temos o Grafana em execução, permitindo a visualização dos dados em tempo real a partir do banco de dados onde as temperaturas são armazenadas.

O banco de dados está armazenado em uma instância RDS t2.micro, com 20 GB de armazenamento, uma CPU e 1 GB de memória RAM. Ao contrário das instâncias EC2, essas instâncias são direcionadas especificamente para bancos de dados, o que significa que não é possível escolher o sistema operacional ou instalar outro software. No grupo de segurança, é comum ter apenas a porta 3306 (padrão do banco de dados MYSQL) liberada para configurações de tabelas e execuções de consultas.

Para realizar a simulação de um sensor de temperatura, foi criado um script em Python que utiliza uma biblioteca chamada "paho-mqtt" para enviar as temperaturas para o Mosquitto. O Mosquitto é um servidor de mensageria que segue o protocolo MQTT (*Message Queue Telemetry Transport*) e é amplamente utilizado em sistemas de IoT para a comunicação entre dispositivos e serviços.

Neste projeto, utilizamos o Mosquitto para receber as temperaturas simuladas pelo script Python e encaminhá-las para a instância de banco de dados, que armazena as informações em uma tabela específica.

Para garantir a eficiência do processo, a conexão entre o script Python e o Mosquitto é estabelecida utilizando o protocolo MQTT, que é otimizado para a comunicação em redes de baixa largura de banda e alta latência. Além disso, a conexão com a instância de banco de dados é realizada por meio do protocolo MySQL, que é amplamente utilizado em sistemas de gerenciamento de bancos de dados relacionais.

O script Python simula a leitura do sensor de temperatura a cada 5 segundos, gerando diversas temperaturas aleatórias enviadas para o Mosquitto a cada segundo. Cada mensagem enviada pelo script contém o valor da temperatura, o horário em que a leitura foi realizada e um identificador único utilizado para indexar os dados na tabela do banco de dados.

A instância de banco de dados é responsável por armazenar as informações enviadas pelo script Python, permitindo que as temperaturas sejam visualizadas e analisadas posteriormente por meio de uma ferramenta de visualização, como o Grafana, por exemplo.

Em resumo, o funcionamento do projeto envolve a simulação de um sensor de temperatura por meio de um script Python, o envio das temperaturas para o Mosquitto e o armazenamento dos dados em uma instância de banco de dados, permitindo a visualização e análise das informações em tempo real.

A visualização dos dados no Grafana é feita através da configuração do data source, onde deve-se inserir as informações do banco de dados, tais como endereço IP, porta, nome de usuário e senha. Depois que a conexão for estabelecida, é possível criar um painel no Grafana e especificar a consulta SQL desejada para buscar os dados na tabela do banco de dados e exibi-los através de um painel configurado conforme as necessidades do usuário. Um exemplo de sua utilização está disponível na Figura 5.2.

Panel Title

codigo	temperatura	dispositivo	descricao	date_timestamp
1	40.2	dispositivo_1	sensor_1	2023-05-21 10:10:02
2	27.8	dispositivo_1	sensor_1	2023-05-21 10:11:02
3	28.5	dispositivo_1	sensor_1	2023-05-21 10:12:03
4	36.5	dispositivo_1	sensor_1	2023-05-21 10:14:54
5	21.9	dispositivo_2	sensor_2	2023-05-21 10:14:54
6	24.8	dispositivo_3	sensor_3	2023-05-21 10:14:54
7	43.1	dispositivo_4	sensor_4	2023-05-21 10:14:54
8	41.9	dispositivo_5	sensor_5	2023-05-21 10:14:54
9	45.5	dispositivo_6	sensor_6	2023-05-21 10:14:54
10	41.1	dispositivo_7	sensor_7	2023-05-21 10:14:54
11	28.0	dispositivo_8	sensor_8	2023-05-21 10:14:54
12	32.9	dispositivo_9	sensor_9	2023-05-21 10:14:54
13	20.3	dispositivo_10	sensor_10	2023-05-21 10:14:54
14	44.1	dispositivo_1	sensor_1	2023-05-21 10:15:54
15	18.6	dispositivo_2	sensor_2	2023-05-21 10:15:54
16	24.2	dispositivo_3	sensor_3	2023-05-21 10:15:54
17	29.4	dispositivo_4	sensor_4	2023-05-21 10:15:54
18	43.4	dispositivo_5	sensor_5	2023-05-21 10:15:54
19	30.7	dispositivo_6	sensor_6	2023-05-21 10:15:54
20	25.6	dispositivo_7	sensor_7	2023-05-21 10:15:54
21	37.3	dispositivo_8	sensor_8	2023-05-21 10:15:54

Figura 5.2 – Grafana para visualização da tabela no banco de dados MySQL (Fonte: Autor)

Este capítulo apresentou a implementação da arquitetura de monitoramento de temperatura usando a AWS. Foi detalhado o processo de instalação e configuração dos softwares necessários para o funcionamento adequado da arquitetura. No próximo capítulo, será realizada uma avaliação do desempenho e funcionamento da arquitetura implementada. Serão discutidos os resultados obtidos, destacando os pontos fortes e testes de carga realizados, proporcionando uma visão mais completa sobre a eficácia e viabilidade dessa solução de monitoramento de temperatura.

6. AVALIAÇÃO DA ARQUITETURA DE MONITORAMENTO DE TEMPERATURA

Para validar o desenvolvimento prático realizado, realizamos um monitoramento detalhado do desempenho do servidor, incluindo a CPU, a memória RAM e outros componentes críticos para garantir a estabilidade e eficiência de um servidor de produção. Para isso, realizamos testes de carga no servidor Mosquitto para avaliar a sua capacidade de lidar com múltiplas requisições simultâneas.

Durante os testes de carga, a máquina se mostrou extremamente estável e capaz de lidar com as diversas requisições sem qualquer problema. Isso garantiu que as informações de temperatura fossem processadas adequadamente e transmitidas para o Grafana, bem como armazenadas no banco de dados hospedado no serviço Amazon RDS.

Em resumo, o monitoramento constante e os testes de carga realizados comprovaram a eficiência e estabilidade do servidor Mosquitto e asseguraram que o mesmo está apto para o uso em um ambiente de produção exigente. Utilizamos dez, vinte e cinco, cinquenta, setenta e cinco e por fim, cem sensores enviando requisições a cada 1 segundo, durante dez minutos. Cada execução foi repetida cinco vezes, obtendo 100% de recebimento e entrega das informações contidas.

Para demonstração dos testes, foi desenvolvido um gráfico apresentado na Figura 6.1. Os valores da vertical indicam os valores de utilização de CPU do servidor, enquanto os valores da horizontal, tratam-se do número de sensores que estavam executando no momento que a utilização média de CPU foi alcançada. No geral, o valor médio de utilização foi de 0,7%.

Para garantir a estabilidade da instância de banco de dados RDS, foi adicionado o serviço da Amazon SNS para enviar alertas por e-mail caso ocorra alguma indisponibilidade do banco de dados. Para que o administrador receba os alertas corretamente, é necessário aceitar o e-mail de inscrição enviado pelo serviço AWS. Isso ajuda a evitar o cadastramento de e-mails inválidos ou spam.

O trecho apresentado no Código 6.1 é responsável pela definição de um tópico SNS (*Simple Notification Service*) da AWS, que é um serviço de mensagem que permite a entrega de mensagens para múltiplos destinatários. Nas linhas 1, 2 e 3, é definido o nome do tópico, que pode ser escolhido pelo usuário e neste caso é denominado "rds-status-topic".

Entre as linhas 5 e 8, é tratada a inscrição do e-mail que irá receber os alertas em caso de inconsistências no banco de dados RDS. A função de output na linha 60 é utilizada por outros recursos do Terraform para permitir o acesso ao endereço de e-mail inscrito posteriormente.

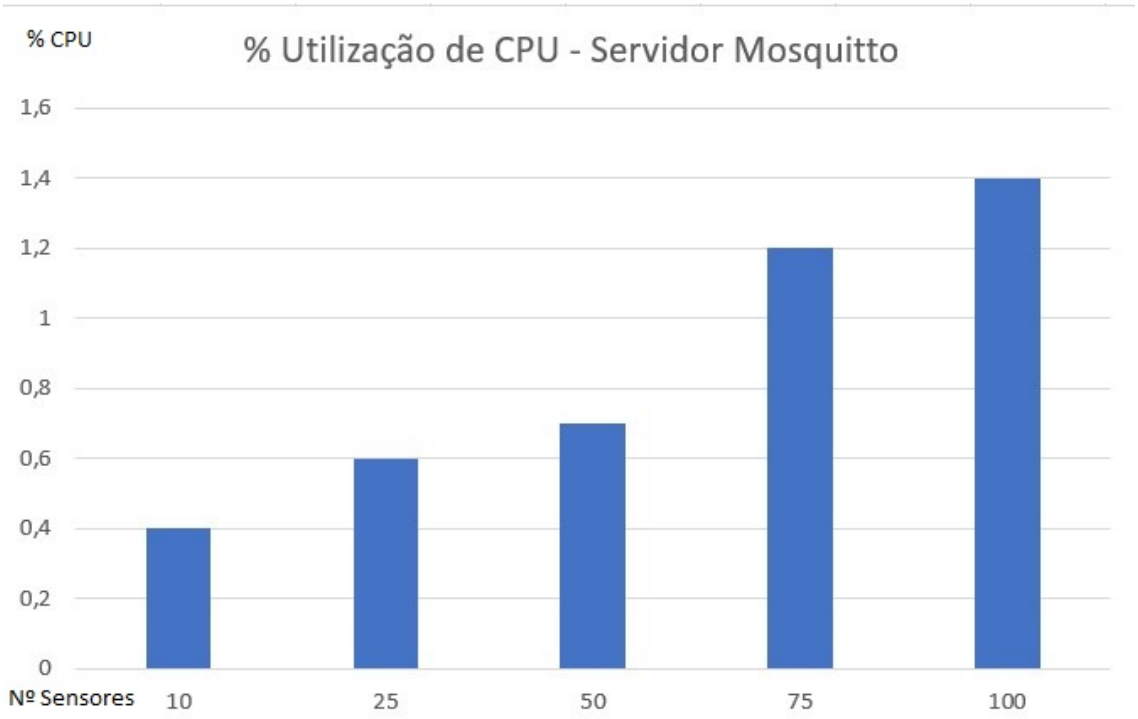

Figura 6.1 – Gráfico de utilização de CPU - Servidor Mosquitto (Fonte: Autor)

Por fim, nas linhas 15 até o final do código, é definido o alarme de métricas do CloudWatch para monitorar o status do banco de dados RDS. O CloudWatch é um serviço de monitoramento e logging da AWS, que permite monitorar e coletar dados de diversos recursos da AWS para fins de análise e diagnóstico.

O CloudWatch pode ser utilizado para monitorar diversas outras métricas, desde status de máquinas até logs de banco de dados, sendo uma ferramenta muito utilizada por diversas empresas na atualidade, pois trata-se de um serviço de simples configuração e com um custo considerado baixo para todos os recursos que disponibiliza.

Após concluir todas as implementações e testes necessários para o funcionamento do projeto, foi realizada uma nova estimativa de custo mensal para manter a arquitetura hospedada na Amazon. Os valores estimados desta nova análise foram semelhantes aos previamente calculados, como demonstrado na Tabela 6.1.

Serviço	Modelo	Disco	Backup	Região	Quantidade	Custo Mensal	Custo Anual
Amazon EC2	t2.micro	10 GB - Un	Não	Norte da Virgínia	2 instâncias	14,36 USD	172,32 USD
Amazon RDS-MySQL	db.t2.micro	30 GB	Sim	Norte da Virgínia	1 instância	22,21 USD	404,21 USD
Amazon SNS	SMS-E-mail	Não aplica	Não aplica	Norte da Virgínia	2 E-mails	0,50 USD	6,00 USD
						31,72 USD	**582,53 USD**

Tabela 6.1 – Tabela de custos mensal/anual final para realização do projeto.

Código 6.1 – Configuração do SNS.

```
resource "aws_sns_topic" "rds-status-topic" {
  name = "rds-status-topic"
}

resource "aws_sns_topic_subscription" "email-subscription" {
  topic_arn = aws_sns_topic.rds-status-topic.arn
  protocol  = "email"
  endpoint  = "seuemail@domio.com"
}

output "rds-status-topic-arn" {
  value = aws_sns_topic.rds-status-topic.arn
}

resource "aws_cloudwatch_metric_alarm" "db-temperatura-alarm" {
  alarm_name          = "db-temperatura-alarm"
  comparison_operator = "GreaterThanOrEqualToThreshold"
  evaluation_periods  = "1"
  metric_name         = "DBInstanceStatus"
  namespace           = "AWS/RDS"
  period              = "60"
  statistic           = "Maximum"
  threshold           = "0"
  alarm_description   = "Alarm when the RDS instance is not available"
  alarm_actions       = [aws_sns_topic.rds-status-topic.arn]

  dimensions = {
    DBInstanceIdentifier = aws_db_instance.db-temperatura.identifier
  }
}
```

7. CONCLUSÃO

O controle de temperatura em Data Centers é uma parte crucial para manter a integridade da informação e dos componentes que compõem a sua infraestrutura. Com o avanço acelerado da Internet das Coisas, surgiram dispositivos inteligentes com baixo consumo energético. Estes dispositivos são capazes de controlar o ambiente de maneira automatizada, atendendo os mais diversos tipos de cenários e modelos de operação. Seus componentes internos foram desenvolvidos para avaliar as métricas do ambiente físico e enviá-las para um microcontrolador que trata os dados recebidos, transferindo-os para aplicativos auxiliares que armazenam e mostram estes índices de maneira mais legível para seus administradores.

A Computação em Nuvem vem sendo adotada por empresas dos mais diversos tamanhos e segmentos por conta da sua facilidade de utilização. A sua vasta gama de serviços escaláveis e pré-configurados, tornam a sua operação mais simples, podendo em casos específicos, dispensar a necessidade de um profissional da área de tecnologia para realizar a sua execução. Os seus custos costumam ser mais em conta do que adquirir uma infraestrutura própria, adotando um modelo *pay-per-use*, terceirizando a manutenção e atualização dos equipamentos do Data Center.

Através do protótipo desenvolvido, foi possível adquirir mais conhecimento referente ao modo de operação de um sensor de temperatura e as ferramentas auxiliares que juntas formam uma poderosa solução para controle de equipamentos e monitoramento (Grafana) e de banco de dados (MySQL). Para a continuidade do projeto, deve-se buscar as melhores práticas de realizar esta arquitetura utilizando a Computação em Nuvem, através dos serviços oferecidos pela Amazon Web Services. Os estudos realizados sobre a plataforma nos mostram diversas maneiras de transmitir os seus dados para o ambiente *cloud*, levando em consideração os custos para realizar o monitoramento do ambiente de forma segurança e com um custo relativamente baixo, considerando a necessidade de operação e monitoramento de forma ininterrupta.

No desenvolvimento prático, foi utilizado um conjunto de ferramentas e serviços da AWS para criar uma arquitetura em nuvem eficiente e escalável. O Terraform foi utilizado para automatizar a criação e gerenciamento da infraestrutura, proporcionando facilidade e consistência no provisionamento dos recursos. Com o uso do Ansible, foi possível automatizar a configuração e instalação dos aplicativos necessários, simplificando o processo de implantação das ferramentas utilizadas no projeto.

A arquitetura criada consiste em instâncias EC2 executando o Mosquitto, um servidor MQTT, e o script Python simulando um sensor de temperatura. As temperaturas são enviadas para o Mosquitto e armazenadas em uma instância RDS. A visualização dos da-

dos é feita por meio do Grafana, que se conecta ao banco de dados e exibe as informações em tempo real.

Os testes de carga realizados comprovaram a estabilidade e eficiência do servidor Mosquitto, demonstrando que ele é capaz de lidar com múltiplas requisições simultâneas sem problemas. O monitoramento constante dos componentes críticos do servidor também garantiu sua estabilidade e desempenho adequados.

Para garantir a estabilidade da instância RDS, foi implementado o serviço de alertas do Amazon SNS, que envia notificações por e-mail em caso de indisponibilidade do banco de dados. Isso permite que o administrador seja informado imediatamente sobre qualquer problema e possa tomar as medidas necessárias para resolver a situação.

Além disso, o uso do CloudWatch possibilitou o monitoramento das métricas do banco de dados RDS, fornecendo dados valiosos para análise e diagnóstico. O CloudWatch é uma ferramenta poderosa que contribui para o bom funcionamento e manutenção da arquitetura.

Por fim, a estimativa de custo mensal para manter a arquitetura na AWS foi realizada, e os valores estimados foram compatíveis com as expectativas iniciais, demonstrando que a arquitetura é viável e possui um custo adequado.

Em conclusão, o desenvolvimento deste trabalho foi bem-sucedido na criação de uma arquitetura em nuvem utilizando ferramentas e serviços da AWS. A automação de infraestrutura e configuração de servidores proporcionou agilidade, consistência e facilidade na implantação do projeto. Os testes de carga e monitoramento constante garantiram a estabilidade e eficiência da arquitetura. Além disso, a implementação de alertas e o monitoramento com o Grafana/CloudWatch contribuíram para a detecção e resolução rápida de problemas. No geral, o projeto demonstrou a aplicação prática e efetiva de tecnologias em nuvem para a criação de uma solução de IoT escalável e confiável.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] AMAZON. “Amazon web services, inc.” Último acesso em 11 de setembro de 2022, Capturado em: https://aws.amazon.com/pt/, 2022.

[2] and-tino teuber, T. “Visão geral do terraform no azure – o que é o terraform?” Último acesso em 16 de abril de 2023, Capturado em: https://learn.microsoft.com/pt-br/azure/developer/terraform/overview, 2023.

[3] Brunetti, R. “Windows Azure Step by Step”. Estados Unidos: On ne Tra n ng So utons, Inc, 2011.

[4] David, M. P.; Schmidt, R. R. “Impact of ashrae environmental classes on data centers,”, *Fourteenth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, maio 2014, pp. 1092–1099.

[5] de Oliveira Soares, W.; dos Santos Faria, N. C.; Marcos Antônio de Sousa, B. Q. d. O. e. A. M. d. M. M. “MONITORAMENTO E CONTROLE DA CLIMATIZAÇÃO DE DATA CENTER VIA IOT”, 2021.

[6] Edelhoff, R. “A serverless iot backend with aws iot”. Último acesso em 27 de outubro de 2022, Capturado em: https://blogs.itemis.com/en/a-serverless-iot-backend-with-aws-iot, 2022.

[7] Escola, J. P. L.; Freitas, J. d. S. “Sistema de monitoramento de baixo custo para datacenter baseado em internet das coisas”, *Revista Interface Tecnológica*, vol. 17–1, ago 2020, pp. 65–81.

[8] GARCIA, D. “Centro de dados (datacenters)”, *Intr@ ciência*, 2012, pp. 15.

[9] Grance, P. M. . T. “The NIST Definition of Cloud Computing”, *NIST Technical Series Publications*, 2011.

[10] Junior, A. C.; Miers, C. C.; Koslovski, G. P.; Pillon, M. A. “Análise de Zonas Térmicas em Data Center Não-CRAC. [Em anais do XVIII simpósio em sistemas computacionais de alto desempenho]”, *SBC - Sociedade Brasileira de Computação*, 2017, pp. 124–135.

[11] Kang, D.-H.; Park, M.-S.; Kim, H.-S.; Kim, D.-y.; Kim, S.-H.; Son, H.-J.; Lee, S.-G. “Room Temperature Control and Fire Alarm/Suppression IoT Service Using MQTT on AWS”, 2017.

[12] Lerch Lunardi, G.; Simões, R.; Saraiva Frio, R. “Ti verde: uma análise dos principais benefícios e práticas utilizadas pelas organizações.”, *Revista Eletrônica de Administração*, vol. 20–1, abr 2014, pp. 1–30.

[13] Locatelli, C. “Monitoramento e controle por aplicativo - mqtt”. Último acesso em 27 de outubro de 2022, Capturado em: https://curtocircuito.com.br/blog/Categoria%20IoT/monitoramento-e-controle-por-aplicativo-mqtt, 2020.

[14] Microsoft. “Tecnologias e protocolos de iot”. Último acesso em 12 de setembro de 2022, Capturado em: https://azure.microsoft.com/pt-br/solutions/iot/iot-technology-protocols/, 2022.

[15] Milani, A. “MySQL - Guia do Programador”. São Paulo - BR: Novatec Editora Ltda, 2006.

[16] Oliveira, A.; Neves, J.; Rezende, T.; Teixeira, P. “Aplicações de automação em IOT - internet of things”, *Revista Científica e-Locução*, vol. 1–10, dez 2016, pp. 19.

[17] Romancini, E. M. R.; Zanon, V. R.; Morales, A. S.; de Oliveira Ourique, F.; Moraes, R. “Monitoramento inteligente do consumo de energia elétrica em residências utilizando recursos de iot”, *Anais do Computer on the Beach*, vol. 13, 2022, pp. 134–141.

[18] Rosenberger. “Data center cabling solutions”. Último acesso em 5 de setembro de 2022, Capturado em: https://www.rosenbergerap.com/solutionDetail.html?rootId=76&id=227, 2022.

[19] Santos, B. “Como sensores iot estão revolucionando contact centers?” Último acesso em 11 de setembro de 2022, Capturado em: https://www.khomp.com/pt/como-sensores-iot-estao-revolucionando-contact-centers/, 2022.

[20] Uptime Institute, L. “Certificação tier”. Último acesso em 11 de setembro de 2022, Capturado em: https://pt.uptimeinstitute.com/tier-certification, 2022.

[21] Veras, M. “Virtualização - Componente Central do Datacenter”. Rio de Janeiro, RJ: Brasport Livros e Multimídia Ltda, 2011.

[22] Veras, M. “Computaçao em Nuvem - Nova Arquitetura”. Rio de Janeiro, RJ: Brasport Livros e Multimídia Ltda, 2015.