UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Elen Priscila de Souza Lobato

Implantação de um *middleware* IoT escalável para aplicações de Mobilidade Elétrica Multimodal

DM: 10/2022

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém – Pará – Brasil
2022

Elen Priscila de Souza Lobato

Implantação de uma *middleware* IoT escalável para aplicações de Mobilidade Elétrica Multimodal

Dissertação apresentada no Programa de Pós-Graduação em Engenharia Elétrica do Instituto de Tecnologia como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica na área de Sistemas de Energia Elétrica.

Universidade Federal do Pará

Orientador: Prof. Dr. Denis Lima do Rosário
Coorientador: Prof. Dr. Ubiratan Holanda Bezerra

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém – Pará – Brasil
2022

Elen Priscila de Souza Lobato

Implantação de uma *middleware* IoT escalável para aplicações de Mobilidade Elétrica Multimodal

Dissertação apresentada no Programa de Pós-Graduação em Engenharia Elétrica do Instituto de Tecnologia como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica na área de Sistemas de Energia Elétrica.

Conceito: _____

Belém, 25 de fevereiro de 2022.

BANCA EXAMINADORA

_____

Prof. Dr. Denis Lima do Rosário

Orientador – PPGEE / UFPA

_____

Prof. Dr. Ubiratan Holanda Bezerra

Coorientador – PPGEE / UFPA

_____

Profª. Drª. Maria Emília de Lima Tostes

Avaliadora interna – PPGEE / UFPA

_____

Prof. Dr. Wellington da Silva Fonseca

Avaliador interno – PPGEE / UFPA

_____

Prof. Dr. Fabricio de Souza Farias

Avaliador externo – Campus Universitário do Tocantins / UFPA-CAMETÁ

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

S719i Souza Lobato, Elen Priscila de.
Implantação de um middleware IoT escalável para aplicações de Mobilidade Elétrica Multimodal / Elen Priscila de Souza Lobato. — 2022.
99 f. : il. color.

Orientador(a): Prof. Dr. Denis Lima do Rosário Coorientador(a): Prof. Dr. Ubiratan Holanda Bezerra Dissertação (Mestrado) - Universidade Federal do Pará,
Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2022.

1. Mobilidade Elétrica Multimodal. 2. Dojot. 3. Internet das Coisas. 4. Docker Compose. 5. Kubernetes.
I. Título.

CDD 004.6782

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.
E a todas as vítimas do COVID-19.

Agradecimentos

A Deus, criador do universo, o autor da minha história, que também é o meu Pai (Abba), o meu eterno agradecimento que transcendem espaço-tempo. Tudo o que fui, sou e ainda serei, é para a honra e glória do nome DEle. Agradeço a toda a minha família, em especial a minha mãe, Rosiane Lobato, por suas orações e por tudo apoio desde o meu nascimento. Agradeço também a minha avó Engracia e meu avô Carlos Lobato. E aos meus tios Carlos e Patricia, que mesmo distantes sempre se fizeram presentes.

Agradeço ao meu orientador, Prof. Dr. Denis Rosário, que ainda na graduação, me orientou em um Projeto de Extensão e agora no Mestrado voltou a ser meu orientador, me conduziu no desenvolvimento desta dissertação, tornando este e outros trabalhos possíveis. Agradeço também ao meu coorientador Prof. Dr. Ubiratan Bezerra, que compartilhou comigo sua sabedoria e experiências da sua longa jornada acadêmica.

Agradeço a todos os membros da minha valorosa banca. A Prof. Dra. Maria Emília Tostes, minha mentora, que me serve de inspiração, a quem eu tanto admiro. Ao Prof. Dr. Wellington Fonseca, meu mentor, com quem tenho a honra de trabalhar desde a graduação e tem me orientado na minha jornada acadêmica. Ao Prof. Dr. Fabrício Farias e ao Haelton Carvalho pelas valiosas contribuições no desenvolvimento deste trabalho.

Gratidão a toda a Família CEAMAZON, onde tenho a honra de trabalhar com excelentes como profissionais e que também são pessoas incríveis. A toda a Equipe do projeto SIMA que tornou esse P&D possível, em especial a minha equipe de computação, Rodrigo Muniz e Izidio Carvalho que muito somaram na entrega dos produtos que juntos desenvolvemos.

Agradeço também as seguintes instituições: PPGEE da UPFA; a CAPES, pelo apoio financeiro dado a esta pesquisa; a Norte Energia S.A, financiadora do Projeto SIMA, que originou está dissertação; e ao CPqD, mais especificamente, ao Antônio

Bizetti, Marcio Funes, Leonardo Mariano, Daniela Petito e Anderson Luiz Ribeiro (*in memória*).

Minha gratidão a todas as pessoas que, por mais que eu não tenha citado o nome, sabem que me ajudaram de alguma forma, direta ou indireta, para que eu chegasse até aqui. Todos vocês foram, são e sempre farão parte da minha história.

Soli Deo Glória!

Resumo

Este trabalho irá apresentar a implementação da plataforma Dojot de forma escalável e customizada, como o *middleware* de Internet das Coisas (IoT, do inglês *Internet of Things*) de um Projeto Pesquisa e Desenvolvimento (P&D) de Mobilidade Elétrica Multimodal na Região Amazônica. Foram implementados dois ambientes com a Dojot, sendo um ambiente de desenvolvimento com *Docker Compose*, em apenas uma máquina, e um ambiente de produção em um cluster *Kubernetes,* com uma máquina *master*, duas máquinas *workers* e um balanceador de carga. Como forma de verificar o consumo das métricas de CPU e memória, nos dois ambientes, foram realizadas simulações de vários dispositivos de mobilidade elétrica recebendo dados simultaneamente na Dojot. Dessa forma, observou-se que houve um menor consumo de CPU no ambiente de desenvolvimento, menor consumo de memória no ambiente de produção e o ambiente que melhor respondeu à questão da escalabilidade (com o aumento da quantidade de dispositivos recebendo dados) foi o de produção. Concluindo-se assim, que a implantação da Dojot no ambiente de produção, em um *cluster Kubernetes*, é mais vantajosa em relação a escalabilidade.

Palavras-chave: Mobilidade Elétrica Multimodal, Dojot, Internet das Coisas, *Docker Compose*, *Kubernetes*.

Abstract

This work will present the implementation of the Dojot platform in a scalable and customized way, such as the Internet of Things (IoT) middleware of a Research and Development (R&D) Project of Multimodal Electric Mobility in the Amazon Region. Two environments were implemented with Dojot, being a development environment with Docker Compose, in only one machine, and a production environment in a Kubernetes cluster, with a master machine, two worker machines and a load balancer. To verify the consumption of CPU and memory metrics, in both environments, simulations were performed of several electric mobility devices receiving data simultaneously in Dojot. Thus, it was observed that there was a lower CPU consumption in the development environment, lower memory consumption in the production environment and the environment that best responded to the scalability issue (with the increase in the number of devices receiving data) was production. In conclusion, the implementation of Dojot in the production environment, in a Kubernetes cluster, is more advantageous in terms of scalability.

Keywords: Multimodal Electric Mobility, Dojot, Internet of Things, Docker Compose, Kubernetes.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de abreviaturas e siglas

ANEEL	Agência Nacional de Energia Elétrica
API	*Application Programming Interface*
CEAMAZON	Centro de Excelência em Eficiência Energética da Amazônia
CNCF	*Cloud Native Computing Foundation*
CPLF	Companhia de Distribuição de Utilidades Local
CPqD	Centro de Pesquisa e Desenvolvimento em Telecomunicações
CTIC	Centro de Tecnologia da Informação e Comunicação
FTP	*File Transfer Protocol*
GIS	*Geographic Information System*
GUI	Interface Gráfica do Usuário
HCI	*Hyper Converged Infrastructure*
HTTP	*Hypertext Transfer Protocol*
IED	*Intelligent Electronic Devices*
IP	*Internet Protocol*
KVM	*Kernel-based Virtual Machine*
LXC	*Linux Containers*
ONU	Organização Internacional das Nações Unidas
P&D	Projeto de Pesquisa e Desenvolvimento
PPGEE	Programa de Pós-Graduação em Engenharia Elétrica
REST	*Representational State Transfer*
RISE	Rede de Inovação do Setor Elétrico
RSSI	*Received Signal Strength Indication*

SG2IoT	*Legacy Smart Grid to IoT Protocol Integration Approach*
SIMA	Sistema Inteligente da Amazônia
SNR	*Signal-to-Noise Ratio*
TIC	Tecnologias da Informação e Comunicação
UFPA	Universidade Federal do Pará
UNICAMP	Universidade de Campinas
UNICAMP	Universidade de Campinas
VE	*Virtual Enviroment*
VM	*Virtual Machines*

Sumário

1 INTRODUÇÃO

Neste Capítulo introdutório, será descrita uma abordagem geral sobre os principais temas desta dissertação. Em seguida, será feita uma contextualização sobre a pesquisa desenvolvida, e serão apresentadas, respectivamente, as justificativas e motivações que levaram ao desenvolvimento desta dissertação. Posteriormente, serão apresentados o objetivo geral e os objetivos específicos. Por fim, será descrita a estrutura deste trabalho.

1.1 Visão geral

Após o fenômeno da urbanização, a mobilidade urbana e a emissão de gases poluentes pelo setor de transportes passaram a serem alguns dos principais problemas enfrentados pela humanidade. Não é por menos que o objetivo 10.7 da Agenda 30 da Organização Internacional das Nações Unidas (ONU) é facilitar a mobilidade de forma ordenada, segura, regular e responsável, para ajudar a reduzir a desigualdade (ONU, 2015a). Além disso, em 2015 foi assinado um acordo mundial, o Acordo de Paris, em que os países signatários se comprometem em manter o aumento da temperatura do planeta abaixo de 2ºC, mais precisamente 1,5ºC, até 2100 (ONU, 2015b). Demonstrando assim a preocupação mundial em torno desses temas.

Em relação a mobilidade urbana, este conceito ganhou força no Brasil a partir da Lei 12.587/12 de Política Nacional de Mobilidade Urbana de 2012, que tem como objetivo integrar os diferentes modos de transporte, por exemplo, ônibus, barco, bicicleta, e melhorar a acessibilidade e mobilidade das pessoas e cargas nas cidades (BRASIL, 2012). Hoje, a mobilidade urbana, incorpora também o conceito de sustentabilidade, através do incentivo do uso de transporte coletivos, não motorizados,

ou elétricos, que são ecologicamente mais sustentáveis (DUARTE, LIBARDI, SANCHEZ, 2007). Inserindo-se, assim, uma forte tendência na mobilidade urbana, a partir da mobilidade elétrica, que por sua vez é um dos principais temas pilares desta dissertação.

De acordo com o 1º Anuário Brasileiro da Mobilidade Elétrica (PNME, 2021) a mobilidade elétrica corresponde ao processo de eletrificação do setor de transporte. Quando a mobilidade elétrica faz uso de Tecnologias da Informação e Comunicação (TIC), ela consegue entregar serviços e aplicações que tornam a mobilidade urbana em uma mobilidade inteligente (MAHREZ *et al*., 2021). À exemplo desses serviços e aplicações, pode-se citar os postos de recarga, os aplicativos de localização, monitoramento e gerenciamento de eletropostos, sistemas de armazenamento de energia em baterias, sistemas fotovoltaicos, entre outros (VIJAYAKUMAR, 2021).

Usualmente o conceito de mobilidade inteligente está presente nos *Campus* Inteligentes (do inglês, *Smart Campus*) (AZEVEDO *et al*., 2020) e nas Cidades Inteligentes (do inglês, *Smart Cities*) (DIRAN *et al*., 2021). Na mobilidade inteligente os veículos e usuários estão conectados a uma infraestrutura inteligente, a nuvem e a outros usuários (DONNELLAN, 2018). Os *Campus* inteligentes, por sua vez, adotam TIC para monitorar e controlar de forma automática as instalações de um *Campus* (AZEVEDO *et al*., 2020) e tem como objetivo melhorar a qualidade de vida da sua população (IMBAR; SUPANGKAT; LANGI, 2020).

Já as Cidades Inteligentes, de acordo com a ABNT NBR ISO 37122:2020, usam as informações de dados e tecnologias modernas, para fornecer melhores serviços e qualidade de vida para os habitantes das cidades (ABNT, 2020). Pois, as Cidades Inteligentes devem ser Cidades Humanas, Inteligentes, Criativas e Sustentáveis, gerando-se o termo CHICS. Construindo-se, assim, uma cidade boa para viver, estudar, trabalhar, investir e visitar, de forma sustentável, criativa e com alta qualidade de vida (IBCIHS, 2018 apud IBCIHS, 2020) .

Uma das principais aliadas dos *Campus* Inteligentes, da mobilidade inteligente e das Cidades Inteligentes, é a Internet das Coisas (IoT, do inglês *Internet of Things*) (PINTO, 2017; MOTA; RIKER; ROSÁRIO, 2019). De acordo com Nascimento (2020), a

IoT é a combinação de diferentes tecnologias de TIC como dispositivos e sensores, conectividade e mobilidade, computação em nuvem, geolocalização e redes sociais.

Quando aplicada à mobilidade elétrica, a IoT possui a capacidade de criar aplicações que: coletam/fornecem dados em tempo real (BUSATO; ALMEIDA, 2019), realizam o processamento de dados (AZEVEDO *et al*., 2020), e realizam o monitoramento e a gestão da infraestrutura física dos meios de transporte, entre outras (CIVALI *et al*., 2019; AZEVEDO *et al*., 2020). Assim, com tantas aplicações possíveis e a enorme quantidade de objetos/dispositivos conectados à internet, aumenta-se exponencialmente a quantidade de dados gerados, nesse cenário de IoT (KHARE; TOTARO, 2019; CISCO, 2022).

De acordo com a literatura (IEEE, 2020; KHARE; TOTARO, 2019) os dados coletados por IoT, são processados, armazenados e por fim podem ser usados pelas aplicações. Mediante a isso, *middlewares IoT* são de suma importância para coletar, processar, monitorar e gerenciar esses dados (AZEVEDO *et al*., 2020). Além de também, garantir disponibilidade dos dados para as aplicações através de estratégias de escalabilidade (MITRPANONT *et al*., 2017).

Quando uma arquitetura tem a capacidade de redimensionar os seus recursos para que suas aplicações possam atender a uma determinada carga de trabalho variável ao longo do tempo, diz-se que ela é escalável (MEDEIROS; CAMPOS, 2020). Dentre as estratégias de escalabilidade possíveis, pode-se citar a escalabilidade vertical e horizontal. Onde a escalabilidade vertical corresponde a adicionar mais recursos (CPU e memórias) em uma máquina[1] já existente. E a escalabilidade horizontal, corresponde a adicionar mais recursos, através da adição de mais máquinas[2] (AZEVEDO *et al*., 2020). Dentro de todo esse contexto, está dissertação irá apresentar como todos esses conceitos e tecnologias, foram utilizados na implantação de um *middleware* IoT de forma escalável para ser utilizado por aplicações de mobilidade elétrica multimodal.

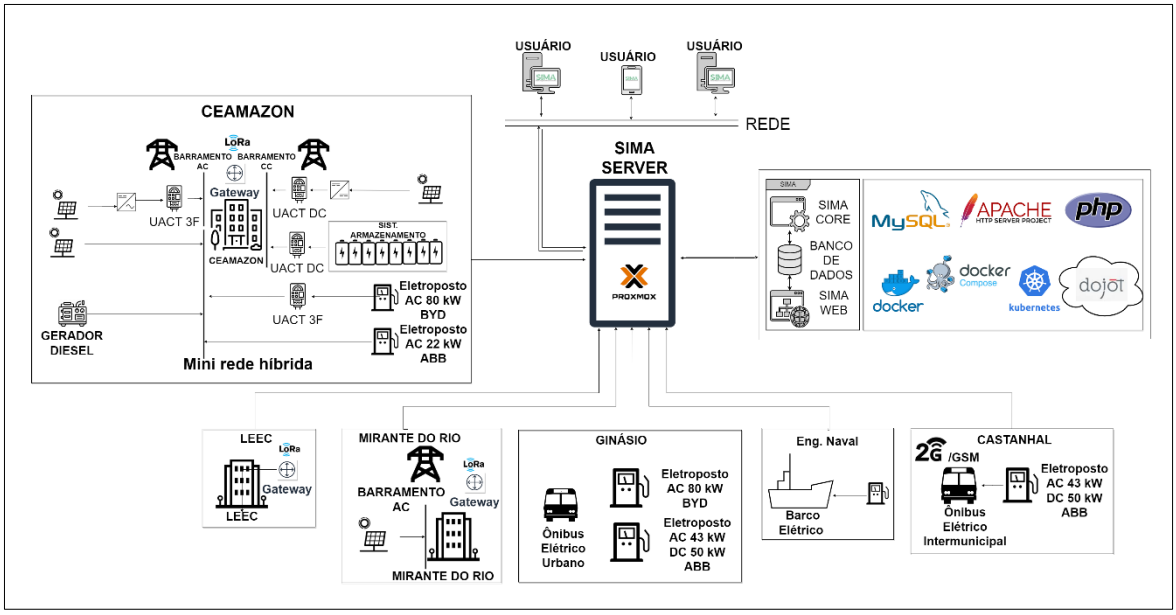
[1] Uma máquina também pode ser chamada de “nó”.
[2] Várias máquinas também podem ser chamadas de “nós”.

1.2 Contexto da Pesquisa

A partir de 2019, o Pará começou a percorrer as estradas e rios amazônicos, a partir da Mobilidade Elétrica Multimodal, através do desenvolvimento do Projeto de Pesquisa e Desenvolvimento (P&D) do Sistema Inteligente da Amazônia (SIMA). Esse projeto foi originado da chamada estratégica de mobilidade elétrica eficiente nº 22/2018 da ANEEL, é financiado pela Norte Energia S.A e é categorizado como um cabeça de série dentro da Cadeia de Inovação da ANEEL. Implantado na Cidade Universitária Professor José Silveira Netto da Universidade Federal do Pará (UFPA), o projeto transformou o *Campus* do Guamá em um "laboratório vivo", a céu aberto, e inseriu nele elementos presentes nos *Campus* e Cidades Inteligentes (LOBATO *et al*., 2021).

O SIMA tem como objetivo o desenvolvimento de um sistema inteligente de mobilidade elétrica multimodal para a Amazônia, composto pelos modais **ônibus elétrico** e **barco elétrico-fotovoltaico (modelo catamarã)**. O SIMA conta ainda, com uma infraestrutura de apoio composta por **geração fotovoltaica**, **armazenamento de energia**, **eletropostos, medidores de energia, rede LoRaWAN** (do inglês, *Long Range Wide Area Network*), para transmissão dos dados; **um *middleware IoT***; **aplicações de mobilidade;** e **sistemas de gestão e monitoramento** de uma parte dessa infraestrutura, conforme ilustrado na Figura 1. Destaca-se, ainda, que o SIMA implantou o primeiro "**Corredor verde**" da região amazônica e irá gerar um **modelo de negócio** sobre mobilidade elétrica específico para essa região.

Figura 1. Visão Geral do Projeto SIMA.



Fonte: produzido pela Autora.

Todos os sistemas e dispositivos irão gerar dados que serão enviados para a plataforma Dojot, a qual é o *middleware IoT* desse projeto. Dessa forma, a Dojot tem a função de receber, armazenar e enviar os dados, quando solicitados, para as aplicações do projeto SIMA. Uma das principais aplicações (produtos) do SIMA, será um “Sistema Inteligente de Gestão Eficiente de Mobilidade Elétrica Multimodal”, composto por outras aplicações que requisitam dados de telemetria[3] armazenados na Dojot.

Diante do exposto, essa dissertação de mestrado irá apresentar a implementação da plataforma Dojot de forma escalável e customizada, como o *middleware IoT* do projeto SIMA. Será descrita a implementação de dois ambientes com a Dojot, sendo um ambiente de desenvolvimento com *Docker Compose*, em uma máquina, e um ambiente de produção em um *cluster Kubernetes*, em duas máquinas. Como forma de verificar qual o consumo das métricas de CPU e memória, foram realizadas simulações de vários dispositivos de mobilidade elétrica recebendo dados simultaneamente na Dojot. Para isso, foram utilizadas as ferramentas, *InfluxDB* e *Grafana*, para monitorar as métricas supracitadas.

[3] Neste trabalho será adotada a nomenclatura de “dados de telemetria” para os dados de medições gerados pelos dispositivos físicos.

1.3 Justificativas

A primeira justificativa para o desenvolvimento deste trabalho, é a sua relevância teórica, pois conforme será visto no Capítulo 3, foram encontrados poucos trabalhos na literatura que utilizam ou, pelo menos, citam a plataforma Dojot, por conta dela ser relativamente “nova” – lançada em 2017 no Brasil. Dos trabalhos encontrados, apenas três mencionam a Dojot aplicada a mobilidade elétrica, são eles Da Silva *et al* (2021), Sá *et al.* (2021) e Moraes *et al* (2021), sendo o último trabalho originado desta dissertação.

Outra justificativa, é a relevância prática desta dissertação para a sociedade. Onde, a implementação da plataforma Dojot como *middleware IoT* do P&D supracitado, possibilitou o desenvolvimento de várias aplicações de mobilidade urbana e elétrica. Facilitando, assim, para os gestores o gerenciamento e o monitoramento da infraestrutura física de mobilidade elétrica instalada. Além disso o projeto como um todo contribui com a melhoria da qualidade de vida da população paraense – especialmente dos frequentadores do *Campus* Universitário do Guamá, onde o P&D é desenvolvido – uma vez que essas pessoas podem utilizar meios de transporte menos poluentes e mais silenciosos para se locomover.

Por fim, destaca-se ainda como justificativa, a viabilidade técnica deste trabalho, uma vez que a Dojot foi instalada, está funcional e em operação, servindo como *middleware IoT* para as aplicações do projeto SIMA. Além, da viabilidade financeira, uma vez que a Dojot é gratuita, ou seja, não precisa realizar nenhum tipo de pagamento para usá-la.

1.4 Motivação

A principal motivação para implantação da plataforma Dojot, tanto no ambiente de desenvolvimento, quanto no ambiente de produção, é garantir a sua escalabilidade (vertical e horizontal) em ambos os ambientes, como também garantir a sua disponibilidade para receber dados e enviá-los para as aplicações quando solicitados. Além disso, outra motivação deste trabalho é verificar se os ambientes de

desenvolvimento e produção criados irão suportar o recebimento de dados de vários dispositivos de mobilidade elétrica previstos no projeto SIMA.

1.5 Objetivos

1.5.1 Objetivo Geral

O objetivo geral dessa dissertação de mestrado é implantar a plataforma Dojot de forma escalável para ser o *middleware IoT* de aplicações de mobilidade elétrica multimodal.

1.5.2 Objetivos específicos

Para alcançar esse objetivo geral foram definidos os seguintes objetivos específicos:

- Implantar a plataforma Dojot em um ambiente de desenvolvimento com *Docker Compose* em uma máquina;
- Implantar a plataforma Dojot em um ambiente de produção em um cluster *Kubernetes* com 2 máquinas *workers*;
- Criar um *IoT Agent* (simulador *Python*), que realize o envio de dados das aplicações de mobilidade elétrica para a Dojot;
- Simular o envio de dados de vários dispositivos a Dojot nos dois ambientes (desenvolvimento e produção) implantados; e
- Comparar o uso de CPU e memória das máquinas utilizadas pela Dojot no ambiente de desenvolvimento (*Docker Compose)* e no ambiente de produção (cluster *Kubernetes*).

1.6 Estrutura do Trabalho

Esta dissertação é composta por seis Capítulos, mais as referências bibliográficas e um apêndice. Sendo estruturados da seguinte forma:

- **Capítulo 2 – Tecnologias utilizadas:** neste Capítulo serão apresentadas todas as tecnologias utilizadas nesta dissertação. Sendo elas: Proxmox, *Docker*, *Docker Compose*, *Kubernetes* e Dojot.
- **Capítulo 3 – Revisão da literatura:** será apresentada uma revisão da literatura, dos trabalhos encontrados que utilizam a Dojot em diversas áreas, inclusive na área da mobilidade elétrica.
- **Capítulo 4 – Implantação de um *middleware IoT* escalável para aplicações de mobilidade elétrica multimodal:** será descrito os procedimentos de instalação, customização e personalização da plataforma Dojot para que ela seja utilizada como *middleware IoT* das aplicações de um projeto de mobilidade elétrica.
- **Capítulo 5 – Resultados e análises:** serão apresentados os resultados e análises do uso de CPU e memória das máquinas que alocam a plataforma Dojot enquanto ela recebe dados simultaneamente em vários dispositivos de mobilidade elétrica de um *IoT Agent* (simulador *Python*).
- **Capítulo 6 – Conclusão:** por fim será feita a conclusão do trabalho, com as considerações finais, apontando algumas propostas de trabalhos futuros e apresentado as produções acadêmicas geradas a partir desta dissertação.

2 TECNOLOGIAS UTILIZADAS

Neste Capítulo 2, inicialmente, será abordada uma visão geral sobre virtualização para o bom entendimento do *software* de virtualização utilizado, o Proxmox. Posteriormente, serão apresentadas, respectivamente, as tecnologias: *Docker*; o orquestrador de contêineres *Kubernetes;* e a plataforma Dojot, sendo esta última a principal tecnologia utilizada neste trabalho.

2.1 Virtualização

A prática de virtualização tem se tornado cada vez mais usual, por grandes e pequenas empresas, como forma de melhorar os serviços e reduzir vários custos com infraestrutura, recursos humanos e consumo de energia (DEWI *et al*., 2019). Indo de encontro ao conceito de "*software*/computação verde" que usa computadores para melhorar a eficiência energética (Guo, 2010 apud MA *et al*., 2012).

A virtualização consiste na possibilidade da criação de componentes como *hardwares*, sistema operacional, armazenamento, processamento, memória e recurso de rede, em ambientes virtuais (ARSLAM *et al*., 2017). A camada e os programas de virtualização são chamados de *hypervisor* (que significa monitor) (JAIN; CHOUDHARY, 2016; RED HAT, 2018a).

Dentro desse contexto de virtualização, a máquina física (real, nativa ou hospedeira) é chamada de *host* e a máquinas virtuais são chamadas de *guest* (MAZIERO, 2008). De acordo com a *Red Hat* (2020)., destaca-se que existem dois tipos de virtualização (*hypervisor*), são elas:

1. **Virtualização do tipo 1**: também chamada de nativa ou *bare-metal*, em que o *hypervisor* é instalado diretamente no *hardware* do *host*

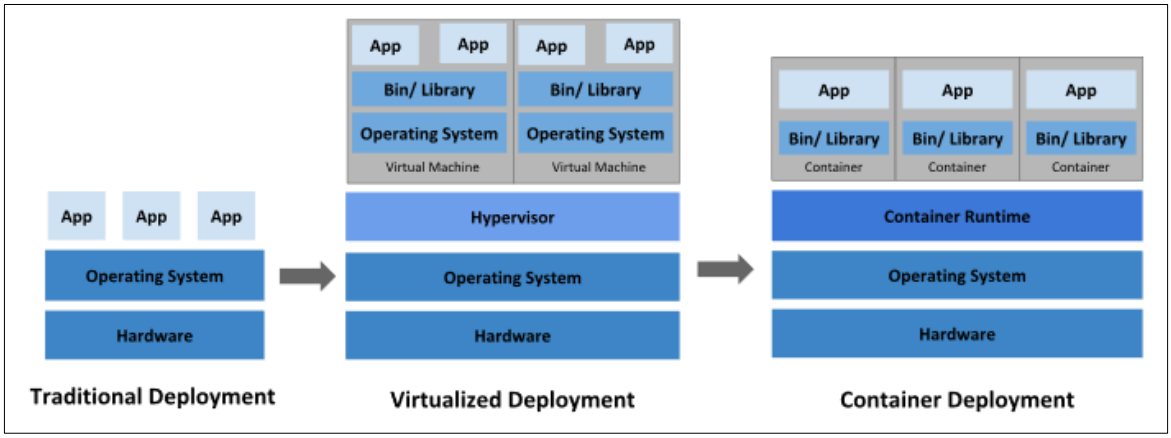
(hospedeiro). Esse tipo de virtualização é mais comum em centro de dados e servidores; e

2. **Virtualização do tipo 2**: também é chamada de *hosted* (hospedado), onde o *hypervisor* é instalado em sobre um sistema operacional convencional já existente, adicionando assim mais uma camada de *software* ou aplicação.

Juntamente com a evolução das técnicas de virtualização ao longo do tempo, as formas de se fazer implantação das aplicações também evoluiu. Inicialmente, antes de existir a virtualização, um sistema operacional era instalado e executado diretamente em uma máquina, fazendo com que o seu *software* e *hardware* ficassem fortemente acoplados e todas as aplicações eram implantadas em uma única máquina. Essa prática, poderia na maioria dos casos, gerar conflitos entre as aplicações (JAIN; CHOUDHARY, 2016).

Posteriormente, com o surgimento da virtualização, passou a existir a possibilidade de se criar várias Máquinas Virtuais (VM, do inglês *Virtual Machines*) em uma única máquina física e realizar a implantação de uma ou várias aplicações dentro dessas VM. Por fim, as técnicas de virtualização evoluíram, o que gerou a possibilidade de se criar contêineres, sendo um contêiner para cada aplicação (KUBERNETES, 2021). A Figura 2 ilustra toda essa evolução da virtualização.

Figura 2. Evolução da virtualização.



Fonte: *Kubernetes* (2021).

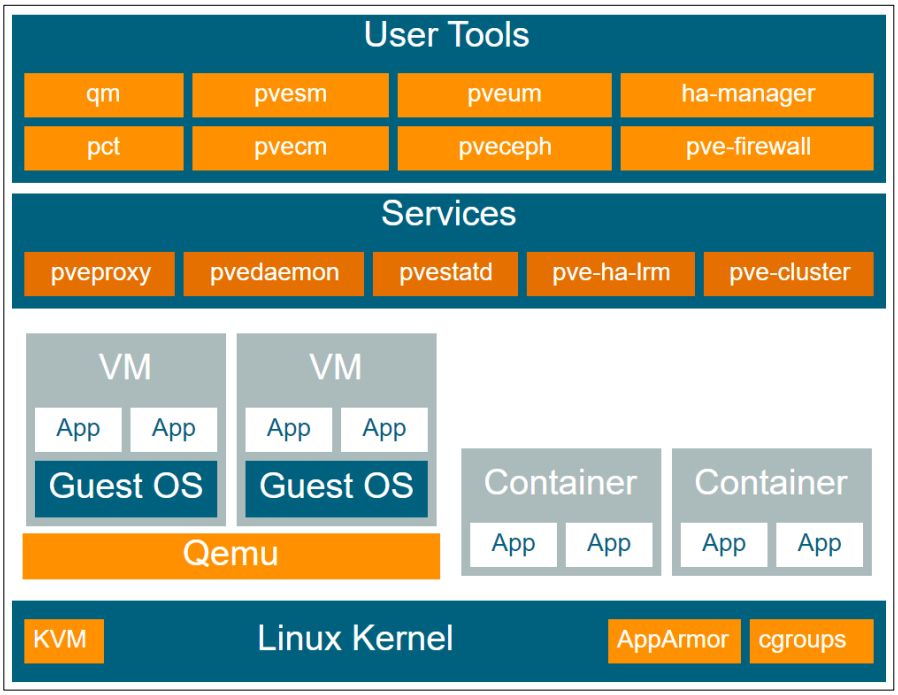
A virtualização baseada em contêineres é a que mais tem ganhado força nos últimos anos (PEREIRA FERREIRA; SINNOTT, 2019). Esse tipo de virtualização também é conhecido como virtualização a nível de sistema operacional (DEWI *et al*.,

2019) ou virtualização leve (ABUABDO; AL-SHARIF, 2019). Ela permite dividir um servidor em várias instâncias, os "contêineres", que compartilham o mesmo *kernel* do sistema operacional.

Os contêineres são semelhantes às VM em relação a ter o seu próprio sistema de arquivos, compartilhamento de CPU, memória, espaço de processo e muito mais. Porém, eles se diferem das VM, em relação às propriedades de isolamento flexibilizadas para poder compartilhar o mesmo sistema operacional entre as aplicações. Justamente por isso, eles são considerados mais leves (MORAES *et al*., 2021).

Lançado em 2008, o Proxmox *Virtual Enviroment* é uma plataforma de *open source* de virtualização (*hypervisor*), com interface web integrada, sistema operacional de distribuição *Linux* baseado em *Debian* com um *kernel* Ubuntu LTS modificado (PROXMOX, 2021). Ele pode ser considerado um *hypervisor* do tipo 1 e suporta os dois tipos de virtualização, isto é, Máquina Virtual baseada em *Kernel* (KVM, do inglês *Kernel-based Virtual Machine*) e virtualização baseada em contêiner (LXC, do inglês *Linux Containers*). Por conta disso, ele consegue executar VM e contêineres, como também possui várias ferramentas e serviços nativos, conforme ilustrado na Figura 3.

Figura 3. Arquitetura do Proxmox.



Fonte: Proxmox (2021).

De acordo com a documentação oficial do Proxmox, um dos seus principais objetivos é facilitar o gerenciamento das VM e contêineres (PROXMOX, 2021). Dentre as suas principais funcionalidades, podem-se citar as seguintes vantagens:

- A possibilidade de gerenciamento de várias VMS, contêineres e até mesmo *clusters*;
- Sua GUI web é “amigável”;
- A presença de interface de linha de comando;
- A facilidade no gerenciamento de usuários e permissões com base em funções; e
- O backup ao vivo, ou seja, sem a necessidade de desligar a máquina.

Destaca-se ainda que, o Proxmox possibilita criar uma Infraestrutura *Hiperconvergente* (HCI, do inglês *Hyper Converged Infrastructure*), bastante útil em implantações que demandam de muita infraestrutura, mas possuem poucos recursos financeiros (PROXMOX, 2021). Em uma HCI, tem-se as seguintes vantagens:

- **Escalabilidade**: expansão contínua de dispositivos de computação, rede e armazenamento, ou seja, é possível dimensionar servidores e armazenamento de forma rápida e independente uns dos outros;
- **Baixo custo**: como o Proxmox é *open source,* ele integra todos os componentes de computação, armazenamento, rede, backup e centro de gerenciamento, substituindo assim uma infraestrutura de computação / armazenamento que pode ser bastante cara;
- **Proteção e eficiência de dados**: serviços como backup e recuperação de desastres;
- **Simplicidade**: fácil configuração e administração centralizada; e
- **Código aberto**: o código-fonte do Proxmox foi lançado sob a *GNU Affero General Public License*, versão 3, isso significa que ele é livre para ser inspecionado e pode receber contribuições, sem a necessidade de depender de um fornecedor.

Além de todos esses recursos supracitados, por padrão o Proxmox também possui a funcionalidade “*Metric Server*” que permite criar um servidor para monitorar: métricas de CPU, memória, I/O (do inglês, *Input/Output*), *hosts*, convidados e

armazenamento (PROXMOX, 2021). Atualmente, o Proxmox suporta os *softwares Grafite* e *InfluxDB* para realizar o monitoramento dessas métricas. Neste trabalho, foi utilizado o *Grafana* (no lugar do *Grafite*), e *InfluxDB*, para monitorar essas métricas.

Assim, por conta de todos esses recursos e funcionalidades o *hypervisor* Proxmox, versão 7.0, foi escolhido para ser instalado diretamente no servidor, caracterizando uma virtualização do tipo 1. Dessa forma, foi possível criar todo o ambiente de virtualização para criação das VM e posteriormente implantação da Dojot.

2.2 Docker

Criado em 2013, o *Docker* é uma plataforma *open source* que serve para o desenvolvimento, envio e execução de aplicações (DOCKER, 2021). De acordo com a sua documentação oficial, no *Docker* é possível escolher a linguagem de programação, o banco de dados e a sua distribuição. Além disso, ele possui a capacidade de empacotar e executar um aplicativo em um ambiente isolado denominado de container (DOCKER, 2021).

Um contêiner é um agrupamento de diretórios, arquivos e dependências de uma aplicação que compartilham o mesmo *kernel* do sistema operacional de uma determinada máquina (*host*), seja ela virtual ou física (VITALINO, 2018) . Vale ressaltar que um contêiner não precisa de um processador com *tags* para virtualização de *hardware* (MERKEL, 2014).

Assim, com o *Docker* é possível isolar as aplicações e executar vários contêineres simultaneamente em um único *host*. No que se refere ao isolamento, é possível estabelecer limites de uso de: memória, CPU, I/O e rede (VITALINO, 2018). Dessa forma, por meio do *Docker* é possível criar *microservices* (microserviços), onde uma grande aplicação é dividida em partes menores e estas por sua vez executam tarefas específicas.

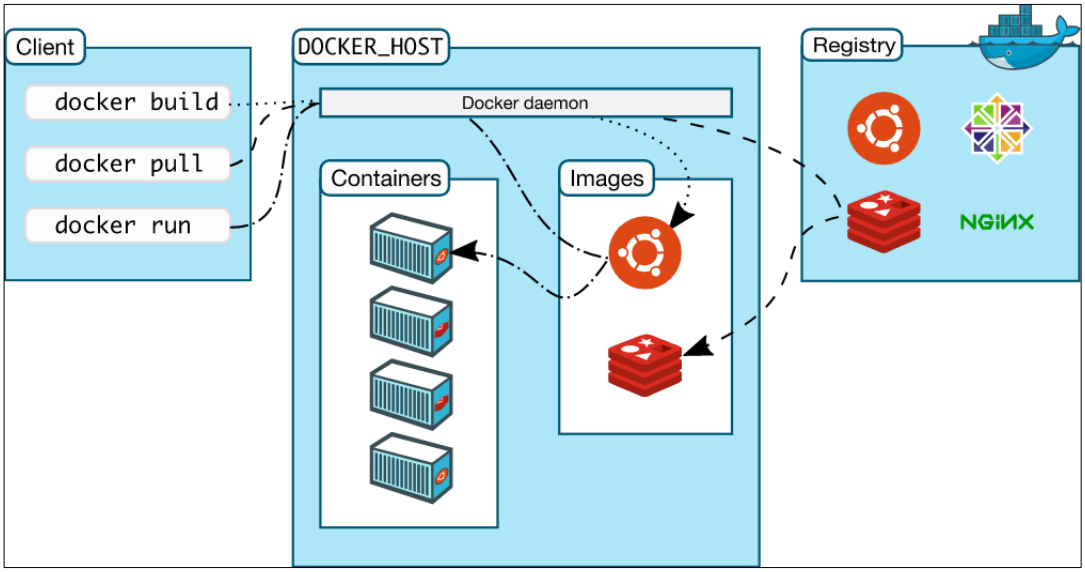
De acordo com Vitalino (2018), dentre as principais vantagens de se utilizar *Docker*, pode-se citar:

- Ganho de agilidade no processo de desenvolvimento das aplicações;
- A diminuição o processo de transição entre os ambientes de desenvolvimento, controle de qualidade e produção; e

- Um melhor aproveitamento e gerenciamento dos recursos de *hardware*, o que resulta na diminuição de gastos financeiros.

O *Docker* utiliza a arquitetura cliente-servidor, conforme ilustra a Figura 4. Nessa arquitetura o *Docker Client* se comunica a partir de uma API REST (do inglês, *Application Programming Interface Representational State Transfer*), com o *Docker daemon*[1] (DOCKER, 2021). Este último por sua vez realiza o trabalho pesado de construir, executar e distribuir seus contêineres *Docker*. O *Docker registry*, por sua vez, armazena as imagens *Docker*. O *Docker client* e o *daemon* podem ser executados juntos, em um mesmo sistema, ou remotamente.

Figura 4. Ilustração arquitetura *Docker*.



Fonte: Docker (2021).

Dentre os principais componentes que compõem o *Docker*, pode-se citar o *Docker-Compose* (VITALINO, 2018). O *Docker Compose* é uma ferramenta capaz de definir e executar aplicações em *Docker* de vários contêineres (DOCKER, 2021). Uma aplicação em *Docker Compose*, pode ser configurada através de um único arquivo, chamado *compose file*, cujo padrão é YML (.yml). Nesse arquivo, geralmente nomeado de "*docker-compose.yml*", é possível definir todos os detalhes de uma aplicação, como por exemplo, quais *services* devem ser criados e quais as características de cada *service* (quantidade de contêineres, volumes, *network*, *secrets* etc.). Após o arquivo *compose file* estar totalmente configurado é possível

[1] *Daemon* é um programa que é executado em segundo plano como se fosse um processo.

criar e iniciar todos os serviços utilizados por uma determinada aplicação com um único comando, de maneira simples, rápida e muito efetiva.

2.3 Kubernetes

Criado originalmente por engenheiros da Google em 7 de junho de 2014, hoje o *Kubernetes* pertence a *Cloud Native Computing Fundation* (CNCF) e está na versão 1.21.3. O *Kubernetes* também pode ser chamado de “K8s”, uma abreviação que contém a primeira e últimas letra, e as oito letras restantes são substituídas pelo número 8, formando (KUBERNETES, 2022) .

De acordo com os desenvolvedores do *Kubernetes*, ele é um orquestrador de contêineres *Open Source* utilizado para automatizar a implantação (instalação), o dimensionamento e o gerenciamento de aplicativos em contêiner. Em ambientes de produção, onde geralmente há mais de uma aplicação sendo executada, ou seja, vários contêineres, é recomendável o uso de ferramentas semelhantes ao *Kubernetes*. No caso do *Kubenetes*, ele é ideal para hospedar aplicações nativas em nuvem que exigem escalabilidade rápida, como por exemplo a transmissão de dados em tempo real (RED HAT, 2018b).

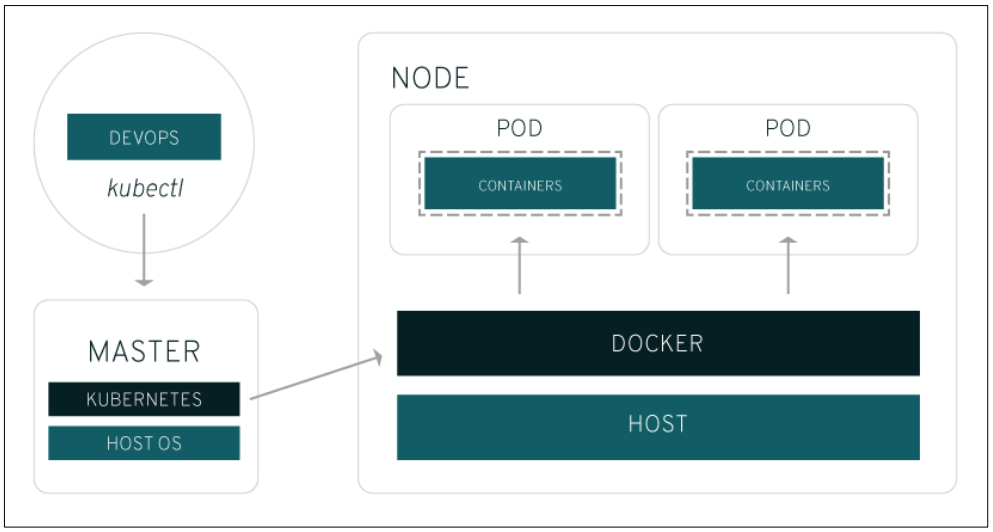
O *Kubernetes* segue o modelo mestre-escravo, sendo que a máquina mestre é chamada de “***master***" e os escravos são chamados de “***nó***" ou “***workers***”. O *master* é usado para controlar os *“**workers**”,* e estes por sua vez realizam as tarefas que lhe são solicitadas ou atribuídas. O *master* e os *workers* juntos formam um “***cluster Kubernetes***”. Quando um grupo de um ou mais contêineres são implantados em um único nó, esse grupo é chamado de “***pod***”.

Cada *pod* é geralmente restrito em relação a uma quantidade máxima de recursos (memória, CPU e armazenamento) que pode consumir (KUBERNETES, 2022). Assim, uma das maiores vantagens do *Kubernetes* é o uso do *hardware* de forma otimizada e inteligente, o que ajuda a economizar tais recursos (DEWI *et al*., 2019).

Existem ainda alguns importantes componentes do *Kubernetes*, como o controlador de replicações, serviço, *kubelet* e *kubectl*. O **controlador de replicações** controla quantas cópias idênticas de um *pod* devem ser executadas em um

determinado local do cluster. Os *proxies*[1] de **serviço** levam automaticamente as solicitações de serviço para o *pod* correto. Já o ***kubelet*** é um serviço que garante que os contêineres definidos foram iniciados e estão em execução. E o ***kubectl*** é a ferramenta de configuração de linha de comando do *Kubernetes* (RED HAT, 2018b). A Figura 5 ilustra todos os componentes descritos anteriormente.

Figura 5.Ilustração arquitetura cluster *Kubernetes*.



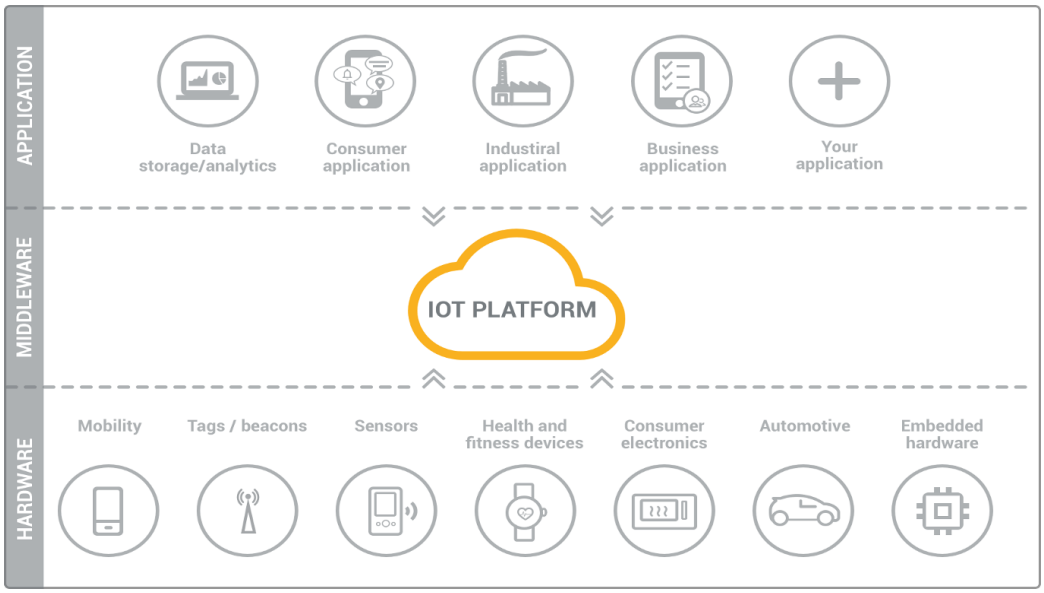
Fonte: *Kubernetes* (2021).

2.4 Middlewares IoT

Um *middleware* é um *software* que está localizado entre a camada de infraestrutura e a camada das aplicações que o utilizam (CHAQFEH; MOHAMED, 2012), conforme ilustra a Figura 6. O seu principal objetivo é reunir diferentes sistemas para que eles possam interagir uns com os outros (HAMMERGREN; SIMON, 2009). Os *middlewares* IoT também podem ser chamados de plataformas de IoT ou plataformas de *middleware* de IoT (DA CRUZ *et al*., 2018).

[1] Segundo Tanenbaum (2003), um proxy pode ser definido como um tipo de servidor que redireciona as requisições de um usuário por algum serviço fornecido.

Figura 6. Ilustração de camadas IoT.



Fonte: Pineli, (2019).

Um *middleware IoT* integra dados de dispositivos, permitindo que eles se comuniquem e tomem decisões com base nos dados coletados (DA CRUZ *et al*., 2018). Ainda de acordo com Da Cruz *et al.* (2018), os *middlewares* IoT possuem vários requisitos, dentre esses requisitos, destacam-se: escalabilidade, disponibilidade, interoperabilidade, segurança e facilidade de implantação, manutenção e uso.

Para Chaqfeh e Mohamed (2012), alguns desses requisitos são apresentados em seu artigo como desafios técnicos dos *middlewares* IoT, como por exemplo, a escalabilidade, interoperabilidade e a segurança. Dentre essas funcionalidades, que também são desafios, destaca-se a questão da escalabilidade que pode influenciar diretamente nas outras funcionalidades. Uma vez que o sistema não é escalável ele pode comprometer, por exemplo, a disponibilidade do sistema e suas aplicações.

Segundo Mitrpanont *et al* (2017), há duas estratégias que podem ser feitas para garantir a escalabilidade de uma plataforma IoT:

1. Melhorar o desempenho de uma única máquina: adicionando mais recurso de CPU e memória, utilizando CPU multicores, entre outros. Essa estratégia seria equivalente a escalabilidade vertical (ZELA; SILVA, 2013).
2. Agrupar várias máquinas/clusters: nessa estratégia é possível adicionar mais máquinas, com a mesma ou mais quantidade de recursos, e as

máquinas dividem o trabalho igualmente entre si. Esta estratégia seria equivalente a escalabilidade horizontal (ZELA; SILVA, 2013).

Hoje existem diversos *middlewares* IoT em nuvem, como por exemplo, *Amazon Web Services*, *Google Cloud Platform*, *Amazon IoT*, *Microsoft Azure*, *IBM Watson,* entre outras (LOPES, 2018), conforme ilustra a Figura 7. Contudo a maioria dessas nuvens são proprietárias e depois de um período de teste (que geralmente varia entre um mês a um ano) é preciso pagar para se ter acesso aos recursos, o que acaba inviabilizando o uso delas em P&D ou em Projetos que não possuem muitos recursos financeiros.

Figura 7. Plataformas IoT.

Plataforma	Responsável	Origem	Aplicações	Tipo
Dojot	CPqD	Brasil	O foco é desenvolver aplicações para cidades inteligentes (segurança pública, mobilidade urbana e saúde).	Pública
Kaa IoT	KAA	Emirados Árabes Unidos	Permite a conexão de atividades agrícolas, industriais, eletrônicos, *Wearables* e saúde.	Pública
Azure IoT Suite	Microsoft	Estados Unidos	Permite a conexão de fábricas, monitoramento remoto, manutenção preditiva, serviço de campo, automóveis e edifícios.	Privada
Thing Space	verizon	Estados Unidos	Permite o gerenciamento de conectividade, localização de dispositivo, serviço de mensagens curtas (SMS), *MapQuest* , entre outros.	Privada
Watson	IBM	Estados Unidos	Permite a conexão de automóveis, eletrônicos, energia e utilidades, seguros, manufatura e varejo.	Privada
AWS IoT	amazon	Estados Unidos	É uma plataforma de nuvem gerenciada que permite a interação fácil e segura de dispositivos com aplicativos de nuvem e outros dispositivos.	Privada
Oracle IoT	ORACLE	Estados Unidos	Permite a conexão entre as mais diversas atividades do negócio, além de poder ser aplicada aos campos da engenharia, construção, saúde e seguros.	Privada
HPE IoT	Hewlett Packard Enterprise	Estados Unidos	É uma plataforma universal que pode ser aplicada nos campos: Serviços financeiros, Manufatura, Mídia e entretenimento, Provedores de serviços, Saúde e Ciências biológicas, Telecomunicações, Setor público e Pequenas/Médias empresas.	Privada
JASPER	CISCO	Estados Unidos	Permite a conexão e o monitoramento em tempo real nos mais diversos campos, tais como: Agricultura, Automóveis, Casas e Construção, Saúde, Equipamentos industriais, Varejo e soluções de pagamentos, Cidades inteligentes, Transportes e Logística.	Privada
HANA	SAP	Alemanha	Permite o processamento de dados de qualquer tipo de máquina, dispositivo, sensor ou atuador e o combina com dados comerciais transacionais, serviços geoespaciais e informações não estruturadas das mídias sociais.	Privada
ARTIK	SAMSUNG	Coreia do Sul	Permite a integração de dispositivos instalados em casas, edifícios, saúde, varejo e indústria.	Privada

Fonte: Lopes (2018).

Dentre as plataformas IoT brasileiras, pode-se citar a Dojot, *Denox*, *Eugenio*, *Geniot* e *Konker*. Destaca-se que, das plataformas citadas, até onde se sabe, apenas a plataforma Dojot é totalmente gratuita, e não possui nenhuma limitação em relação a quantidade máxima de dispositivos conectados e tempo de persistência de dados.

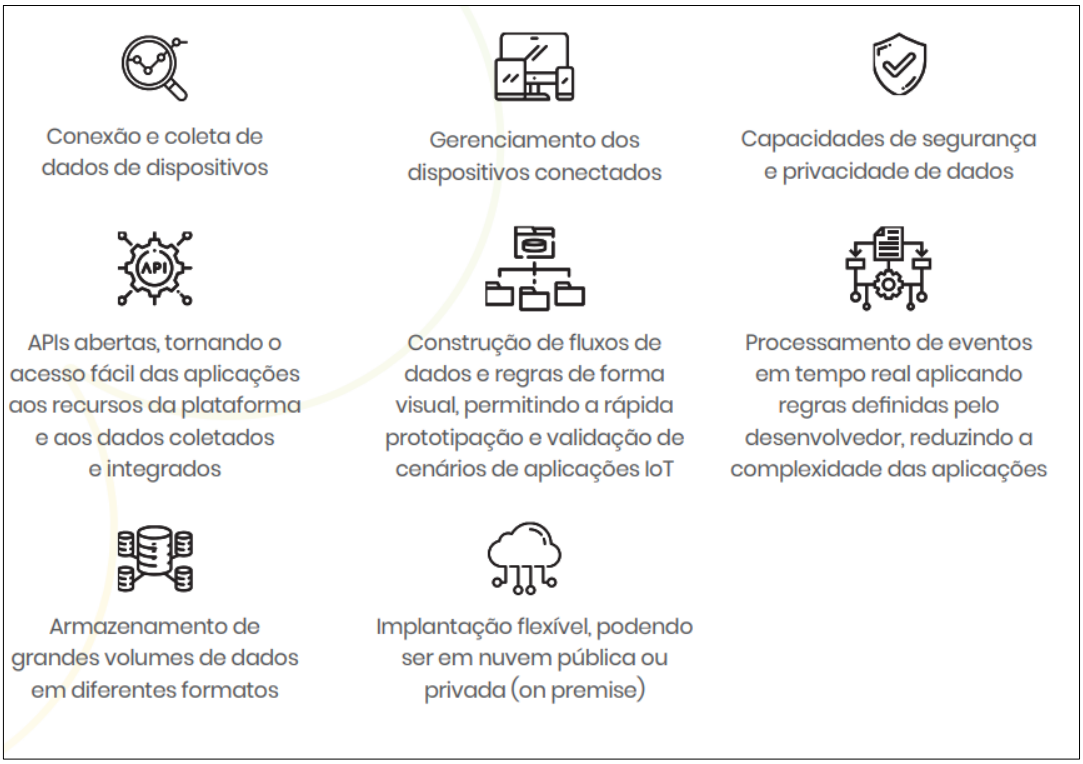
2.5 Plataforma Dojot

Criada em 2017, a Dojot é uma plataforma brasileira *open source* que nasceu com o objetivo de desenvolver e demonstrar tecnologias para as cidades inteligentes e ser utilizada em aplicações de IoT voltadas para as necessidades brasileiras (CPQD, 2017). Desenvolvida pelo Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD), a Dojot utiliza tecnologias no estado da arte e adota uma arquitetura de micro serviços em *Docker*. Dessa forma, sua implantação pode ocorrer de forma funcional, customizada e escalável, isto é, de acordo com a necessidade da aplicação. Além disso, ela possui fácil integração com outros sistemas baseados em serviços (CPQD, 2017).

A Dojot tem como base o *framework FIWARE*. Ele foi criado pela União Europeia e desenvolvido por uma comunidade global independente, que também *é open source* e tem como objetivo o desenvolvimento de soluções inteligentes portáteis, interoperáveis, fáceis, acessíveis e inovadoras (FIWARE, 2021). Dentre as suas principais funcionalidades (Figura 8), de acordo com Goes *et al*. (2019), podem-se citar a possibilidade de:

- Conectar e coletar dados de dispositivos;
- Gerenciar os dispositivos conectados, segurança e privacidade dos dados;
- Permitir a utilizar de API abertas;
- Acessá-la através de um GUI (do inglês, *Graphical User Interface*);
- Construir fluxos de dados de forma visual, processamento de eventos em *real time*;
- Garantir o armazenamento grande volume de dados;
- Instalá-la em nuvem pública ou privada;

Figura 8. Funcionalidades da Dojot.



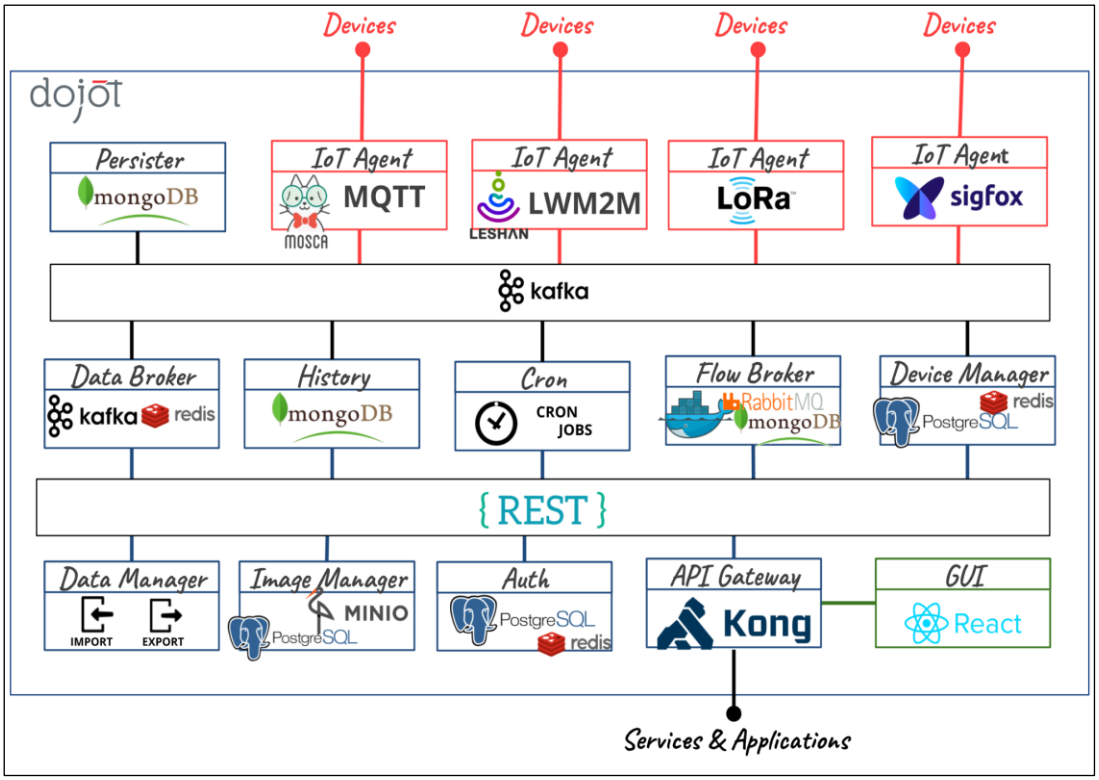
Fonte: CPqD (2017).

Uma das principais características da Dojot, é o fato de suportar a escalabilidade horizontal (CPQD, 2021), através da sua implantação em um cluster *Kubernetes*. Dentre os benefícios de se utilizar a plataforma Dojot, pode-se citar o fato dela ser baseada em padrões internacionais, ser confiável, flexível, livre de *vendor lock in*[1] e gratuita (CPQD, 2017).

Dentro da arquitetura implementada nesta dissertação, a plataforma Dojot tem como função ser o *middleware IoT* de todo o sistema, ou seja, é ela quem recebe, persiste, processa e fornece os dados para as demais aplicações. Na Figura 9 é possível visualizar a arquitetura com os componentes e alguns micro serviços da Dojot. Essa arquitetura faz uso de componentes de código aberto conhecidos e de alguns componentes que foram desenvolvidos pela própria equipe da Dojot.

[1] Expressão em inglês que significa aprisionamento ao fornecedor ou a tecnologia.

Figura 9. Arquitetura da Dojot.



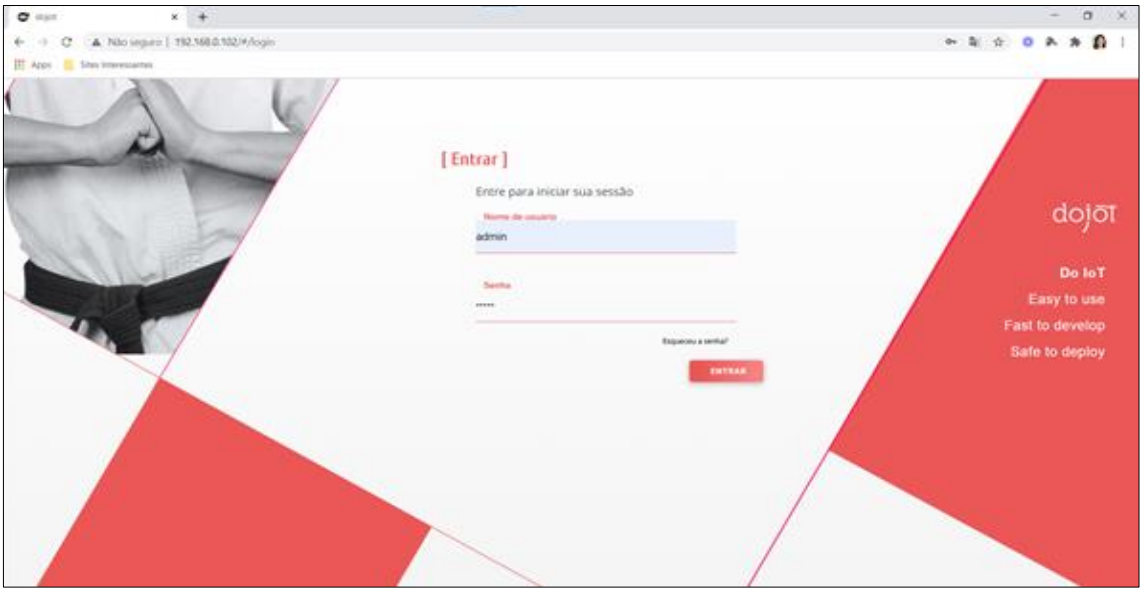
Fonte: CPqD (2021).

Um dos principais componentes presentes na arquitetura da Dojot são os ***IoT Agents***. De acordo com a documentação oficial da Dojot, o *IoT Agent* é um serviço de adaptação entre dispositivos físicos e os componentes da Dojot e pode ser entendido como um *driver* de dispositivo para um conjunto de dispositivos (CPQD, 2021). A plataforma Dojot pode ter vários *IoT Agents*, variando de acordo com o tipo de aplicação, onde cada um deles pode ser especializado em um protocolo específico como, por exemplo, o protocolo **MQTT** (do inglês, *Message Queue Telemetry Transport*) (CPQD, 2021).

O MQTT é um dos principais protocolos de IoT. Ele utiliza a estratégia de *publish/subscribe* de tópicos para transferir dados e tem como principal objetivo minimizar o uso de largura de banda da rede e dos recursos dos dispositivos (SANTOS et al., 2016).

Após o processo de instalação da Dojot, ela pode ser acessada através da sua **GUI** em um navegador de internet, conforme ilustra a Figura 10. Um dos primeiros passos após efetuar o login é criar um modelo (*template*), com os seus respectivos atributos (variáveis) e posteriormente criar um dispositivo (*device*) e vinculá-lo a um ou mais modelo(s) criado(s) anteriormente.

Figura 10. GUI da Dojot.



Fonte: produzido pela Autora.

2.5.1 Componentes

O Quadro 1 a seguir apresenta uma visão geral dos componentes da arquitetura Dojot e suas respectivas funções, com base na sua documentação oficial (CPQD, 2021). Mais informações sobre cada um desses componentes podem ser encontradas diretamente nas respectivas documentações dos próprios componentes.

Quadro 1 – Visão geral dos componentes da Dojot.

Componete	**Função**
Persister	Componente responsável por conduzir todos os dados de telemetria e eventos gerados pelos dispositivos para serem armazenados no banco de dados *MongoDB*.
IoT Agent	Serviço que faz a adaptação entre os dispositivos físicos e os componentes principais do Dojot.
Kafka + Data Broker	Componente responsável por transmitir ou consumir dados em canais através de tópicos.
History	Micro serviço que consulta os dados de telemetria persistidos no banco de dados através da *API* Rest.
Cron	Micro serviço que permite agendar eventos a serem emitidos – ou requisições a serem feitas – para outros micros serviços dentro da Dojot.
Flowbroker	Serviço que fornece mecanismos para construir fluxos de processamento dos dados de telemetria para executar um conjunto de ações.
Device Manager	Entidade central responsável por armazenar as estruturas de dados de dispositivos e modelos (*templates*). Ele também é responsável por publicar quaisquer atualizações para todos os componentes interessados por meio do *Kafka*.
Data Manager	Serviço que gerencia a configuração de dados da Dojot, possibilitando importar e exportar os dados de modelos (*templates*), dispositivos, fluxos e tarefas agendadas da Dojot através de um arquivo JSON (.json). Vale ressaltar que os dados de telemetria não fazem parte da função de importar e exportar.
Image Manager	Componente responsável pelo armazenamento e recuperação de imagens de *firmware* de dispositivos.
Serviço de autorização de usuário (*Auth*)	Serviço que gerencia perfis de usuários e controla os acessos.
Kong API Gateway	É utilizado como um ponto de fronteira (*entry point*) entre as aplicações e serviços externos e os serviços internos da Dojot.
GUI	É uma aplicação *web* que provê interfaces responsivas para gerenciamento da Dojot e inclui as seguintes funcionalidades de gerenciamento: perfil de usuários, usuários, modelos de dispositivos, dispositivos, fluxos de processamento, notificações, alterar a senha, importar e

	exportar a Dojot.
Kafka2Ftp	Serviço *kafka2ftp* permite o encaminhamento de mensagens do *Apache Kafka* para servidores FTP (do inglês, *File Transfer Protocol*).
InfluxDB Storer e Retriever	Os serviços *InfluxDB Storer* e *InfluxDB Retriever* trabalham juntos. O *InfluxDB Storer* é responsável por consumir os dados *Kafka* de dispositivos e gravá-los no *InfluxDB,* Já o *InfluxDB Retriever* tem a função de obter os dados que foram escritos pelo *InfluxDB Storer* no *InfluxDB* via API REST .
Gerenciador de identidade X.509	Componente responsável por atribuir identidades a dispositivos, tais identidades são representadas na forma de certificados x.509
Kafka WS	Componente é responsável por obter dados em tempo real do *Apache Kafka* via uma conexão *websocket*.

Fonte: produzido pela Autora com base em CPQD (2021).

2.5.2 Infraestrutura

O Quadro 2 apresenta outros componentes utilizados na arquitetura da Dojot, com base na sua documentação oficial (CPQD, 2021). Mais informações sobre cada um desses componentes podem ser encontradas diretamente na sua própria documentação.

Quadro 2 – Componentes da infraestrutura da Dojot.

Componente	**Função**
Postgres	Banco de dados utilizado para persistir informações de vários componentes, como, por exemplo, do gerenciador de dispositivos.
Redis	Banco de dados em memória usado como cache em vários componentes, como o serviço de orquestração, gerenciador de subscrição, agentes IoT e outros.
RabbitMQ	*Broker* de mensagens usado no orquestrador de serviço para implementar fluxos de ação relacionados que devem ser aplicados a mensagens recebidas de componentes.
Mongo	Banco de dados usado para armazenar os dados recebidos na Dojot.
Zookeeper	Manter sob controle os serviços replicados em cluster.

Fonte: produzido pela Autora com base em CPQD (2021).

2.5.3 Comunicação

Ainda de acordo com a documentação oficial da Dojot (CPQD, 2021), todos os componentes na Dojot podem se comunicar de duas formas, isto é, a partir de (1) **Requisições HTTP** ou (2) **Mensagens Kafka**.

1. **Requisições HTTP** (do inglês, *Hypertext Transfer Protocol*): se um componente necessita recuperar informações de um outro componente, ele pode enviar uma requisição HTTP. Por exemplo, se um *IoT Agent* precisa da lista de dispositivos configurados, ele deve enviar uma solicitação HTTP para o gerenciador de dispositivos.
2. **Mensagens *Kafka***: se um componente precisa enviar novas informações sobre um determinado recurso controlado por ele, esse componente pode publicar essas informações através do *Kafka* e assim qualquer outro componente que esteja interessado em tal informação precisa apenas estar “ouvir” um tópico específico (CPQD, 2021).

2.6 Considerações Finais do Capítulo

Neste Capítulo conhecemos as tecnologias utilizadas neste trabalho, bem como importantes conceitos e definições de termos utilizados por essas tecnologias. Além disso, foi apresentada a plataforma Dojot juntamente com todos os seus componentes, infraestrutura e formas de comunicação.

3 REVISÃO DA LITERATURA

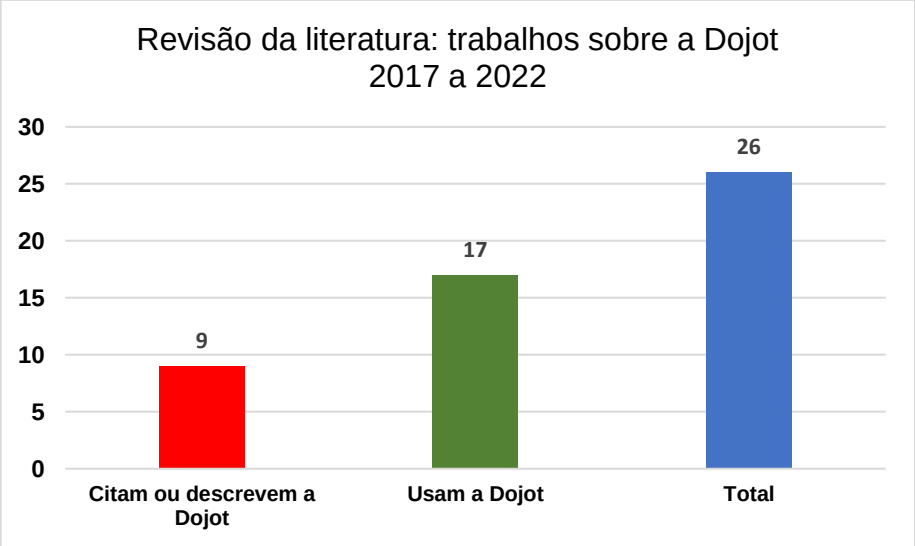
Neste Capítulo 3 serão apresentados os alguns trabalhos encontrados na literatura sobre a plataforma Dojot. Por se tratar de uma plataforma IoT relativamente nova no Brasil (lançada em 2017), ainda são encontrados poucos trabalhos na literatura. Dessa forma serão, descritos os critérios utilizados para realizar a pesquisa dessa revisão da literatura e, posteriormente, serão apresentados os trabalhos encontrados.

3.1 Descrição da pesquisa

Foi pesquisado pela palavra-chave “Dojot”, na base de dados do Google Acadêmico e diretamente na ferramenta de pesquisa do Google, considerando o período de 2017 a 2022. Assim, dentro do período pesquisado foi encontrado um total de 23 trabalhos, relacionados a Dojot. Desse total mencionado, 9 trabalhos apenas citam ou descrevem a Dojot e 17 trabalhos afirmam que utilizam a plataforma Dojot em algum tipo de aplicação ou na sua arquitetura, conforme ilustra o Gráfico 1. Desses 17 trabalhos, 3 deles abordam a Dojot sendo aplicada especificamente na área de mobilidade elétrica.

Dentro do escopo do projeto SIMA, também foram publicados alguns trabalhos relacionados a plataforma Dojot, que podem ser encontrados nas bases de dados dos respectivos eventos onde foram publicados. Alguns desses trabalhos (PINTO *et al*., SILVA *et al*. 2020; SILVA *et al*., Sá *et al*. 2021), serão apresentados na revisão que será apresentada a seguir.

Gráfico 1. Trabalho encontrados na literatura sobre a Dojot

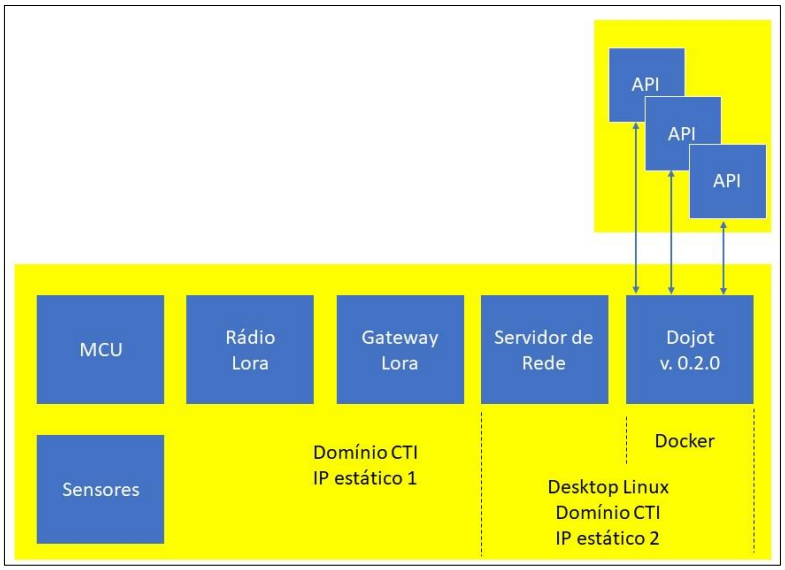


Fonte: produzido pela Autora.

3.2 Trabalhos relacionados a plataforma Dojot

No trabalho de Goes *et al*. (2019) é proposta uma arquitetura de referência para soluções de IoT utilizando a plataforma Dojot, em uma aplicação com uma estação meteorológica. Neste trabalho são apresentadas arquiteturas sistêmicas gerais voltadas para IoT, composta por sensores, atuadores, conectividade, processos e pessoas. A arquitetura sistêmica proposta por Goes *et al*. (2019), que pode ser visualizada na Figura 11, é dividida em três estágios: no estágio 1, há a presença de sensores, um microcontrolador Arduino, rádio LoRa e um *gateway* LoRa; no estágio 2 há um servidor de rede e a Dojot versão 0.2.0; e no estágio 3 estão as aplicações consumindo dados da Dojot por meio das API.

Figura 11. Arquitetura de referência para aplicações de IoT utilizando a Dojot.



Fonte: Goes *et al*. (2019).

Nessa arquitetura os dados de telemetria são aferidos pelos sensores são “lidos” pelo microcontrolador, armazenados em um cartão de memória e enviados periodicamente através do rádio LoRa. O *gateway* LoRa, por sua vez, recebe esses dados e encaminha para o servidor LoRaWAN, através do número IP (do inglês, *Internet Protocol*) do servidor que está configurado como IP estático. Esse servidor também pode ser usado como um repositório desses dados e para o seu pré-processamento, caso precise. Após fazer a análise e a interpretação (*parsing*) dos dados recebidos, esses dados são enviados para a Dojot que também possui um IP estático.

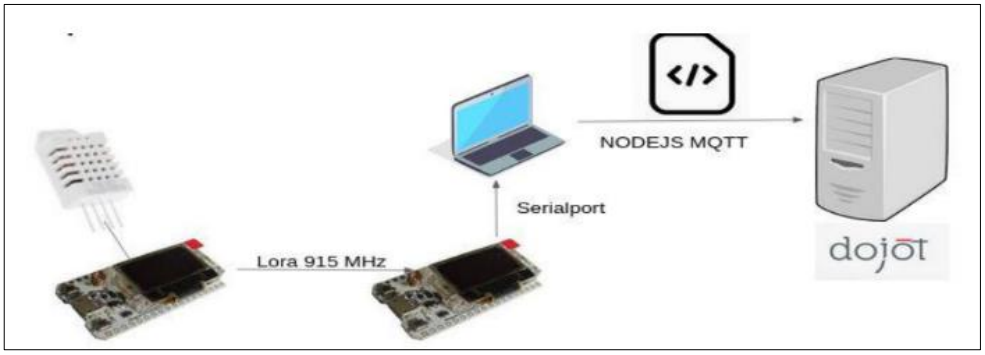
Os autores, Goes *et al*. (2019), destacam ainda que todos os componentes (sensores, microcontrolador, rádio LoRa, servidor LoRaWAN e a Dojot) que compõem essa arquitetura proposta, podem estar a quilômetros de distância uns dos outros. Essa possibilidade é garantida por conta da utilização da tecnologia LoRa. Ressaltando-se, assim, o principal objetivo da implementação da arquitetura proposta por Goes *et al*. (2019) que foi validar a conectividade LoRa.

A arquitetura do trabalho de Goes *et al*. (2019), se assemelha à arquitetura do ambiente de desenvolvimento implementada nesta dissertação, conforme será visto na Subseção 4.3.2. Outros pontos de semelhança são em relação a utilização do IP estático onde a Dojot está alocada e a utilização da tecnologia LoRa para envio dos dados.

A arquitetura do trabalho de Goes *et al*. (2019) se difere da arquitetura implementada nesta dissertação em relação a versão da Dojot, onde na arquitetura proposta nesta dissertação é utilizada a versão 0.7.0. Essa versão, é a versão mais atual e estável da Dojot até agora. Ressalta-se que na versão 0.7.0 houve correções de *bugs* bastante significativos.

No trabalho de Pinto *et al.* (2020) a plataforma Dojot foi implantada com *Docker-Compose* para receber dados enviados através da tecnologia *LoRa*. A Figura X ilustra a arquitetura criada, onde foram utilizados, um (1) sensor de umidade e temperatura DHT22, dois (2) ESP *LoRa Heltec* para serem os *endnodes* e realizarem a comunicação sem fio, um (1) notebook para enviar dados via protocolo MQTT e se comunicar com *gateway* LoRa e um (1) servidor com sistema operacional Linux Ubuntu 18.04 onde foi instalada a Dojot. Apesar de não ser ilustrado na arquitetura da Figura 12, os autores relatam que também foi utilizado 01 *gateway* LoRa para enviar os dados para a Dojot.

Figura 12. Arquitetura proposta para enviar dados via LoRa.



Fonte: Pinto *et al.* (2020).

O principal objetivo do trabalho de Pinto *et al.* (2020) era avaliar os dados recebidos através da tecnologia LoRa. Mediante a isso, foram feitas campanhas de medições dentro do *Campus* Guamá da UFPA, conforme ilustra a Figura X, onde foram enviados os dados de temperatura, umidade, distância, RSSI (do inglês *Received Signal Strength Indication*) e SNR (do inglês *Signal-to-Noise Ratio*). Dessa forma, com as campanhas de medições foi possível identificar a variação de perda de sinal, à medida que houve um aumento da distância entre o *gateway* e *endnode*. Os autores concluíram ainda que o ecossistema Dojot é viável para a recepção de dados advindos de redes IoT LoRa e enviados através de protocolo MQTT.

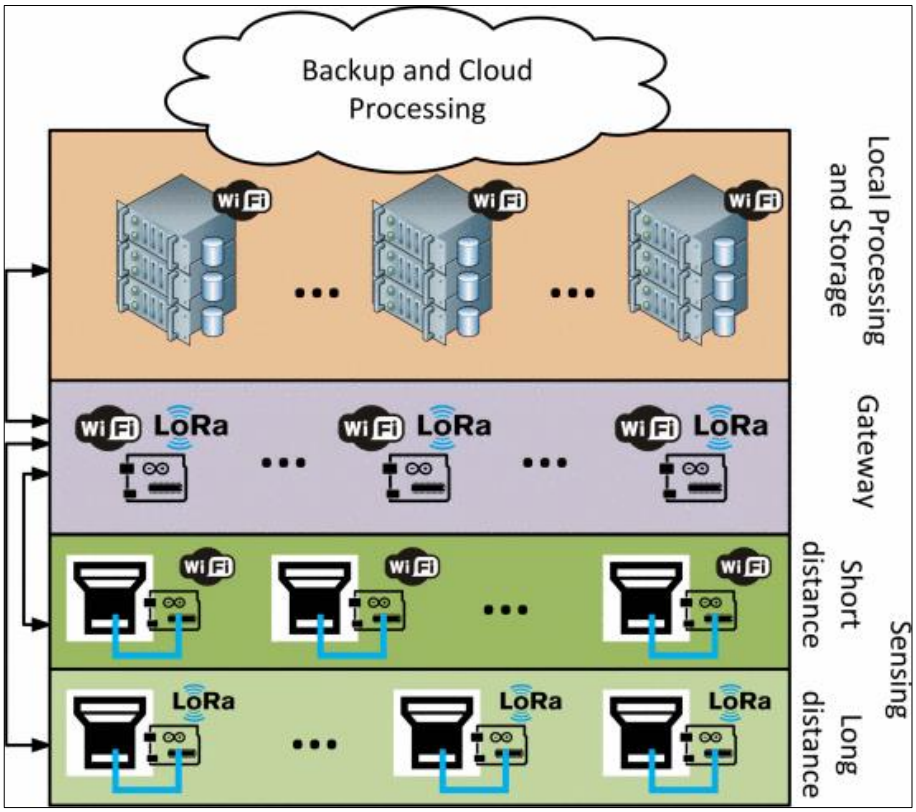
Figura 13. Pontos de medição no Campus Guamá da UFPA.

Fonte: Pinto *et al.* (2020).

A arquitetura do trabalho de Pinto *et al.* (2020) se assemelha à arquitetura do ambiente de desenvolvimento implementada nesta dissertação, conforme será visto na Subseção 4.3.2. Além disso o servidor onde a Dojot foi implantada é semelhante ao servidor utilizado nesta dissertação em relação às suas especificações técnicas.

Outro trabalho que utiliza a plataforma Dojot é o de Santos *et al.* (2020). Onde eles apresentam o protótipo de um aplicativo IoT para monitoramento de chuvas baseado em tecnologias de comunicação sem fio (*WiFi* e LoRa). O aplicativo consome dados da plataforma Dojot implantada com *Docker-Compose*, na versão 0.4.2. A arquitetura proposta possui três camadas: camada de sensoriamento, onde residem os medidores digitais de chuva responsáveis pela coleta de dados; a camada do *gateway*, responsável por fornecer as tecnologias de comunicação adequadas para comunicar servidores e sensores; e a camada de processamento e armazenamento Local, onde os servidores são implantados, um para cada região de interesse, conforme ilustra a Figura 14. O objetivo do trabalho de Santos *et al.* (2020) foi avaliar a utilização das redes *WiFi* e LoRa.

Figura 14. Arquitetura IoT utilizando a Dojot para aplicação de monitoramento de chuva.



Fonte: Santos *et al*, (2020).

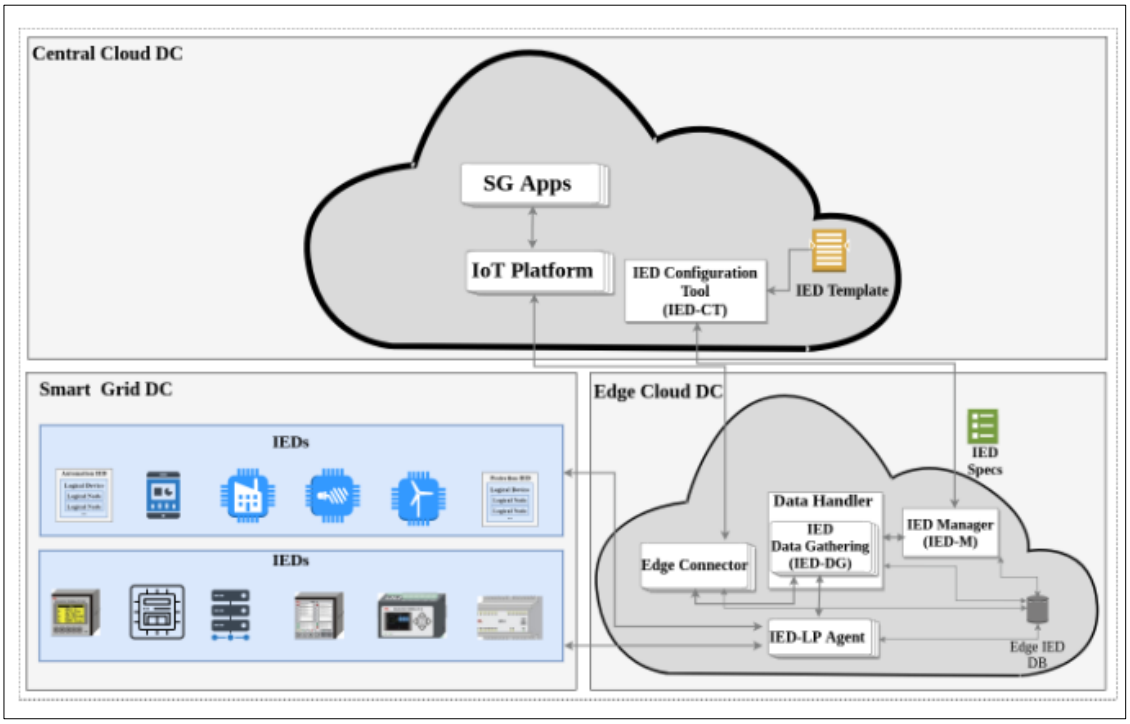
Xavier (2020) implantou a plataforma Dojot para ser utilizada com uma aplicação de monitoramento de distância entre objetos. No seu trabalho há um Capítulo inteiro dedicado a apresentar outras plataformas de *middlewares* IoT, semelhantes a Dojot, como por exemplo a *Sentilo, OpenIoT, Smart Santander, Cidap, Concinnity, Clout* e a *FIWARE*. Das plataformas citadas, destaca-se a *FIWARE*, que serviu como base para o desenvolvimento da Dojot.

O objetivo do trabalho de Xavier (2020) foi desenvolver uma aplicação de monitoramento de distância para ele poder avaliar as funcionalidades da Dojot. Para isso, ele implantou o ambiente de desenvolvimento da Dojot com *Docker Compose*, semelhante ao ambiente de desenvolvimento desta dissertação. Contudo a sua aplicação possuía apenas dois modelos e dois dispositivos. Além disso, ele comparou a Dojot com a plataforma *FIWARE* em relação às suas funcionalidades de criar modelos, dispositivos, fluxos, entre outras.

Conforme supracitado, a plataforma *FIWARE* serviu de base para o desenvolvimento da plataforma Dojot. Em relação a *FIWARE*, ela é um *middleware open source* voltado para o desenvolvimento de soluções inteligentes, portáteis e

interoperáveis (FIWARE FOUNDATION, 2021). No trabalho de Modesto *et al.* (2021), é proposto o SG2IoT (do inglês, *Legacy Smart Grid to IoT Protocol Integration Approach*) que utiliza a plataforma IoT *FIWARE* em sua arquitetura. O sistema proposto, permite a coexistência de múltiplos dispositivos eletrônicos inteligentes (IED, do inglês *Intelligent Electronic Devices*) em um ambiente escalável e flexível, possibilitado pelo ecossistema *Smart Grid – Cloud – IoT*. Apesar de a Figura 15 ilustra apenas uma única máquina com o SG2IoT, os autores destacam que ele pode ser implantado de forma distribuída em várias máquinas.

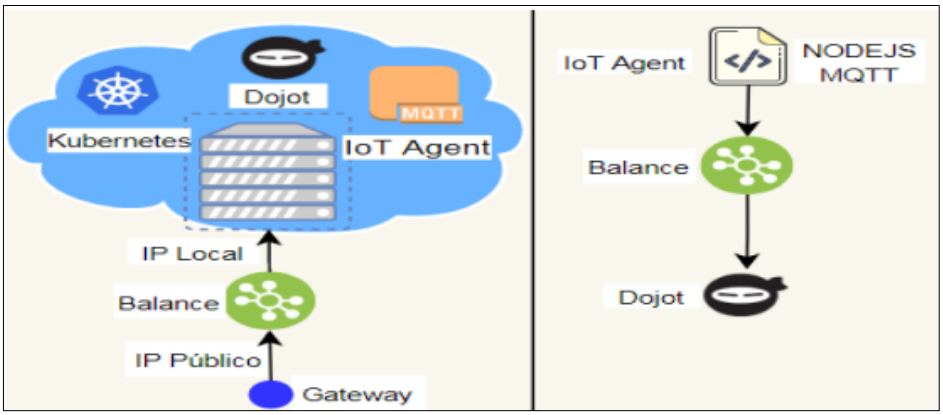
Figura 15. Arquitetura SG2IoT.



Fonte: Modesto *et al.* (2021).

No artigo de Carvalho, Sá e Farias (2021), é apresentado um projeto de *software* para monitoramento ambiental, em tempo real, utilizando a plataforma Dojot, em um cenário de *Smart Campus*. No projeto, foi proposta uma rede IoT, composta por sete *endnodes* e um *gateway* LoRa, para monitorar os dados de temperatura, umidade e dos gases: monóxido de oxigênio (CO), Dióxido de Oxigênio CO2, mais conhecido como gás carbônico, e butano (C4H10). Dessa forma, foi proposta uma arquitetura, composta pela Dojot implantada em um *cluster Kubernetes* e uma IoT *Agent* que simula os dados dos *endnodes* e do *gateway*, conforme ilustra a Figura 16.

Figura 16. Arquitetura IoT proposta para software de monitoramento ambiental.



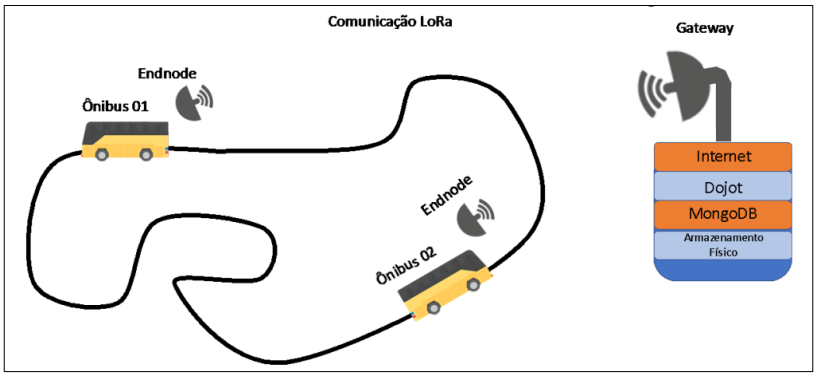
Fonte: Carvalho, Sá e Farias (2021).

O *cluster Kubernetes* onde a Dojot foi instalada é composto por uma máquina *master*, um *worker* e um balanceador de carga *Nginx*. O IoT *Agent*, é representado por um *script* que simula os equipamentos (*gateway* e *endnodes*) instalados no *Campus* de Cametá da UFPA. Toda essa infraestrutura foi implantada em um servidor com processador *quard core* com 4 núcleos e 2.9GHz por núcleo, 6GB de memória RAM e 120GB de armazenamento.

De acordo com os autores, Carvalho, Sá e Farias (2021), os resultados obtidos demonstraram que a arquitetura da Dojot em nuvem é viável para a comunicação, recepção, armazenamento e fácil manutenção de dados ambientais coletados a partir de dispositivos instalados em campo. Eles destacam ainda que a arquitetura proposta, da Dojot implantada em um *cluster Kubernetes*, é uma alternativa satisfatória para a implantação em um *Smart Campus*.

Agora, abordando a utilização da Dojot em aplicações de mobilidade urbana, pode-se citar o trabalho Silva *et al*. (2020), onde a Dojot foi utilizada como um banco de dados em tempo real para receber dados de geolocalização dos ônibus circulares da UFPA, conforme ilustra a Figura 17. Nesse cenário os *endnodes*, que serão futuramente instalados nos ônibus do projeto SIMA, enviam para o *gateway* os dados de latitude e longitude dos ônibus, através de uma rede sem fio LoRA. O *gateway* por sua vez, envia os dados para a Dojot que os persiste no seu banco de dados *MongoDB*.

Figura 17. Ilustração arquitetura em aplicação de mobilidade urbana.



Fonte: Silva *et al.* (2020).

Para testar a utilização do banco de dados em tempo real na plataforma Dojot, foi criado um *script* (que pode ser considerado um *IoT Agent*), em linguagem *Javascript*, que simulou a trajetória em tempo real dos ônibus circulares da UFPA. A Figura 18 ilustração os 7 pontos de deslocamento do ônibus através da interface gráfica da Dojot. Dessa forma, os autores destacam a possibilidade de visualizar praticamente em tempo real o deslocamento do ônibus através da própria Dojot, sem a necessidade de uma aplicação externa. Além disso, os autores apontam como trabalho futuro desenvolvimento de um *software* para o usuário final que seja capaz de consumir a geolocalização de ônibus e barcos em tempo real.

Figura 18. Interface gráfica da Dojot apresentando mapas recebendo dados em tempo real.

Fonte: Silva *et al* (2020).

Ainda nessa área de mobilidade urbana, outro trabalho também desenvolvido por Silva *et al.* (2021) descreve a aplicação da Dojot para receber dados de geolocalização dos ônibus circulares da UFPA e o consumo desses dados por um *software* capaz de gerar possíveis rotas para os usuários. Os dados de geolocalização foram gerados e enviados por um *IoT Agent*. Este, por sua vez,

utiliza dois algoritmos, onde um algoritmo serve para gerar rotas para usuários irem de um ponto inicial, a um ponto final, a pé, de ônibus, ou rotas integradas podendo ir uma parte do percurso a pé e outra parte de ônibus. E o outro algoritmo prever o horário de chegada dos ônibus circulares em pontos de parada do *Campus Guamá* da UFPA, conforme ilustra a Figura 19.

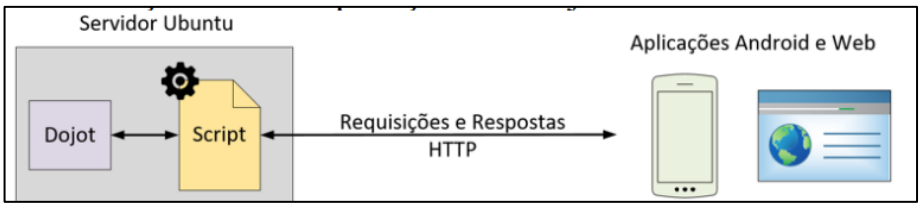
Figura 19. Planejamento automático de rotas por algoritmos.

Fonte: Silva *et al* (2021).

Para que a possível rota seja gerada o usuário deve inserir um ponto inicial e final, e os algoritmos geram as rotas a pé e de ônibus. Sendo que um desses algoritmos faz a previsão do horário em que o ônibus chegará nos seus pontos de parada. Apesar de o artigo não citar, destaca-se que um dos ônibus circulares presente no *Campus*, simulado pelo *IoT Agent*, é elétrico e pertence ao projeto SIMA.

Dentre os poucos trabalhos encontrados na literatura que afirmam que utilizam a Dojot aplicada na área da mobilidade elétrica, está o artigo de Sá *et al.* (2021), onde foi realizado um estudo de caso da comunicação da Dojot com aplicações *Android* e *web*. Ressalta-se que essas aplicações são de mobilidade elétrica. Dessa forma, foi criada a arquitetura de comunicação ilustrada na Figura 20, onde a Dojot foi implantada em um servidor Ubuntu, juntamente com um *script* (que pode ser considerado um *IoT Agent*) que simula o trajeto do ônibus elétrico circular no *Campus* Guamá da UFPA. As aplicações, por sua vez, realizam requisições HTTP do tipo GET a Dojot para consumir os seus dados.

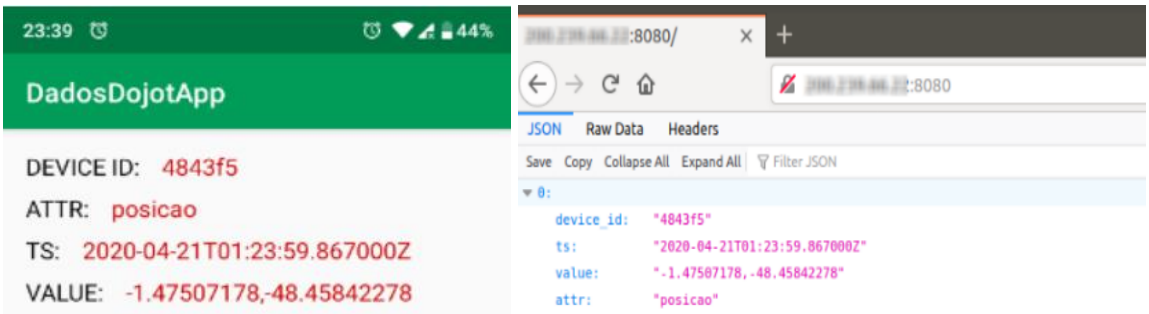
Figura 20. Arquitetura de comunicação entre a Dojot e aplicações *Android e web*.



Fonte: Sá *et al.* (2021).

Na Figura 21 é possível visualizar em uma aplicação *Android* e um navegador *web* os dados solicitados a Dojot. Os autores concluem o seu trabalho destacando que a Dojot é eficiente para armazenar e recuperar dados utilizando aplicações *Android* e *web*.

Figura 21. Aplicação Android e navegador *web* exibindo dados da Dojot.



Fonte: Sá *et al.* (2021).

Em Da Silva *et al*. (2021), eles descrevem o projeto *Smart Campus* que está em execução na Universidade de Campinas (UNICAMP). No trabalho é proposta uma nova metodologia composta por TIC e vários softwares, que formam uma plataforma, que ajudam a melhorar a gestão do *Campus*, através de uma ferramenta inovadora de gerenciamento de energia baseada em IoT

Dentro desse projeto, a Dojot faz o papel de *middleware* entre os sensores/coletores e o banco de dados. Permitindo, dessa forma, a exportação de dados, de acordo com o perfil em uso no software, para a análise integrada no sistema ou uso *offline*. Dentre os dados que podem ser coletados, pode-se citar:

- Rastreamento de várias séries temporais, incluindo medições em tempo real em todo o *campus* e métricas de desempenho de várias entidades;
- Medidores de consumo elétrico;
- Medidores de consumo de água;
- Geradores fotovoltaicos;

- Estações meteorológicas;
- Sensores de IoT distribuídos em geral;
- Ônibus elétricos e outros veículos, incluindo rastreamento em tempo real;
- Dados GIS (do inglês *Geographic Information System*) e parâmetros elétricos da rede elétrica interna;
- Dados GIS gerais, como caminhos, pontos de interesse e informações de construção;
- Rastreamento geral de ativos;
- Modelos detalhados de acordo com os estudos de destino;
- Parâmetros e resultados de simulação

Os dados coletados alimentarão a plataforma de *software* e ajudarão os tomadores de decisão nas áreas de design de eficiência energética, gerenciamento de operação e serviços inteligentes. Além disso, esses dados também apoiam os tomadores de decisão nas áreas de design de eficiência energética, gerenciamento de operação e serviços inteligentes. A plataforma criada faz parte de uma iniciativa *Smart Campus* para transformar a UNICAMP em um “laboratório vivo” e ser um modelo replicável para outros campi sustentáveis.

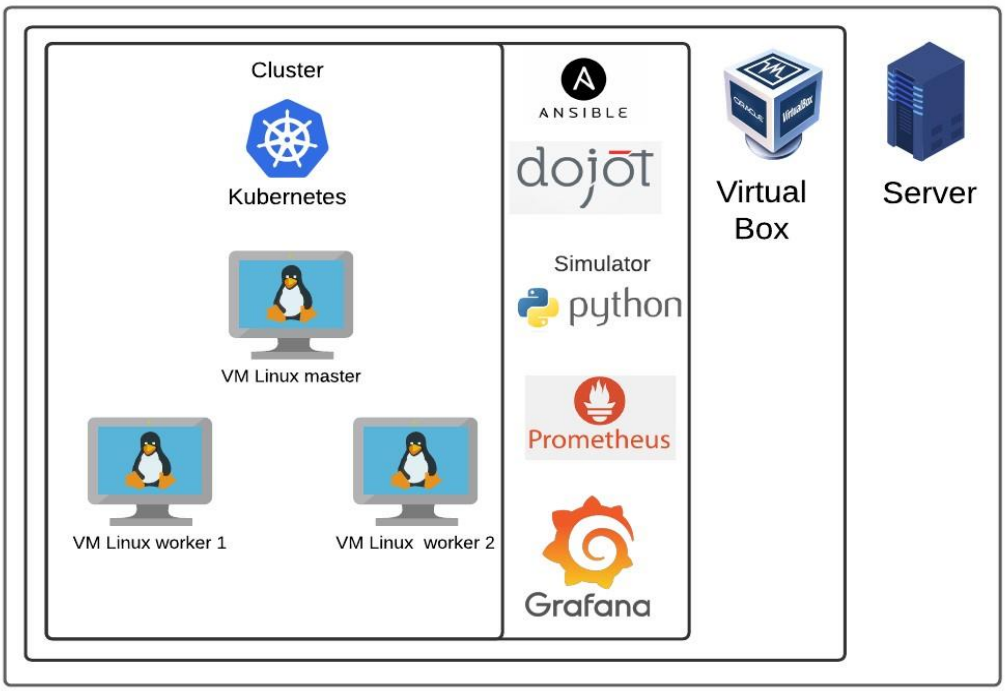
Através do que foi descrito anteriormente, percebe-se que no geral o trabalho de Da Silva *et al.* (2021), que descreve sobre o projeto *Smart Campus*, possui bastante similaridade com o trabalho desenvolvido no projeto SIMA. Sendo o principal ponto em comum a utilização da plataforma Dojot com o *middleware* IoT e sendo utilizada por aplicações de mobilidade elétrica, composta também por um ônibus elétrico. Os projetos se diferenciam, em relação às regiões do país onde os projetos são desenvolvidos, sendo o *Smart Campus* da UNICAMP na região Sul e o projeto SIMA, na região Norte, mais especificamente na região amazônica. Destacando-se que o SIMA, é inovador justamente por conta de ser o primeiro P&D de mobilidade elétrica multimodal desenvolvido nessa região e possibilitou a implantação de uma infraestrutura pioneira de mobilidade elétrica em vários pontos.

Destaca-se ainda que o projeto SIMA, colocou em circulação o primeiro ônibus elétrico em circulação nessa região, está criando o primeiro barco elétrico (catamarã) do Brasil, implantou o primeiro “Corredor verde” da região amazônica, criou uma

Rede de Inovação do Setor Elétrico (RISE) e irá gerar um **modelo de negócio** sobre mobilidade elétrica exclusivamente para essa região.

O segundo trabalho, da Dojot aplicada a mobilidade elétrica, é de o Moraes *et al.* (2021), originado desta dissertação. Neste trabalho foi analisada apenas a performance de um *cluster Kubernetes* com dois *workers* e uma *master*, contendo a plataforma Dojot, conforme ilustra a Figura 22.

Figura 22. Cluster *Kubernetes* com a Dojot.



Fonte: Moraes *et al.* (2021).

A versão final desta dissertação, realizou alguns pontos de melhorias, em relação a versão apresentada no trabalho de Moraes *et al.* (2021). Dentre elas, pode-se citar que a técnica de virtualização utilizada. Em Moraes *et al.* (2021), foi utilizada a virtualização do tipo 2, por conta disso havia mais uma camada de *software* na arquitetura, o *VirtualBox*, o que tornava o sistema mais pesado, em alguns momentos as VM dos ambientes de desenvolvimento e de produção desligavam sozinha e ficavam sem conexão à internet em alguns momentos, o que levava a Dojot a ficava *offline*. A conexão só era restabelecida após reiniciar as máquinas manualmente.

Outra melhoria realizada, foi em relação a configuração de rede utilizada. Em Moraes *et al.* (2021) era utilizada a conexão do tipo *Nat Networking*, por conta disso

era preciso fazer *Port forwarding*, ou seja, encaminhamento de portas para que as máquinas conseguissem se comunicar. Na implementação final, apresentada nesta dissertação, é utilizada a conexão do modo *Bridge*, onde cada máquina possui um IP próprio, todas as máquinas conseguem se comunicar entre si, como também suportam conexões para comunicação das VM para o *host*, do *host* para as VM, da internet para as VM e das VM para a internet. Conexões primordiais no processo de implantação e operação da Dojot, onde todas as máquinas precisam se comunicar.

3.3 Considerações Finais do Capítulo

Mediante ao exposto percebe-se que a Dojot pode ser utilizada em diversos tipos de aplicações. Ainda há poucos trabalhos encontrados na literatura brasileira que utilizem a plataforma Dojot, principalmente dela sendo utilizada em aplicações de mobilidade elétrica. Além disso, destaca-se que a maioria dos trabalhos encontrados relatam a implantação do ambiente de desenvolvimento da plataforma Dojot em uma máquina com *Docker Compose*. Um dos motivos para que isso ocorra, deve-se ao fato da curva de aprendizado da tecnologia *Kubernetes* ser maior e porque é preciso mais recursos de computação para a sua implementação.

4 IMPLANTAÇÃO DE UM MIDDLEWARE IOT ESCALÁVEL PARA APLICAÇÕES DE MOBILIDADE ELÉTRICA MULTIMODAL

Neste Capítulo 4, inicialmente, será apresentada uma síntese do P&D de mobilidade elétrica, do qual a Dojot é o *middleware IoT*. Posteriormente, será apresentada a arquitetura do sistema implementado, a metodologia utilizada para realizar a implantação da Dojot de forma escalável, e as especificações técnicas do servidor utilizado na implantação deste trabalho.

Em seguida, serão descritos os procedimentos realizados na instalação customizada da plataforma Dojot no ambiente de desenvolvimento e de produção de forma escalável. Será abordada aplicação da Dojot ao projeto SIMA de mobilidade elétrica multimodal e por fim sobre o *IoT Agent* (simulador *Python*) criado pela equipe pela equipe do projeto e exclusivamente para o Projeto SIMA.

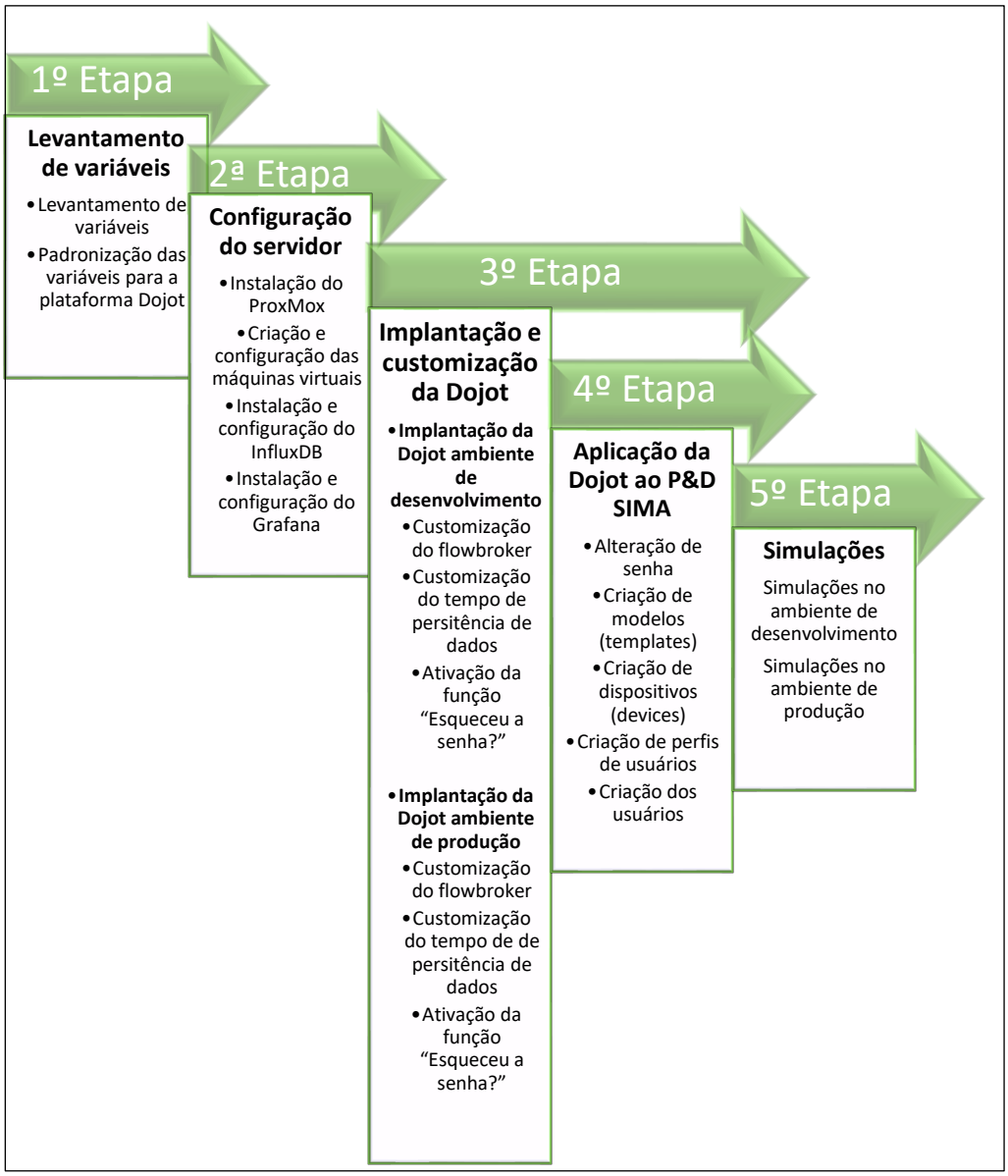
4.1 Metodologia de implantação

A metodologia realizada no desenvolvimento deste trabalho para a implantação da plataforma Dojot, foi realizada através de etapas, sendo que algumas etapas foram realizadas paralelamente, conforme pode ser visualizado na Figura 23. Sucintamente, foram realizados os seguintes procedimentos em cada uma dessas etapas:

1. **Levantamento de variáveis**: nessa primeira etapa, foi realizado o levantamento de variáveis das aplicações de mobilidade elétrica e a padronização para que posteriormente, na etapa 4, fossem inseridas na plataforma Dojot. O Apêndice A ilustra a lista de variáveis.

2. **Configuração do servidor**: foi realizada a instalação do Proxmox no servidor, a criação e configuração das Máquinas Virtuais (VM, do inglês *Virtual Machines*) para recebimento das máquinas da Dojot.
3. **Implantação da Dojot**: foi feita a implantação da Dojot nos ambientes de desenvolvimento e de produção. Além disso, também foram realizadas as seguintes customizações, a nível de código, em ambos os ambientes: atualização do *flowbroker*, aumento do tempo de persistência dos dados de telemetria e ativação da função “Esqueceu a senha?”.
4. **Aplicação da Dojot ao P&D SIMA:** nesta etapa, através da GUI da Dojot, foi alterada a senha padrão da Dojot e foram criados os modelos, dispositivos, fluxos, perfis e usuário.
5. **Simulações:** por fim, foram realizadas as simulações nos ambientes de desenvolvimento e produção, através de um *IoT Agent*.

Figura 23. Etapas da implantação do *middleware* IoT.



Fonte: produzido pela Autora.

4.2 Configurações do servidor

A seguir, serão descritas as especificações técnicas a nível de *hardware* e *software* necessárias para a implantação do sistema desta dissertação.

4.2.1 Especificações técnicas de *hardware*

O servidor onde o ambiente de virtualização foi criado e a plataforma Dojot foi implantada, está localizado no Centro de Excelência em Eficiência Energética da Amazônia (CEAMAZON) e pertence ao projeto SIMA. Este servidor está ligado à

internet em uma rede cabeada local do CEAMAZON, que por sua vez, está ligada a rede UFPA e é gerenciada pelo Centro de Tecnologia da Informação e Comunicação (CTIC). A Tabela 1 apresenta as especificações técnicas desse servidor.

Tabela 1 – Especificações técnicas do servidor.

Modelo	Power Edge T440
Processador	Intel Xeon Silver 4210
Núcleos	40
Memória	2 X 32 GB = 64 GB / 32 MHz/ RDIMM de 2666 MT/s
Controladora RAID	Controlador RAID PERC H330
Armazenamento (HD)	5 x 1TB = 5TB / 7.2K RPM SATA 6Gbps 512n 3.5 in Hot-plug Hard Drive
Placa de rede	NetXtreme BCM5720 2 portas Ethernet de 1 Gbit/s
Potência	450 W

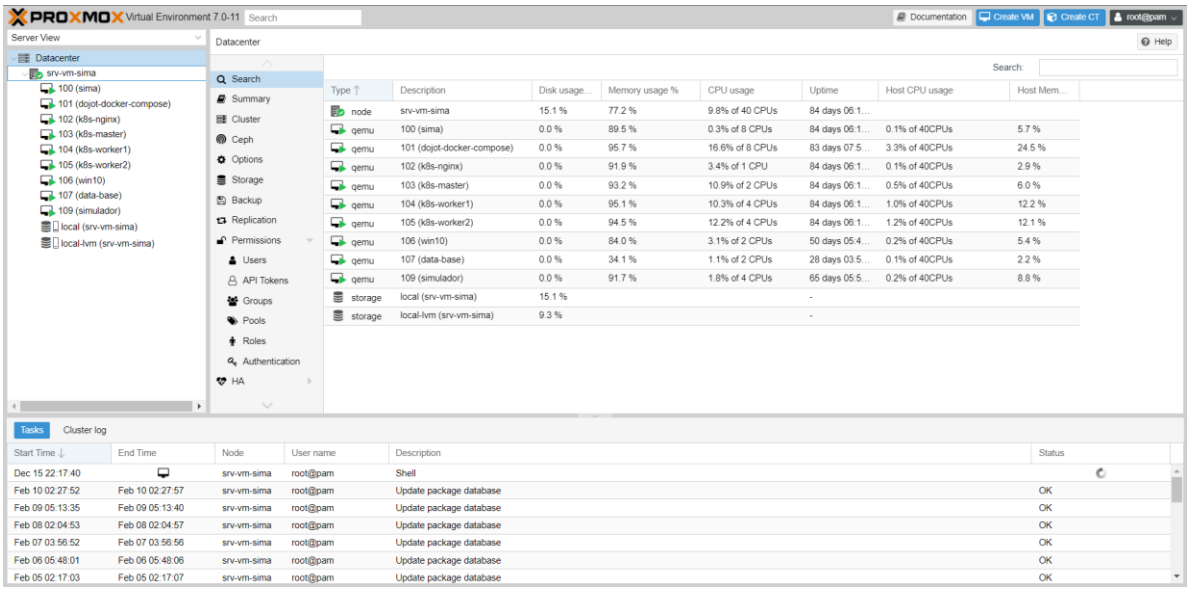
Fonte: Elaborado pela Autora.

Vale ressaltar que, além da Dojot, esse servidor também está sendo utilizado para alocadas outras aplicações de mobilidade elétrica multimodal do projeto SIMA. Por isso, optou-se em criar um ambiente de virtualização que será descrito na próxima subseção.

4.2.2 Especificações técnicas de *software* (Ambiente de virtualização)

Para poder melhor distribuir os seus recursos de *hardware* do servidor e utilizá-los de forma mais eficiente, optou-se por instalar diretamente no servidor o *software* de virtualização do tipo 1 Proxmox, versão 7.0. Após instalado o Proxmox passou a ser o equivalente ao sistema operacional do servidor através do sistema *Linux Debian* 5.11.22-4. Dentro do Proxmox foram criadas as VM para a Dojot, o *IoT Agent* (simulador *Python*) e as aplicações do projeto SIMA, conforme pode ser visualizada a Figura 24*.*

Figura 24. Interface do Proxmox no servidor do projeto SIMA.



Fonte: produzido pela Autora.

Ressalta-se que esse ambiente de virtualização suporta a escalabilidade vertical e horizontal. Ou seja, é possível adicionar mais recurso em uma única máquina ou contêineres, como também criar máquinas, ou contêineres, ou até mesmo, agrupar vários servidores (*Datacenters*) que também estejam com o Proxmox instalado.

A descrição de como instalar o Proxmox e como criar uma VM no Proxmox está disponível na sua documentação oficial (PROXMOX, 2021). Após a instalação do Proxmox é possível o seu terminal de comando, através do próprio servidor, ou acessar a sua GUI web, através de um navegador de internet de outro computador que esteja conectado na mesma rede, através IP e porta: http://IP_DO_SERVIDOR:8006, onde, “IP_DO_SERVIDOR” deve ser substituído pelo número IP do servidor, que neste caso não foi divulgado por questões de segurança.

4.3 Descrição do sistema implantado

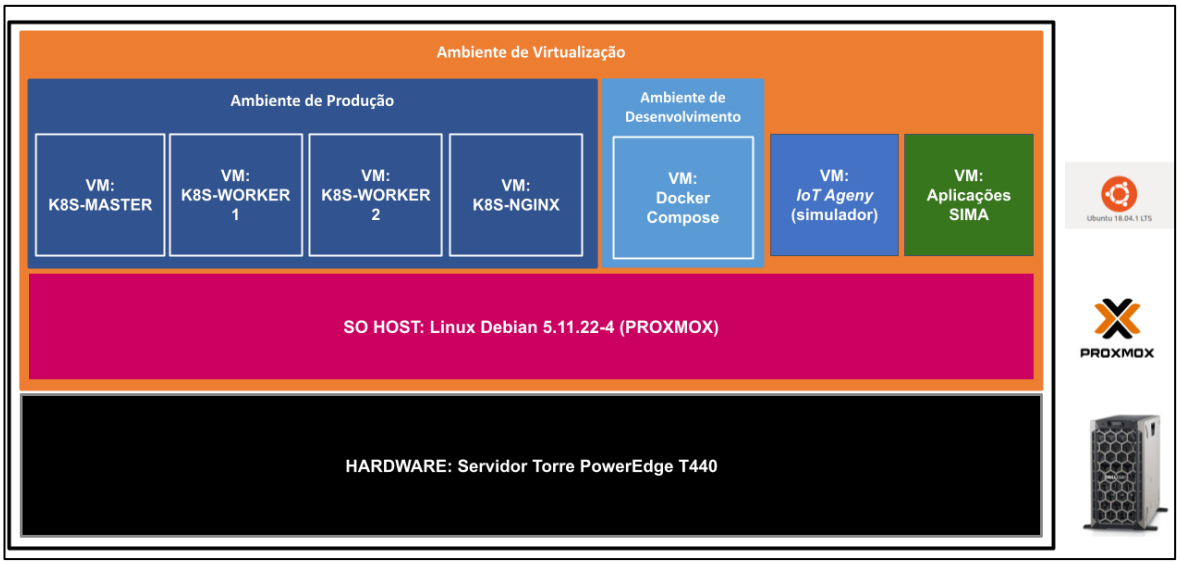
Foram implantados dois ambientes com a plataforma Dojot, sendo um ambiente de desenvolvimento para realização de testes e um ambiente de produção para ser usado pelo sistema final do projeto SIMA. Além desses dois ambientes também foi criado um ambiente com o um *IoT Agent* e outro para receber as aplicações do projeto SIMA, conforme será visto nas próximas subseções.

4.3.1 Arquitetura do sistema

A documentação da própria Dojot recomenda que ela seja implantada em máquinas físicas ou VMS e que seja utilizado o sistema operacional Ubuntu 18.04 (CPQD, 2021). Por conta da limitação em relação a quantidade de máquinas físicas disponíveis e da existência do servidor supracitado com bastante recursos de CPU, memória e armazenamento (Tabela 1), optou-se pela sua implantação em VM. Mediante a isso, foram criadas sete VM, conforme ilustra a Figura 25 , sendo:

- 1 máquina para alocar a Dojot com *Docker-Compose*;
- 4 máquinas para alocar a Dojot no *cluster Kubernetes*;
- 1 máquina para alocar o *IoT Agent* (simulador de dados em Python);
- 1 máquina para locar as aplicações do projeto SIMA.

Figura 25. Arquitetura do Servidor.



Fonte: produzido pela Autora.

A Tabela 2 apresenta a quantidade de recursos alocados em cada uma dessas máquinas. Destaca-se que a máquina *Docker-compose* tem o equivalente ao somatório dos recursos das duas máquinas *workers* do *cluster Kubernetes* (*k8s-worker1 e k8s-worker2*). Isso porque é nessas máquinas que de fato são executados todos os serviços da Dojot.

Destaca-se ainda que, foram alocados 700GB de armazenamento (HD) em cada uma das máquinas *workers*, por conta do tempo de persistência dos dados no ambiente de produção, que por norma precisa ser de no mínimo 30 dias, conforme será descrito na subseção 4.3.4.

Tabela 2 – Recursos das máquinas virtuais.

Máquina	CPU (cores)	RAM (GB)	HD (GB)	Sistema Operacional
docker-compose	8	16	40	Ubuntu Desktop 18.04 LTS
k8s-nginx	1	2	32	Ubuntu Desktop 18.04 LTS
k8s-master	2	4	64	Ubuntu Desktop 18.04 LTS
k8s-worker1	4	8	700	Ubuntu Desktop 18.04 LTS
k8s-worker2	4	8	700	Ubuntu Desktop 18.04 LTS

Fonte: Elaborados pela Autora.

De acordo com os cálculos realizados, em um cenário real de produção, essa quantidade de armazenamento de 700GB em cada *worker* será suficiente para alocar dados de telemetria por até 100 anos no servidor, já considerando uma margem de segurança de 20%, conforme mostra a Tabela 3. Para realizar esse cálculo, foi considerando o caso extremo de 50 medidores (*endnodes*), enviando dado para a Dojot a cada 1 minutos, onde cada medidor pode enviar até 225 *bytes*, ou seja, a cada 1 minuto são enviados o total de 11.250 *bytes* para a Dojot.

Tabela 3 – Cálculo da quantidade de recurso de armazenamento (HD) das máquinas *k8s-worker1 e k8s-worker2*.

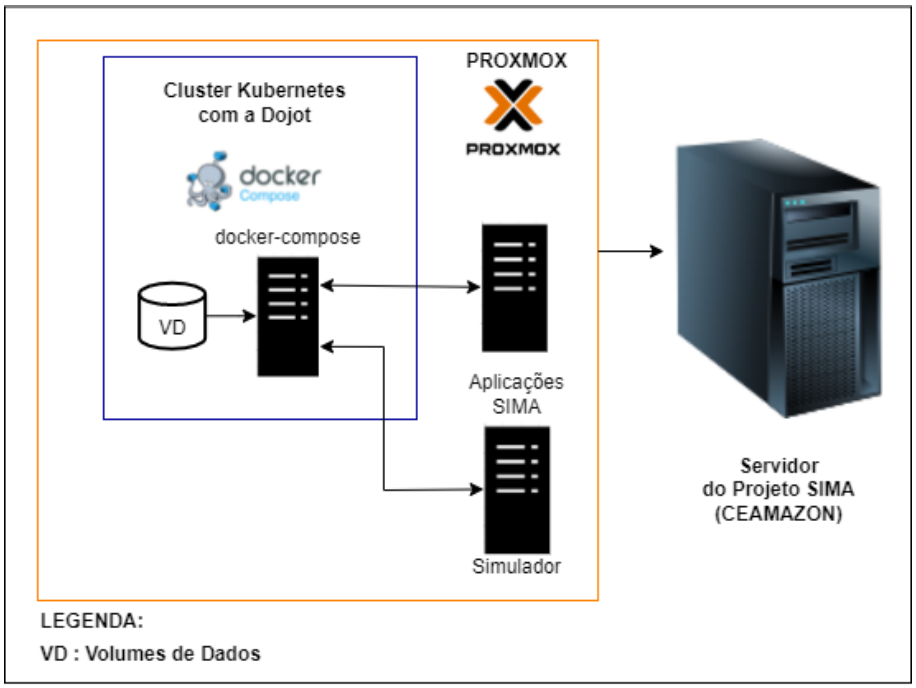
TEMPO	TEMPO EM SEGUNDOS	BYTES	MEGABYTES	GIGABYTES	MARGEM DE SEGURANÇA (20%)
1 MINUTO	60	11250	0,01125	0,00001125	0,0000135
1 HORA	3600	675000	0,675	0,000675	0,00081
1 DIA	86400	16200000	16,2	0,0162	0,01944
1 MÊS	2592000	486000000	486	0,486	0,5832
1 ANO	31104000	5832000000	5832	5,832	6,9984
10 ANOS	311040000	58320000000	58320	58,32	69,984
15 ANOS	4665600000	87480000000	87480	87,48	104,976
20 ANOS	622080000	116640000000	116640	116,64	139,968
25 ANOS	7776000000	145800000000	145800	145,8	174,96
30 ANOS	933120000	174960000000	174960	174,96	209,952
40 ANOS	1244160000	233280000000	233280	233,28	279,936
50 ANOS	1555200000	291600000000	291600	291,6	349,92
100 ANOS	3110400000	583200000000	583200	583,2	699,84

Fonte: Elaborados pela Autora.

4.3.2 Ambiente de desenvolvimento: *Docker-compose*

O ambiente de desenvolvimento foi implantado em uma única máquina com *Docker-Compose*, conforme ilustra Figura 26. Os recursos alocados nessa máquina são os mesmos descritos anteriormente na Tabela 2. Esta máquina possui a função de executar os serviços da Dojot. Como os procedimentos de instalação da Dojot no ambiente de desenvolvimento estão descritos na documentação oficial da própria Dojot, eles não serão descritos aqui.

Figura 26. Arquitetura do ambiente de desenvolvimento



Fonte: produzido pela Autora.

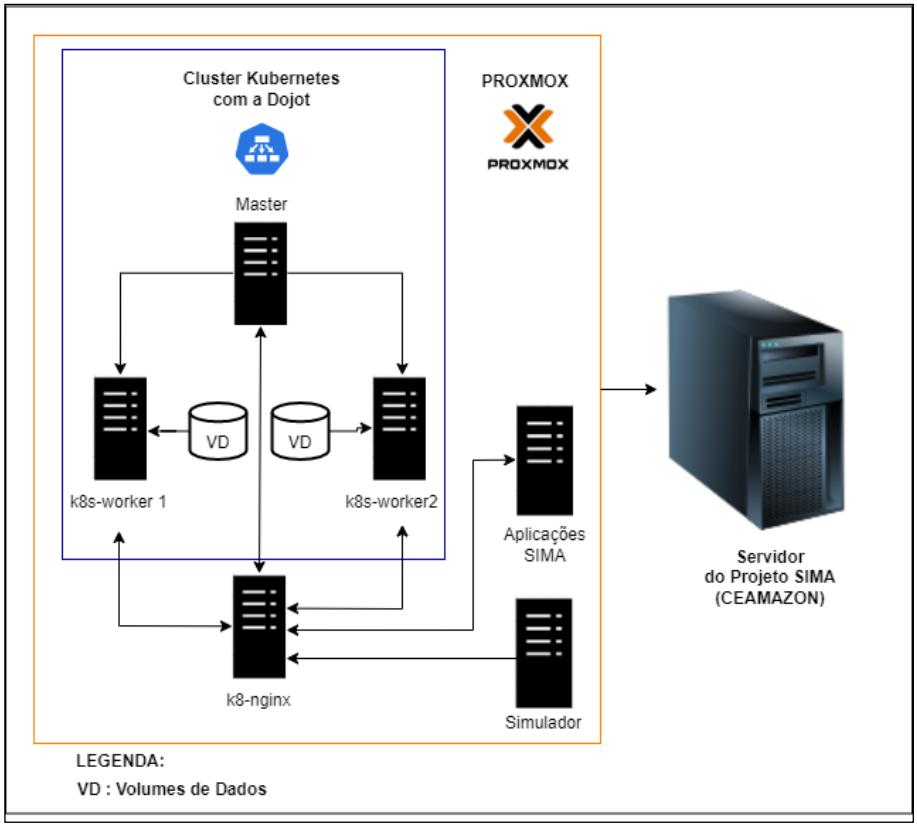
4.3.3 Ambiente de produção: *cluster Kubernetes*

O ambiente de produção foi implantado em um *cluster Kubernetes* criado em VM. Para isso foram criados os seguintes “nós” (máquinas): um (1) nó master e dois (2) nós *workers*, sendo eles *k8s-worker 1* e *k8s-worker 2*. Além desses nós há também um (1) nó balanceador de carga NGNIX. A Figura 27 apresenta a arquitetura do *cluster Kubernetes* criado e os recursos alocados em cada nó (máquina) são os mesmos descritos anteriormente na Tabela 2. Cada nó (máquina) possui a seguinte função:

- **k8s-master**: administra e gerencia o cluster *Kubernetes*;

- **k8s-worker 1** e **k8s-worker 2**: executa os serviços da Dojot. Vale ressaltar que para um ambiente onde há uma carga grande de dispositivos é recomendado a utilização de no mínimo 2 nós *workers* no *cluster Kubernetes*, semelhante ao cluster criado nesta dissertação.
- **k8s-nginx**: recebe todas as requisições TCP, UDP, MQTT e ele realiza a distribuição da carga entre os nós *workers* do *cluster Kubernetes*.

Figura 27. Arquitetura do *cluster Kubernetes*.



Fonte: produzido pela Autora.

Como os procedimentos de instalação deste *cluster Kubernetes* estão descritos na documentação oficial da própria Dojot, eles não serão descritos aqui. Nesta documentação também estão descritos os procedimentos de instalação do *Ansible* e do *NGINX* (balanceador de carga).

4.3.4 Implantação da Dojot de forma customizada

A versão da Dojot implantada neste trabalho foi a 0.7.0. O guia de instalação oficial da Dojot, disponível em CPqD (2021), descreve os procedimentos tradicionais que devem ser realizados para implantá-la no ambiente de desenvolvimento (com

D*ocker-compose*) e no ambiente de produção (no *cluster Kubernetes*). Por isso, esses procedimentos não serão descritos aqui.

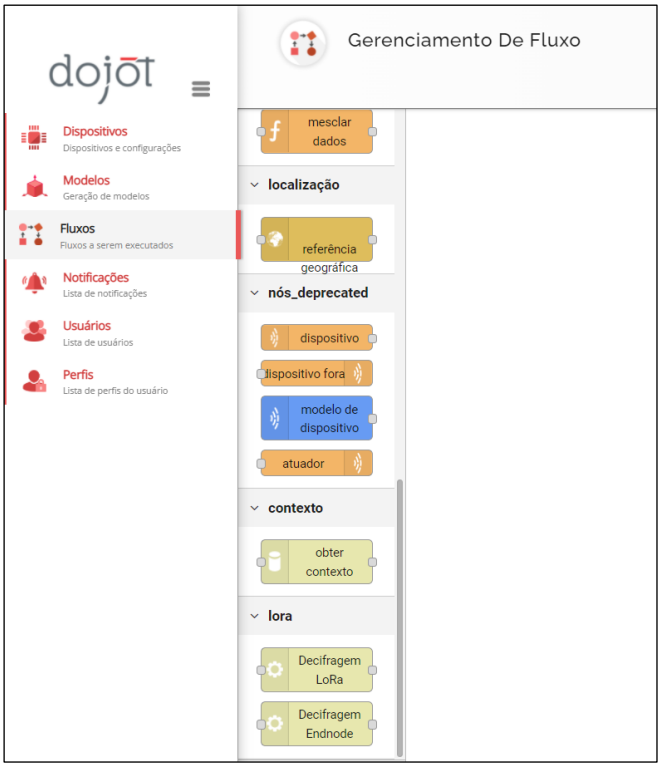
Contudo, destaca-se que foram realizados alguns procedimentos adicionais que não estão descritos no guia de instalação oficial da Dojot. Esses procedimentos adicionais foram realizados a nível de código, a fim de customizar a plataforma Dojot para atender as especificidades das aplicações de mobilidade elétrica multimodal do Projeto SIMA. Por conta do sigilo do projeto, que ainda está em execução atualmente, tais customizações serão apenas listadas e brevemente descritas a seguir.

As seguintes customizações foram realizadas nos dos ambientes (desenvolvimento e produção) da Dojot para atender as especificidades das aplicações de mobilidade elétrica multimodal do Projeto SIMA:

1. **Customização do *flowbroker***

Como uma parte dos dados das aplicações do projeto serão enviados através da tecnologia LoRa, foi preciso customizar o componente *flowbroker* da Dojot para receber esses dados. Mediante a isso, foram inseridos os blocos “Decifragem *LoRa*” e “Decifragem *EndNode*” no *flowbroker* (fluxo) conforme ilustra a Figura 28. Destaca-se que essa customização precisa ser realizada no ambiente de desenvolvimento e no ambiente de produção.

Figura 28. *Flowbroker* customizado.



Fonte: produzido pela Autora.

2. Customização do tempo de persistência dos dados de telemetria

Por padrão a Dojot persiste os dados de telemetria por até sete dias em seu banco de dados *MongoDB*. Como alguma aplicações do projeto precisam de um tempo de persistência de no mínimo 30 dias, de acordo com o Módulo 8 do Procedimentos de Distribuição de Energia Elétrica (PRODIST) no Sistema Elétrico Nacional (ANEEL, 2021), esse tempo foi aumentado para 100 anos. Destaca-se que essa customização precisa ser realizada no ambiente de desenvolvimento e de produção.

3. Fixação local de execução dos serviços da Dojot

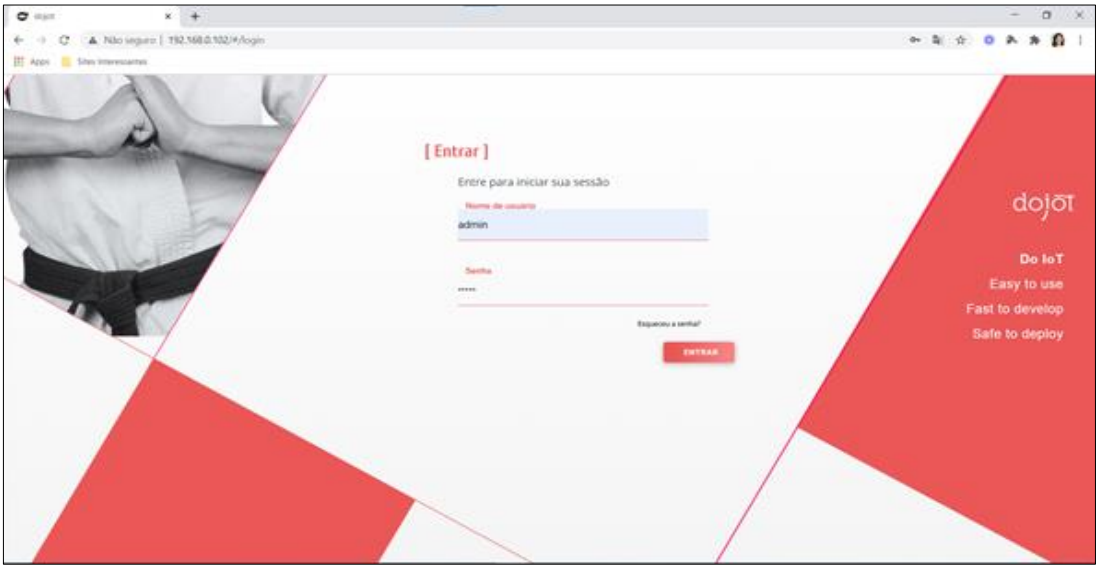
Essa customização precisa ser realizada apenas no ambiente de produção (cluster *Kubernetes*) e foi realizada porque o cluster do projeto possui duas máquinas *workers*. Mediante a isso, foi preciso fixar onde os serviços da Dojot, do *Kafka* e do x509 são executados, juntamente com os seus respectivos volumes de dados. Vale ressaltar que esse procedimento precisa ser realizado apenas até a versão 0.7.0 da Dojot, pois de acordo com a equipe do CPqD nas próximas versões, esse procedimento não será mais necessário.

4. Ativação da função “Esqueceu a senha?”

Por padrão a função “Esqueceu a senha?” vem desativada na Dojot e para ser utilizada precisa ser ativada. Ao ativar essa função, também é ativada a função de envio de e-mail de configuração de senha para o e-mail de todo novo usuário cadastrado na Dojot. Para isso, o seguinte procedimento precisa ser realizado, na máquina *docker-compose*, no ambiente de desenvolvimento:

Após a implantação da Dojot é possível acessar a sua interface gráfica através de um navegador, conforme ilustra a Figura 29. Para acessar o ambiente de desenvolvimento deve ser utilizando o endereço http://localhost:8000 e para acessar o ambiente de produção deve ser utilizado o endereço http://localhost:80. Vale ressaltar que “*localhost*” deve ser substituído pelo número IP da máquina em que a Dojot se encontra.

Figura 29. GUI da Dojot.



Fonte: produzido pela Autora

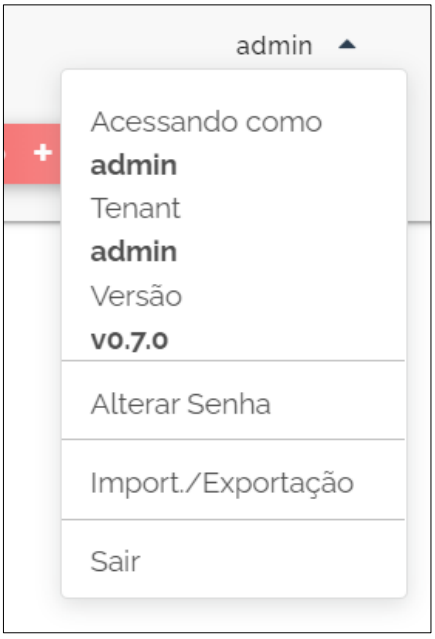
4.3.5 Aplicação da Dojot ao P&D SIMA

Após implantada de forma customizada a Dojot foi aplicada através da sua GUI, para atender as especificidades das aplicações de mobilidade elétrica multimodal do projeto SIMA. Os procedimentos de aplicação que serão descritos a seguir foram realizados, tanto no ambiente de desenvolvimento, quanto no ambiente de produção e podem ser realizados da mesma forma em ambos os ambientes.

4.3.5.1. Alteração de senha

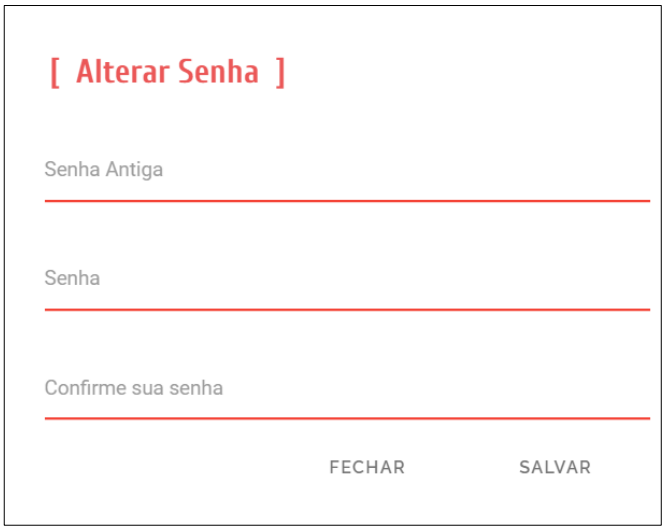
Por padrão a Dojot vem configurada com a senha “admin” para o usuário “admin”, então o primeiro procedimento que deve ser realizado é a alteração dessa senha através da GUI. Para isso, basta clicar na seta ao lado do nome usuário admin e clicar em alterar a senha, conforme ilustra a Figura 30. Posteriormente, na nova janela que será aberta, basta inserir a senha atual, a nova senha desejada e clicar em salvar, conforme ilustra a Figura 31.

Figura 30. Opção alterar senha da Dojot.



Fonte: produzido pela Autora.

Figura 31. Configurando nova senha da Dojot.



Fonte: produzido pela Autora.

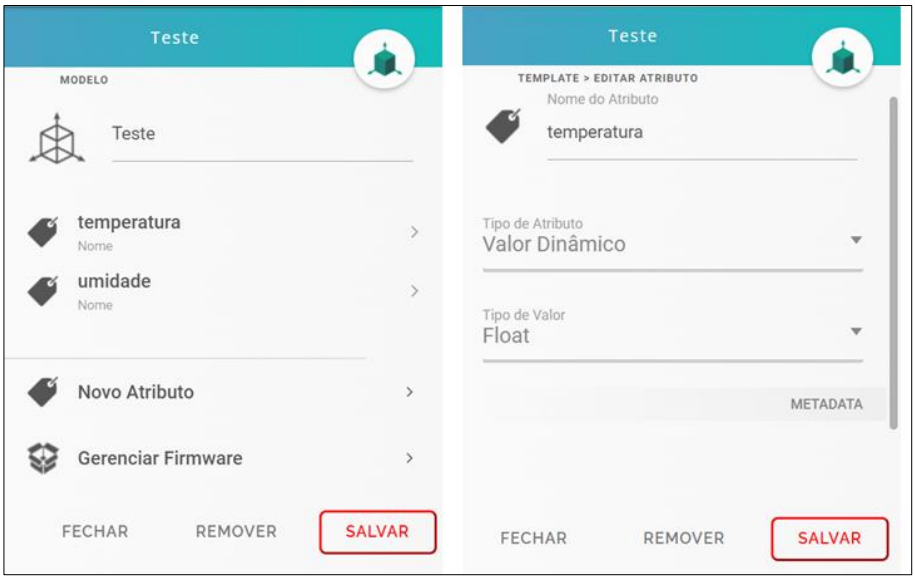
4.3.5.2. Modelos (*Templates*)

Um dos primeiros passos ao acessar a plataforma Dojot, é a criação de modelos de dispositivos (*templates*). Os modelos contêm uma lista de atributos (que neste trabalho também podem ser chamados de variáveis) que são herdados pelos dispositivos a eles associados (CPQD, 2021). Vale ressaltar que qualquer alteração que seja feita em um determinado modelo é repassada automaticamente para dispositivos a ele associado. O componente responsável pelo gerenciamento de modelos é o *Device Manager*. Ao criar um modelo é possível definir:

- **Um nome**: que pode ser definido pelo próprio usuário;
- **Um tipo de atributo**: que pode ser “valor dinâmico”, “valor estático” ou “atuador”;
- **Um tipo de valor**: que pode ser “Boleano”, “Geo”, “*Float*”, “Inteiro” e “Texto”;
- **Um valor**: caso o atributo de do tipo “valor estático”;

Na Figura 32 é possível visualizar um modelo nomeado de “Teste”, com dois atributos “temperatura” e “humidade” e o atributo “temperatura” com o tipo “valor dinâmico” e o tipo do valor “*float*”.

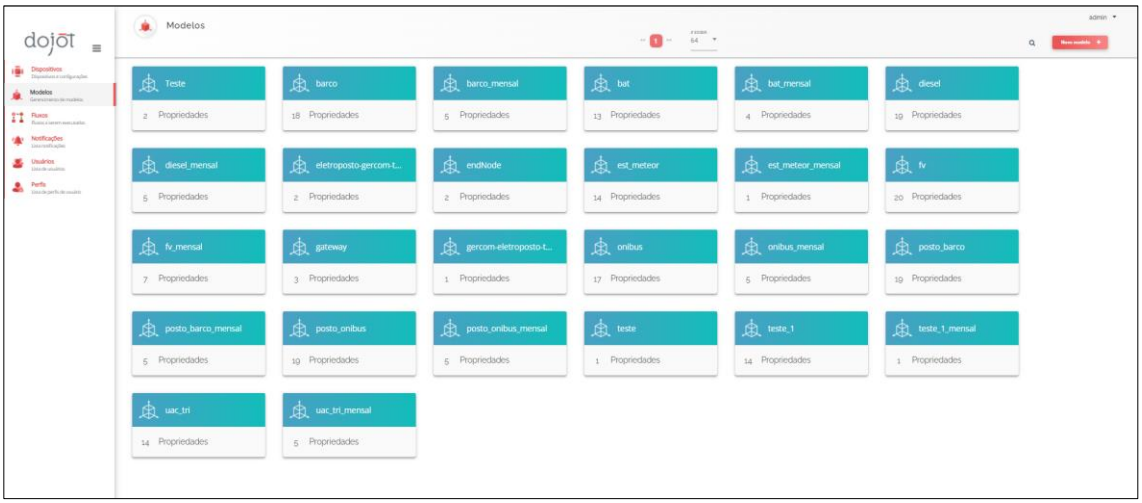
Figura 32. Modelo de teste na Dojot.



Fonte: produzido pela Autora.

Utilizando o *IoT Agent* e a GUI, foram criados 18 modelos (*templates*), com os seus respectivos atributos (variáveis), previstos para as aplicações de mobilidade elétricas do P&D SIMA que utilizam a Dojot. Na Figura 33 e no Apêndice é possível visualizar os modelos criados, juntamente com os seus respectivos atributos.

Figura 33. Modelos das aplicações de mobilidade elétrica na Dojot.



Fonte: produzido pela Autora.

4.3.5.3. Dispositivos (*Devices*)

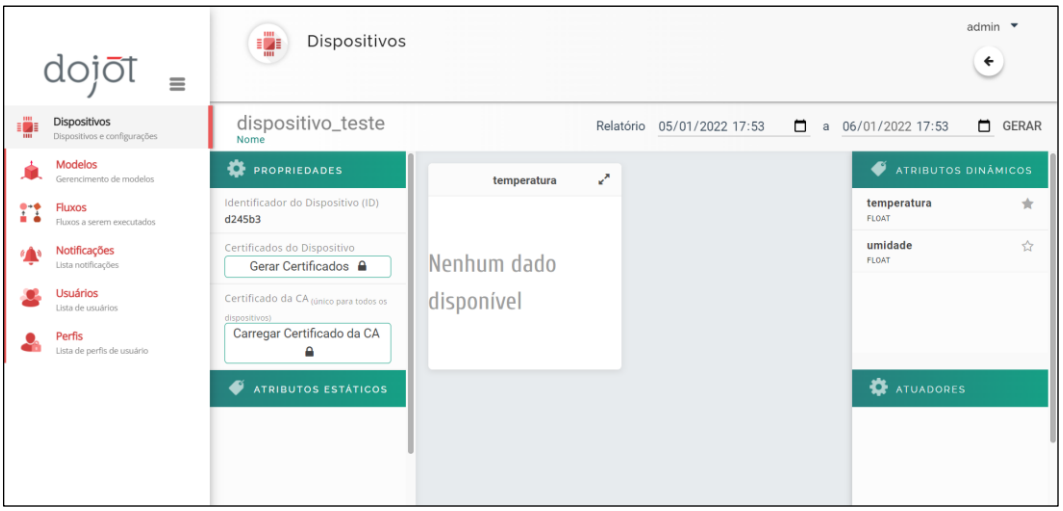
Na Dojot um dispositivo é uma representação digital de um dispositivo ou gateway real com um ou mais sensores, ou também pode ser a representação digital de um dispositivo virtual com sensores/atributos inferidos de outros dispositivos

(CPQD, 2021). Todos os dispositivos são criados com base em um ou mais modelos (*templates)* criados previamente antes dos dispositivos. O componente responsável pelo gerenciamento de dispositivos (reais e virtuais) é o *Device Manager.* Ao criar um dispositivo é possível:

- Definir um nome;
- Adicionar ou remover um ou mais modelos; e
- Gerenciar os seus atributos.

Na Figura 34 é possível visualizar um dispositivo nomeado de “dispositivo_teste”, que foi vinculado ao modelo (*template*) “Teste” criado anteriormente, por isso os seus atributos são “temperatura” e “umidade”. Vale ressaltar, que todo dispositivo criado recebe automaticamente um identificado´(ID) único, que neste caso é “d245b3”.

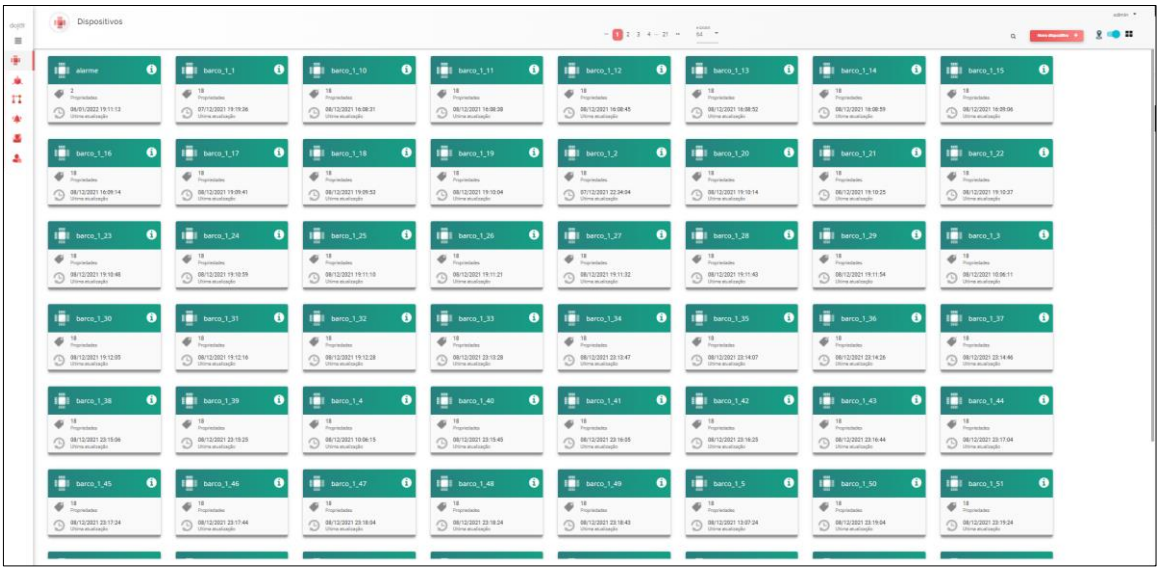
Figura 34. Dispositivo de teste na Dojot.



Fonte: produzido pela Autora.

Utilizando o *IoT Agent* e a GUI, foram criados 640 dispositivos, com os seus respectivos atributos (variáveis), previstos para as aplicações de mobilidade elétricas do P&D SIMA que utilizam a Dojot. Na Figura 35 é possível visualizar uma parte dos dispositivos criados.

Figura 35. Dispositivos das aplicações de mobilidade elétrica na Dojot.



Fonte: produzido pela Autora.

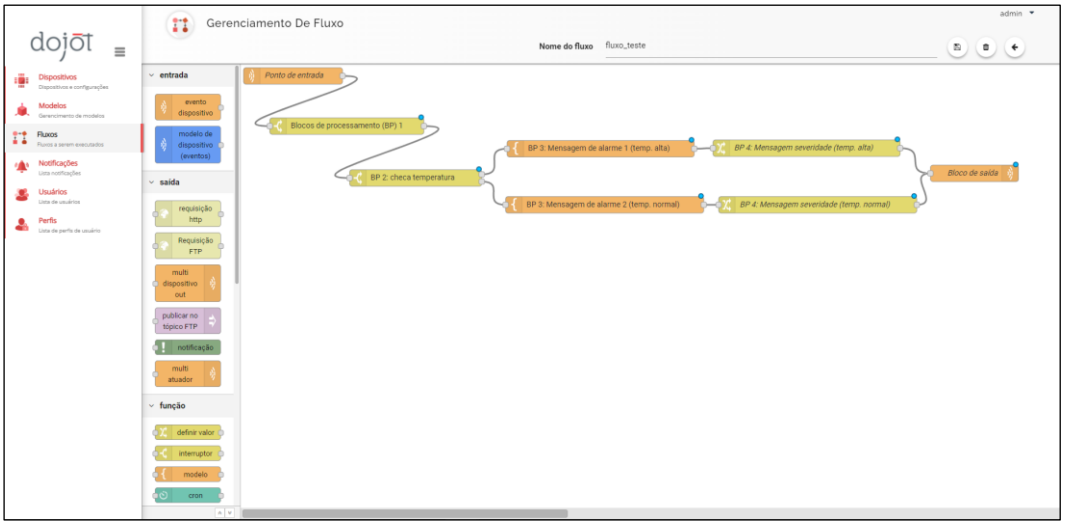
4.3.5.4. Fluxos

O fluxo é uma sequência de blocos que processa um evento ou mensagem de um dispositivo específico (CPQD, 2021). O componente responsável por lidar com esses fluxos é o *Flowbroker*. Um fluxo, deve conter:

- **Ponto de entrada**: um bloco representando qual é o gatilho para iniciar um fluxo específico;
- **Blocos de processamento**: um conjunto de blocos que executam operações usando um determinado evento acontece. Esses blocos podem ou não usar o conteúdo desse evento para processá-lo ainda mais. As operações podem ser: testar conteúdo para valores ou intervalos específicos, análise de posicionamento geográfico, alterar atributos de mensagens, executar operações em elementos externos e entre outras;
- **Ponto de saída**: é um bloco que representa para onde os dados resultantes devem ser encaminhados. Esse bloco pode ser um banco de dados, um dispositivo virtual, um elemento externo e assim por diante.

Na Figura 36 é possível visualizar um fluxo nomeado de “fluxo_teste”, que contém todos os blocos supracitados.

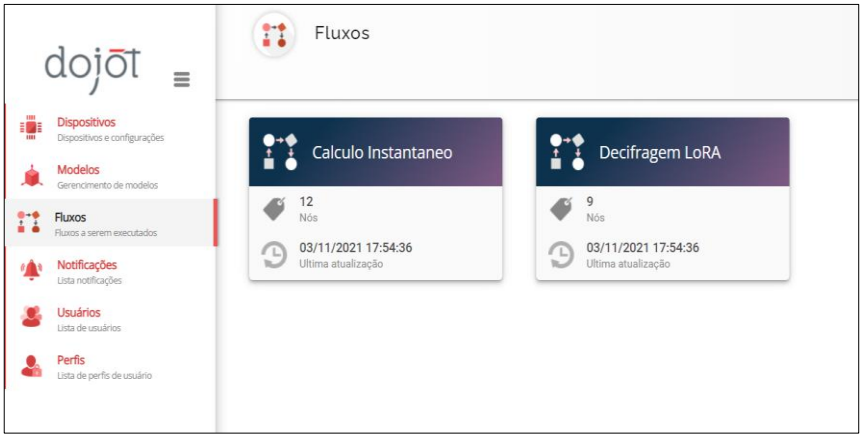
Figura 36. Fluxo de teste na Dojot.



Fonte: produzido pela Autora.

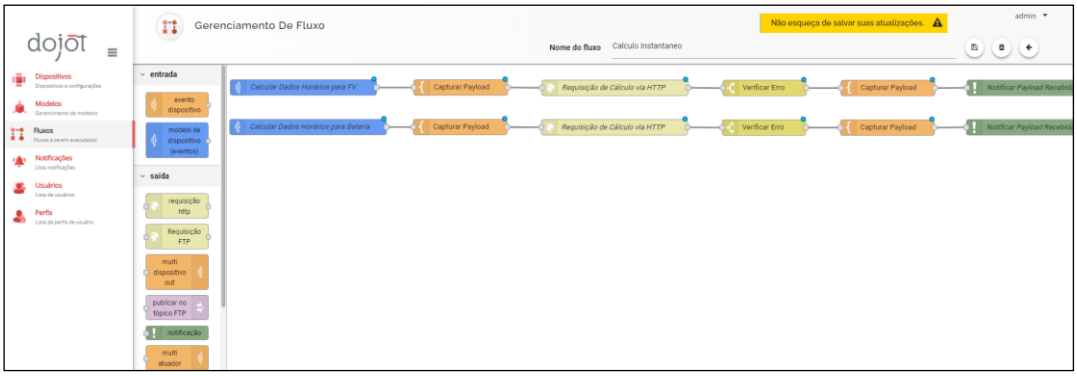
Através da GUI da Dojot, foram criados dois Fluxos na plataforma Dojot, são eles: Cálculo Instantâneo e Decifragem LoRa, conforme ilustra a Figura 37. Adentrando-se em cada um desses fluxo, na Figura 38 é possível visualizar o fluxo “Cálculo Instantâneo”, composto por dois fluxos, sendo um para o sistema fotovoltaico e outro para o sistema de baterias. Em ambos os fluxo o caminho percorrido é basicamente o mesmo: o ponto de entrada um evento para calcular os dados horários de todos os dispositivos que pertencem aos sistemas fotovoltaicos e ao banco de baterias; os blocos de processamento são para capturar os *payloads*, fazer requisições HTTP, verificar erros nos *payloads*; e como ponto de saída é gerada uma notificação que o *payload* foi recebido.

Figura 37. Fluxo de Cálculo Instantâneo criado na Dojot.



Fonte: produzido pela Autora.

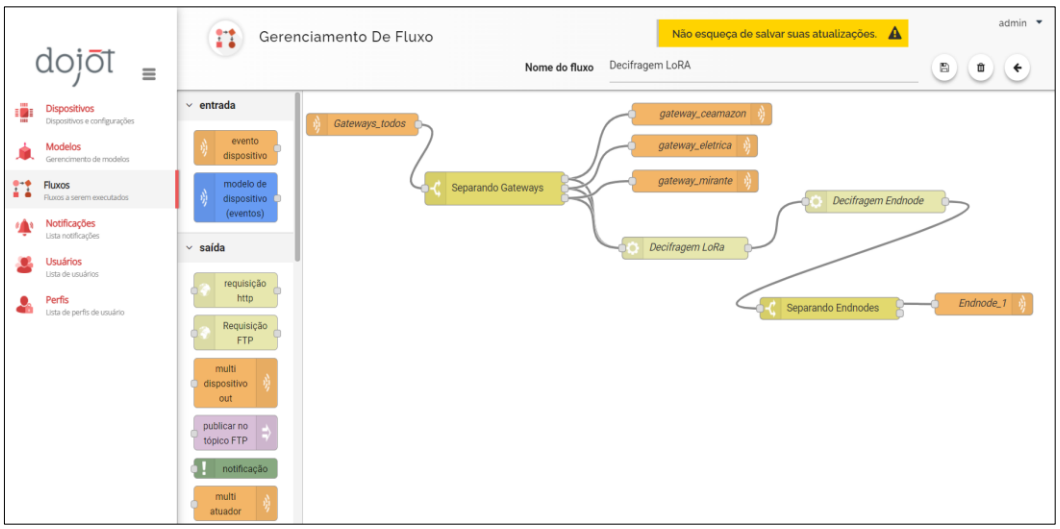
Figura 38. Blocos do fluxo de Cálculo Instantâneo.



Fonte: produzido pela Autora.

Na Figura 39 é possível verificar os blocos que compõem o fluxo de “Decifragem Lora”, onde o ponto de entrada é um bloco que contém um dispositivo que recebe os dados de todos os *gateways*, inicia-se então a fase de processamento, onde é feita a decifragem dos dados para o padrão Lora, posteriormente esses dados são decifrados para o padrão *endnode*. Destaca-se que o padrão *endnode* foi criado pela equipe do projeto, juntamente com o CPqD, e depende do tipo da aplicação que enviou dados. Por fim, como ponto de saída os dados decodificados são enviados para o(s) seu(s) respectivo(s) dispositivo(s).

Figura 39 Blocos do fluxo de Decifragem LoRa.



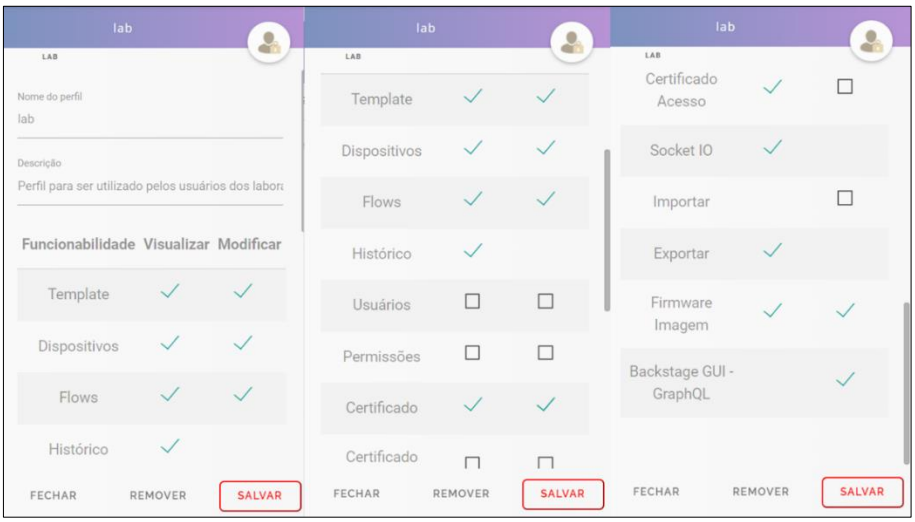
Fonte: produzido pela Autora.

4.3.5.5. Perfis

Através da GUI da Dojot, foram criados dois **Perfis** com diferentes níveis de permissões para o projeto, conforme ilustram as Figura 40 e Figura 41, são eles:

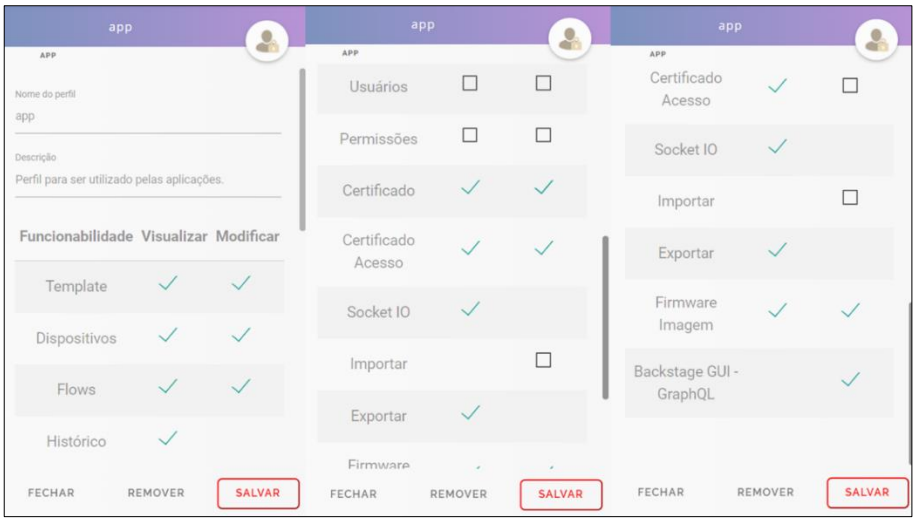
- **lab**: para ser utilizado pelos usuários dos laboratórios;
- **app**: para ser utilizado pelas aplicações.

Figura 40. Permissões do perfil lab.



Fonte: produzido pela Autora.

Figura 41. Permissões do perfil app.

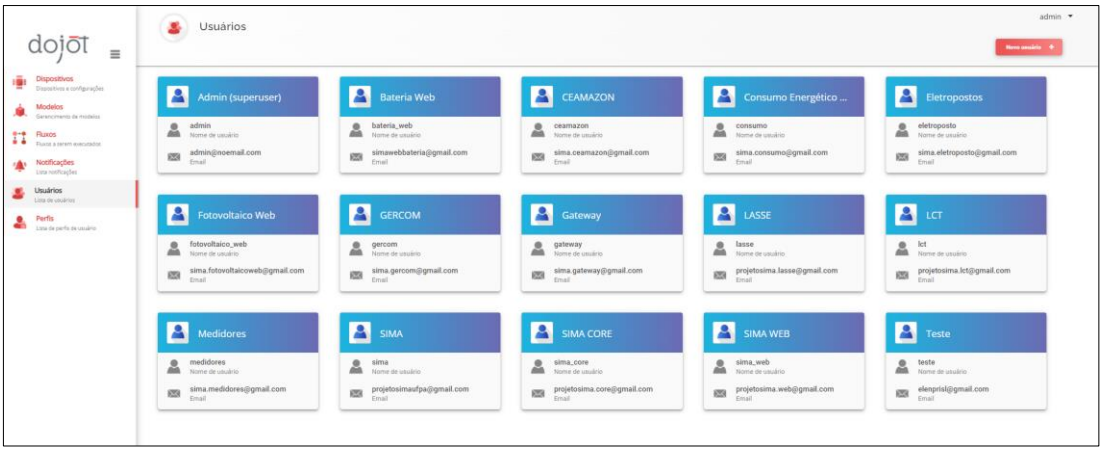


Fonte: produzido pela Autora.

4.3.5.6. Usuários

Com base nos perfis apresentados anteriormente, através da GUI da Dojot, foi criado 1 usuário para cada laboratório e 1 usuário para cada aplicação prevista até agora para o projeto. A Figura 42 e no Quadro 3 estão os usuários criados, até o presente momento.

Figura 42. Usuários do projeto.



Fonte: produzido pela Autora.

Quadro 3 – Usuários da Dojot.

Nome do Usuário	Nome	Perfil
admin	Admin	admin
ceamazon	CEAMAZON	lab
gercom	GERCOM	lab
lct	LCT	lab
lasse	LASSE	lab
eletroposto	Eletropostos	app
gateway	Gateway	app
baterias_web	Baterias	app
consumo	Consumo Energético dos Modais	app
fotovoltaico_web	Fotovoltaico Web	app
medidores	Medidores	lab
sima	SIMA	app
sima_web	SIMA WEB	app
sima_core	SIMA CORE	app

Fonte: produzido pela Autora.

4.4 *IoT Agent*: simulador Python

Para realizar as simulações foi criado um *IoT Agent*, ou seja, um simulador em linguagem *Python* capaz de criar automaticamente modelos, dispositivos e enviar dados randômicos, via protocolo MQTT para os dispositivos da Dojot. Os comandos utilizados pelo *IoT Agent* para realizar esses procedimentos são baseados nos comandos aceitos pelas API da própria Dojot.

Sucintamente, ao ser iniciado o *IoT Agent* se conecta a Dojot e verifica se já existem modelos e dispositivos necessários para a simulação, caso eles existam a simulação é iniciada, caso contrário todos os modelos e dispositivos são criados. Posteriormente é verificada qual a data final da simulação Na Figura 43 é possível visualizar o pseudocódigo do funcionamento do simulador.

Figura 43. Pseudocódigo do *IoT Agent*.

```
algoritmo “IoT Agente SIMA”

var modelos, dispositivos, cont, data_atual, media, primeira_data,
ultima_data, data_final, data_medicao, coeficiente_curva_de_carga

inicio

# O EndNode (simulador) se conecta a Dojot

# Verificar se os modelos e dispositivos já estão criados
Se modelos e dispositivos true
    inicia simulacao
Se nao
    cria modelos e dispositivos

# Receber argumentos da linha de comandos
Se cont
    data final <- data_atual
Se nao
    data_final <- ultima_data

# Ler dispositivos da Dojot

# Inicia Protocolo MQTT
data_inicial <- primeira_data

Enquanto data_medicao <= data_final
    Para todo “atributo” dos dispositivos da Dojot
        Se “atributo” = “media”
            valor_atributo <- Gaussiana “atributo”
            valor_atributo <- valor_atributo * coeficiente_curva_de_carga
            limitar (valor_atributo)
        Se nao se “atributo” = valor
            valor_atributo <- sorteio “atributo”
        Se nao
            valor_atributo <- sequencial “atributo”
Envia atributo para a Dojot

Se cont
    Enquanto true
    Para todo “atributo” dos dispositivos da Dojot
        Se “atributo” = “media”
            valor_atributo <- Gaussiana “atributo”
            limitar (valor_atributo)
        Se nao se “atributo” = valor
            valor_atributo <- sorteio “atributo”
        Se nao
            valor_atributo <- sequencial “atributo”
fim
```

Fonte: produzido pela Autora.

4.5 Considerações Finais do Capítulo

Neste Capítulo foi possível saber os recursos de *hardwares* e *softwares* necessários para realizar a implantação do sistema proposto nesta dissertação. Além disso, foi possível verificar os procedimentos e passos realizados para realizar a implantação da plataforma Dojot de forma escalável, customizada e aplicá-la ao P&D de mobilidade elétrica SIMA.

5 RESULTADOS E ANÁLISES

O principal resultado obtido com o desenvolvimento deste trabalho, foi a implantação dos dois ambientes com a plataforma Dojot, de forma escalável, descrita no Capítulo 4. Assim, neste Capítulo 5, serão apresentados os resultados das simulações realizadas nos dois ambientes onde a Dojot foi implantada para verificar o consumo de CPU e memória das máquinas que contém a Dojot.

5.1 Contextualização dos cenários simulados

As simulações foram realizadas primeiramente no ambiente de desenvolvimento da Dojot com *Docker Compose* e posteriormente no ambiente de produção da Dojot do cluster *Kubernetes*. Para isso, foram criados através do *IoT Agent* e da GUI, modelos e dispositivos que representavam o ônibus elétrico, o barco elétrico, o eletroposto do ônibus e o eletroposto do barco, o medidor trifásico, o banco de baterias, o gerador diesel, a estação meteorológica, o sistema fotovoltaico do CEAMAZON e o sistema fotovoltaico do Mirante do Rio. Os respectivos atributos (variáveis) de cada um desses dispositivos podem ser visualizados no Apêndice A desta dissertação.

Para o envio de dados, foi utilizado o *IoT Agent*, descrito na seção 4.4. Foram enviados dados simultaneamente para 10 dispositivos durante o período de uma hora, a cada 10 minutos. Esse quantitativo de dispositivos enviando dados foi incrementado exponencialmente e respectivamente a cada uma hora para 20, 40, 80, 160 e 320 dispositivos. Para incrementar esse quantitativo inicial de 10 dispositivos, até atingir a quantidade de dispositivos necessários para cada hora de simulação, foram criados, por exemplo: ônibus elétrico 1, ônibus elétrico 2, ... , ônibus elétrico *N*; medidor trifásico 1, medidor trifásico 2, medidor trifásico *N*.

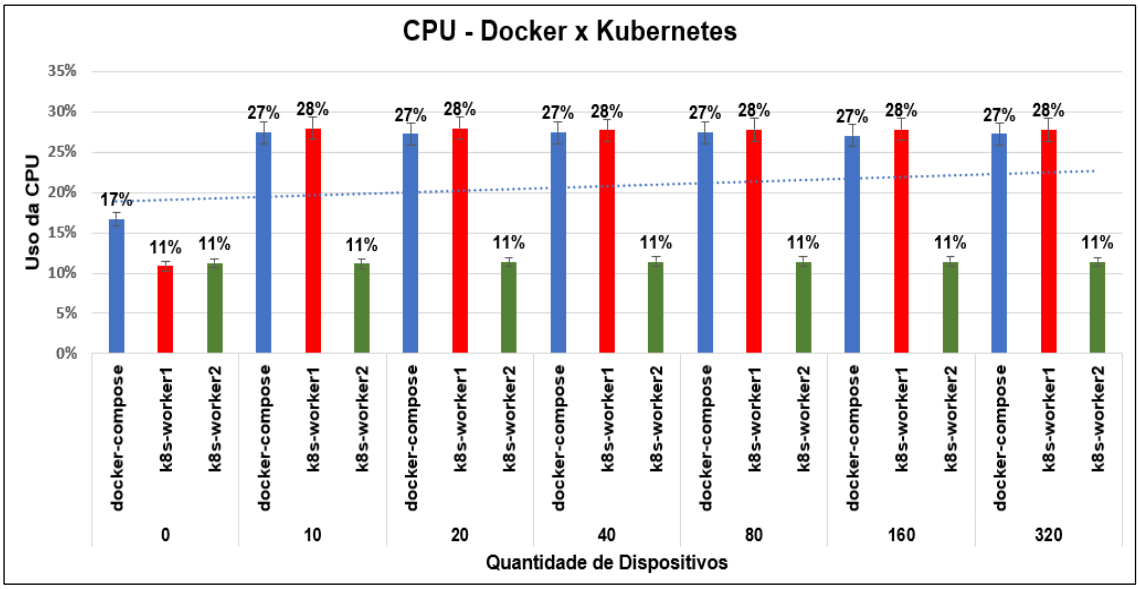
Mediante ao exposto, serão apresentados os resultados e análises dessas simulações realizadas nas máquinas (nós) da Dojot com *Docker Compose* e com *Kubernetes*. Vale ressaltar que as máquinas "***k8s-worker1***" e "***k8s-worker2***", são as que realmente executam os serviços da Dojot e possuem função "equivalente" a máquina "***docker-compose***". Sendo que todos os serviços exclusivamente da Dojot foram fixados para serem executados na máquina "***k8s-worker1***" e os serviços do componente *Kafka* e *x509* foram fixados na máquina "***k8s-worker2***".

5.2 Resultados e análises

No Gráfico 1, é possível visualizar a média do uso de CPU nas máquinas, à medida que o número de dispositivos recebendo dados vai aumentando exponencialmente. É possível visualizar que quando nenhum dispositivo está recebendo dados o consumo médio de CPU na máquina ***docker-compose*** é de 17%, enquanto nas máquinas do cluster *Kubernetes*, ***k8s-worker1*** e ***k8s-worker2***, é de 11%. Logo, é possível verificar, uma diferença de 6% a menos na tecnologia *Kubernetes*, o que a torna mais vantajosa, nesse primeiro momento, quando a Dojot está apenas executando os seus serviços, sem receber dados, em relação a métrica de CPU.

Após começar a receber dados esse consumo aumenta para 27% na máquina ***docker-compose*** e para 28% na máquina ***k8s-worker1***, enquanto na máquina ***k8s-worker2*** ele permanece 11%. O motivo pelo qual ocorreu esse aumento apenas na ***k8s-worker1***, em detrimento da ***k8s-worker2***, é porque todos os serviços (e micro serviços) relacionados exclusivamente a Dojot, estão sendo executados na ***k8s-worke1***, conforme supracitado. Além disso, é possível observar que medida que a quantidade de dispositivos recebendo dados aumenta – para 10, 20, 40, 80 160 e 320 – essa porcentagem também tende a aumenta, porém esse aumento é linear, conforme pode ser visualizado na reta da linha de tendência linear pontilhada em azul.

Gráfico 2. Uso de CPU nas máquinas da Dojot.



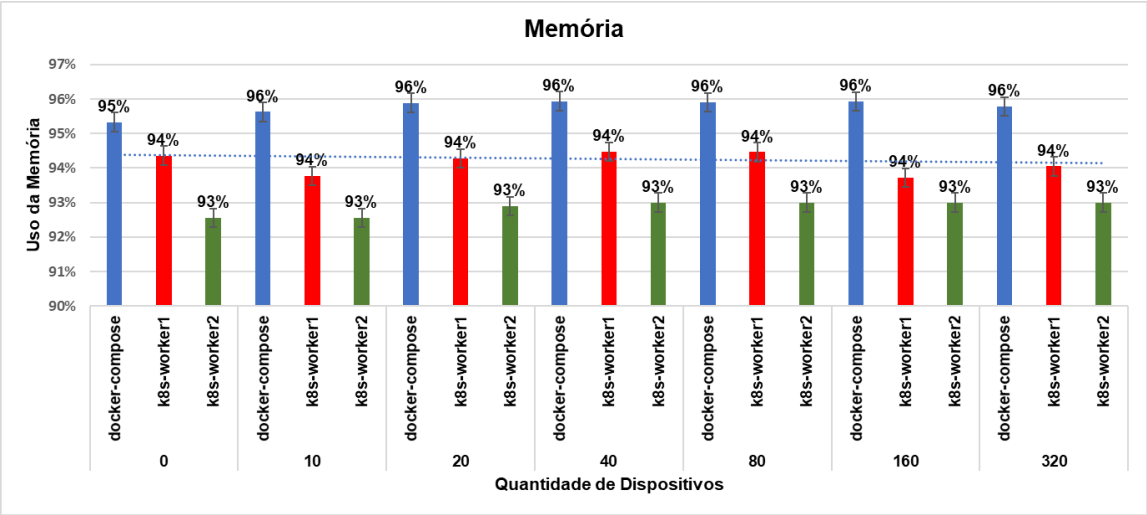
Fonte: produzido pela Autora.

Em relação ao uso de memória, no Gráfico 3, é possível visualizar a média do uso de CPU nas máquinas, à medida que o número de dispositivos recebendo dados vai aumentando exponencialmente para 10, 20, 40, 80 160 e 320. No Gráfico 3, é possível visualizar que quando nenhum dispositivo está recebendo dados o consumo médio de memória nas máquinas ***docker-compose*** é de 95%, enquanto nas máquinas do cluster *Kubernetes*, ***k8s-worker1*** e ***k8s-worker2***, é de 94%. Logo, é possível verificar, uma diferença pequena de apenas 1%, de uma tecnologia em relação a outra. Contudo, ainda assim, o consumo de memória é menor na máquina ***docker-compose***.

Após começar a receber dados esse consumo aumenta para 96% apenas na máquina ***docker-compose*** e se mantém em 94% na máquina ***k8s-worker1*** e 93% na máquina ***k8s-worker2*** – independentemente da quantidade de dispositivos que estejam recebendo dados. Demonstrando assim que a Dojot implantada no *cluster Kubernetes* responde melhor a escalabilidade em relação a métrica de memória.

Além disso, ainda no Gráfico 3, é possível observar que medida que a quantidade de dispositivos recebendo dados aumenta – para 10, 20, 40, 80 160 e 320 – essa porcentagem também tende a diminuir, conforme pode ser visualizado na reta da linha de tendência linear pontilhada em azul. Isso ocorreu porque à medida que a quantidade de dispositivos aumentou, *IoT Agent* demorou mais tempo enviar os dados fazendo com que a Dojot ficasse mais tempo sem receber dados

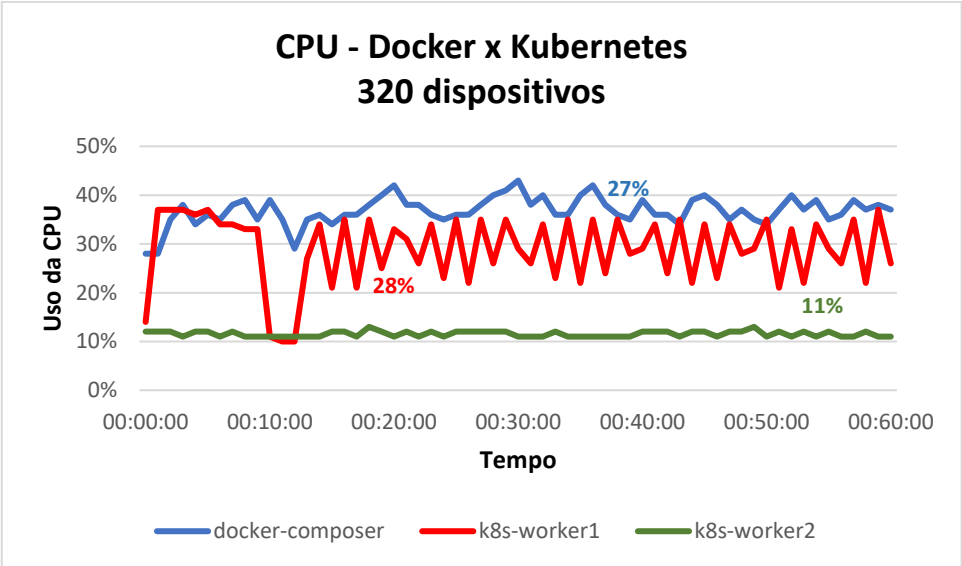
Gráfico 3. Uso de memória nas máquinas da Dojot.



Fonte: produzido pela Autora.

Como forma de analisar as curvas das máquinas ao longo do tempo, durante o período de 60 minutos (1 hora), tem-se o Gráfico 4, mostrando do uso de CPU, quando 320 dispositivos recebem dados simultaneamente. No Gráfico 4 possível verificar que o uso médio da CPU na máquina ***docker-compose*** foi de 27%, já nas máquinas do cluster *Kubernetes*, foi de 28% no ***k8s-worker1*** e 11% ***k8s-worker2***, o que representa uma média total de 25,5% nessas duas máquinas *workers*.

Gráfico 4. Uso de CPU nas máquinas da Dojot recebendo dados simultaneamente em 320 dispositivos.

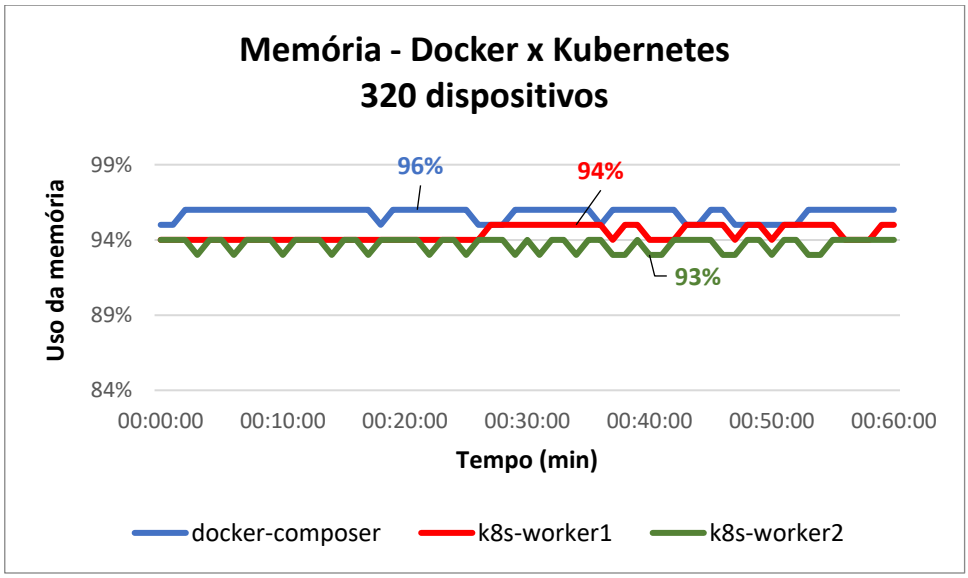


Fonte: produzido pela Autora.

No Gráfico 5 é possível visualizar a curva ao longo, durante o período de 60 minutos, do uso de memória, quando 320 dispositivos recebem dados

simultaneamente. No Gráfico 5 é possível verificar que o uso da CPU foi mais constante ao longo do período analisado, tendo mais variação na máquina ***k8s-worker2***. Além disso, conforme visto anteriormente, o seu consumo foi maior na máquina ***docker-compose*** com média de 96%, enquanto nas máquinas do cluster *Kubernetes*, ***k8s-worker1*** foi de 94% e na máquina ***k8s-worker2*** foi de 93%, o que representa uma média de 93,5%, dessas duas máquinas *workers*.

Gráfico 5. Uso de memória nas máquinas da Dojot recebendo dados simultaneamente em 320 dispositivos.



Fonte: produzido pela Autora.

Assim, com base nas simulações realizadas, destaca-se que a Dojot no **ambiente de desenvolvimento** com ***Docker-Compose*** **apresentou um menor consumo de CPU** e maior consumo de memória. Enquanto **o ambiente de produção** apresentou maior consumo de CPU e menor consumo de memória, conforme ilustra o Quadro 4. Além disso, foi possível observar que independentemente da tecnologia utilizada o recurso de maior consumo da Dojot é a memória.

Quadro 4 – Síntese dos resultados.

Métrica	**Ambiente de desenvolvimento**	**Ambiente de Produção**
CPU	↓	↑
Memória	↑	↓

Fonte: produzido pela Autora.

Em relação ao consumo de energia, considerando que as máquinas onde a Dojot está implantada precisam ficar ligadas 24/7, ou seja, 24 horas por dia, durante todos os dias da semana, é mais vantajoso utilizar a Dojot no ambiente de desenvolvimento, com *Docker-Compose*, haja visto que é preciso utilizar apenas uma máquina para implantá-la, seja ela física ou virtual.

5.3 Considerações Finais do Capítulo

Neste Capítulo foi possível verificar a escalabilidade dos dois ambientes onde a Dojot foi implantada, em relação ao uso das métricas de CPU e memória, através de simulações. Sendo o ambiente de desenvolvimento o de melhor performance em relação ao consumo de CPU e o ambiente de produção o de melhor performance em relação ao consumo de memória e o que responde melhor a questão da escalabilidade. Destaca-se ainda que o recurso mais consumido nas máquinas, em ambos os ambientes, é o de memória.

6 CONCLUSÃO

Neste último Capítulo 6, serão descritas algumas considerações finais sobre este trabalho. Serão feitas algumas sugestões de trabalho futuros e serão apresentadas as produções acadêmicas originadas a partir desta dissertação.

6.1 Considerações finais

Mediante a tudo o que foi descrito ao longo deste trabalho, conclui-se que esta dissertação alcançou o seu objetivo principal que visava implantar a plataforma Dojot de forma escalável para ser o *middleware* IoT das aplicações de um projeto de mobilidade elétrica multimodal. Ressalta-se que em virtude do ambiente de virtualização (Proxmox) onde as VM da Dojot foram implantadas e do *cluster Kubernetes*, essa escalabilidade pode ser horizontal e vertical.

Neste trabalho foram adotadas e investigadas as tecnologias utilizadas Proxmox, *Docker, Docker-Compose, Kubernetes* e Dojot. Destacando-se o fato de que esse trabalho utilizou somente tecnologias *open sources* e gratuitas. Esse fato, soma positivamente para o contexto acadêmico e em caso de desenvolvimento de projeto de P&D, semelhantes ao SIMA.

Na revisão da literatura foi possível observar que a Dojot é uma das principais plataformas (*middleware*) de IoT disponível e ativa no Brasil atualmente. Além disso, foi possível verificar que ela pode ser utilizada em diversos tipos de aplicações, incluindo as de mobilidade elétrica. Sendo possível, utilizá-la sem limitações em relação a quantidade de dispositivos, conexões, entre outras. Destacando-se assim, a importante contribuição desta dissertação para a literatura em relação ao uso da plataforma Dojot.

No que diz respeito ao *middleware* IoT implantado, a Dojot, foram criados dois ambientes, sendo um ambiente de desenvolvimento para teste e um ambiente de

produção para o sistema final do projeto SIMA. Além disso, foram realizadas simulações para testar os ambientes implantados. Com base nas simulações realizadas, foi possível observar que o ambiente de desenvolvimento apresentou uma melhor performance em relação ao consumo de CPU, enquanto o ambiente de produção apresentou uma melhor performance em relação ao consumo de memória e respondeu melhor a escalabilidade. Destaca-se que os ambientes implantados estão funcionais, em operação e suportando escalabilidade.

A partir deste trabalho, conclui-se que a implementação da plataforma Dojot como *middleware* IoT das aplicações de mobilidade elétrica do projeto SIMA, pode contribuir para os avanços da pesquisa e desenvolvimento na área da mobilidade elétrica no Brasil e na região amazônica.

6.2 Trabalhos futuros

Propõe-se como trabalhos futuros os seguintes tópicos:

- Adicionar a métrica de I/O do disco nas análises;
- Avaliar o uso das métricas de CPU, memória e I/O quando a Dojot recebe e envia dados simulados, simultaneamente;
- Avaliar o uso das métricas supracitadas quando a Dojot recebe e envia dados reais, dos dispositivos físicos do projeto;
- Avaliar o uso das métricas supracitadas quando a Dojot recebe e envia dados reais, dos dispositivos físicos do projeto, variando o número de requisições por minuto;
- Avaliar as métricas supracitadas quando os serviços a da Dojot não estão fixos em uma única máquina do *cluster Kubernetes;*
- Aplicar técnicas de estatística para analisar as métricas supracitas, como por exemplo Análise T;

6.3 Produções Acadêmicas

Destaca-se que uma parte do desenvolvimento desta dissertação foi publicada no artigo “**Implementação de um cluster *Kubernetes* com a plataforma Dojot para Aplicações de Internet das Coisas**” aprovado e apresentado no Seminário

Integrado de *Software* e *Hardware* (SEMISH 2021), principal fórum científico do Congresso da Sociedade Brasileira de Computação (CSBC). Citado como:

1. MORAES, Jean; **LOBATO, Elen**; ROSÁRIO, Denis; BEZERRA, Ubiratan; CERQUEIRA, Eduardo; TOSTES, Maria; ANTLOGA, Andréia. **Implementação de um cluster *Kubernetes* com a plataforma Dojot para Aplicações de Internet das Coisas**. In: SEMINÁRIO INTEGRADO DE *SOFTWARE E HARDWARE* (SEMISH), 2021, Evento Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021 . p. 1-8. ISSN 2595-6205. DOI: https://doi.org/10.5753/semish.2021.15801.

6.4 Outras Produções

- **Artigos**

Houve ainda a aprovação e apresentação do artigo "**Smart City: application of the ABNT NBR ISO 37122:2020 Standard in the University City of UFPA**", aprovado na *International Conference on Industry Applications* 2021 (INDUCON 2021) IEEE/IAS. Citado como:

1. **LOBATO, Elen**; SOUZA, Ana; MUSE, Larissa; BEZZERRA, Ubiratan; TOSTES, Emília; PAIXÃO, Ulisses, FONSECA, Wellington; CERQUEIRA, Eduardo; NASCIMENTO, Andréia. Smart City: application of the ABNT NBR ISO 37122: 2020 Standard in the University City of UFPA. In: **14th IEEE International Conference on Industry Applications (INDUSCON)**. IEEE, 2021. p. 1258-1265. DOI: 10.1109/INDUSCON51756.2021.9529522.

- **Registros de *Softwares***

Foi feito o pedido no Instituto Nacional da Propriedade Industrial (INPI) de dois registros de *softwares* de aplicações que utilizam a infraestrutura da plataforma Dojot, implementada neste trabalho, são eles:

1. TOSTES, Emília; BEZERRA, Ubiratan; MUNIZ, João; CARVALHO, Izídio; **LOBATO, Elen**; ALBUQUERQUE, Bruno; PAIXÃO, Ulisses; MORAES, Jean. **Sistema de Gestão de Geração de Energia Elétrica Fotovoltaica**. Titular: Norte Energia S.A; Universidade Federal do Pará; BYD Energy do Brasil

LTDA; ABB Eletrificação LTDA. Procurador: José Carlos Vaz e Dias. BR nº BR512021003140-7. Pedido: 20 dez. 2021.

2. TOSTES, Emília; BEZERRA, Ubiratan; MUNIZ, João; CARVALHO, Izídio; **LOBATO, Elen**; ANDRADE, Vinícius; DUARTE, Ana; LIMA, Áthila; MORAES, Wuanda. **Sistema de Gestão de Armazenamento de Energia Elétrica em Banco de Baterias**. Titular: Norte Energia S.A; Universidade Federal do Pará; BYD Energy do Brasil LTDA; ABB Eletrificação LTDA. Procurador: José Carlos Vaz e Dias. BR nº BR512021003139-3. Pedido: 20 dez. 2021.

Referências

ABNT, Associação Brasileira de Normas Técnicas. **Cidades e comunidades sustentáveis - Indicadores para cidades inteligentes**. Campos Elíseos -SP: [s.n.]. 2020.

ABUABDO, A.; AL-SHARIF, Z. A. **Virtualization vs. Containerization: Towards a Multithreaded Performance Evaluation Approach**. 16th ACS/IEEE International Conference on Computer Systems and Applications AICCSA 2019 : 3 November to 7 November 2019, Al Ain University & Crowne Plaza, Abu Dhabi, UAE. **Anais**...2019.

ARSLAM, I; ÖZBILGIN, I. G.. **Virtualization and security: Examination of a virtualization platform structure**. 2017 International Conference on Computer Science and Engineering (UBMK), 2017.

ANEEL. **Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional (PRODIST) : Módulo 8 - Qualidade da Energia Elétrica**. p. 88, 2021.

AZEVEDO, R. S. DE; MARQUES, B. P. A.; CASTRO, V. M; BARROS, V. A; CABRINI, F. H. FaSCi-Tech Smart Bike: Plataforma Aberta para Monitoramento e Gestão de Transporte Urbano baseado em Bicicletas Elétricas e IoT. **FaSCi-Tech, v. 1, n. 16**, 2020.

BRASIL. **Lei Nº 12.587 da Política Nacional de Mobilidade Urbana.**Brasília, Distrito Federal, Brasil, 2012.

BUSATO, L. F.; ALMEIDA, F. M. M. C. DE. Desenvolvimento de aplicativo para aprimoramento da mobilidade coletiva na Unicamp. **Revista dos Trabalhos de Iniciação Científica da UNICAMP**, v. 27, 2019.

CARVALHO, H.; SÁ, J.; FARIAS, F. **Implantação de uma Arquitetura de Software para Monitoramento de Dados Ambientais em um Cenário de Smart**

Campus. Conferência: Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais, 2021.

CHAQFEH, M. A.; MOHAMED, N. **Challenges in middleware solutions for the internet of things**. Proceedings of the 2012 International Conference on Collaboration Technologies and Systems, CTS 2012. **Anais**...2012.

CISCO. **Relatório de tendências globais de rede para 2022: A ascensão da rede como serviço (NaaS)**. [s.l: s.n.]. 2022.

CIVALI, V; ALMEIDA, M. DE; SANTOS, C. DOS; MARIOTTO, F. Metodologia de análise de presença de dispositivos com Wi-Fi para estimar a demanda de passageiros dos circulares internos da Unicamp. **Revista dos Trabalhos de Iniciação Científica da UNICAMP**, p. 1–1, 2019.

CPQD. **Dojot Soluções para IOT - Plataforma de Desenvolvimento para IOT**. Disponível em: <https://dojot.com.br/>. Acesso em: 27 jan. 2017.

CPQD. **Documentação Dojot v0.7.0**. Disponível em: <https://dojotdocs.readthedocs.io/_/downloads/pt_BR/latest/pdf/>. Acesso em: 27 jan. 2021.

DA CRUZ, M. A. A.; RODRIGUES, J. J. P. C; KOROTAEV, V.; ALBUQUERQUE, V. H.. A Reference Model for Internet of Things Middleware. **IEEE Internet of Things Journal.** Institute of Electrical and Electronics Engineers Inc., , 1 abr. 2018.

DA SILVA, L. C. P.; MEIRA, P. C. M.; CYPRIANO, J. G. I.; AZZINI, H. A. D; SANTOS, A. Q. Software toolchain to enhance the management and integration of a sustainable campus model. **Energy Informatics**, v. 4, 1 set. 2021a.

DEWI, L. P.; NOETJAHYANA, A.; PALIT, H. N.; YESUTUN, K. **Server Scalability Using Kubernetes**. TIMES-iCON2019 : the 4th Technology Innovation Management and Engineering Science International Conference : 11th-13th December 2019, Thailand. **Anais**...2019.

DIRAN, D.; VEENSTRA, A. F. V; TJERK, T.; KIROVA, M. **Artificial Intelligence in smart cities and urban mobility**. [s.l: s.n.]. 2021.

DOCKER. **Docker overview | Docker Documentation**. Disponível em: <https://docs.docker.com/get-started/overview/>. Acesso em: 27 jan. 2022a.

DONNELLAN, P. R. The future of mobility-Electric, autonomous, and shared vehicles. **IEEE Engineering Management Review.** Institute of Electrical and Electronics Engineers Inc., , 1 dez. 2018.

DUARTE, F. , L. ; LIBARDI, R... **Introdução à Mobilidade Urbana**. Juruá Editora.p.108. 2007.

FIWARE FOUNDATION. **About FIWARE - FIWARE**. Disponível em: <https://www.fiware.org/about-us/>. Acesso em: 13 fev. 2022.

GOES, M. A.; CASTRO, M. S.; COIMBRA, M. L. CELASCHI, S. **Uma Arquitetura Integrada de Referência a ser Adotada no Desenvolvimento de Soluções para Internet das Coisas, IoT**. XXXVII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais – SBrT2019, 2019.

IEEE. **"IEEE Draft Standard for Framework of Blockchain-based Internet of Things (IoT) Data Management**. [s.l.] IEEE, 2020.

IMBAR, R. V.; SUPANGKAT, S. H.; LANGI, A. Z. R. **Smart Campus Model: A Literature Review**. 7th International Conference on ICT for Smart Society: AIoT for Smart Society, ICISS 2020 - Proceeding. **Anais**...Institute of Electrical and Electronics Engineers Inc., 19 nov. 2020.

JAIN, N.; CHOUDHARY, S. **Overview of virtualization in cloud computing**. 2016 Symposium on Colossal Data Analysis and Networking, CDAN 2016. **Anais**...Institute of Electrical and Electronics Engineers Inc., 16 set. 2016.

KHARE, S.; TOTARO, M. **Big Data in IoT**. 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). **Anais**...2019.

KUBERNETES. **O que é Kubernetes? | Kubernetes**. Disponível em: <https://kubernetes.io/pt-br/docs/concepts/overview/what-is-kubernetes/>. Acesso em: 12 fev. 2022.

KUBERNETES. **Kubernetes | Kubernetes**. Disponível em: <https://kubernetes.io/pt-br/docs/home/>. Acesso em: 27 jan. 2022.

LOBATO, E. et al. **Smart City: application of the ABNT NBR ISO 37122:2020 Standard in the University City of UFPA**. 2021 14th IEEE International Conference on Industry Applications (INDUSCON), 2021.

LOPES, Y. **O efeito mediador da aplicação da Internet das Coisas na relação entre gestão estratégica da logistica e desempenho operacional**. São Paulo. Dissertação de Mestrado. Programa de Pós-Graduação em Administração de Empresas da Universidade Presbiteriana Mackenzie, 2018.

MA, L; CHEN, Y.; SUNA, Y.; WU, Q. **Virtualization maturity reference model for green software**. Proceedings - 2012 International Conference on Control Engineering and Communication Technology, ICCECT 2012. **Anais**...2012.

MAHREZ, Z.; SABIR, E.; BADIDI, E. SAAD, W; SADIK, M.. Smart Urban Mobility: When Mobility Systems Meet Smart Data. **IEEE Transactions on Intelligent Transportation Systems**, 2021.

MAZIERO, C. **Virtualização: Conceitos e Aplicações em Segurança Doutorado View project Secure E-Voting System View project**. [s.l: s.n.]. Disponível em: <https://www.researchgate.net/publication/237681120>.

MEDEIROS, T.; CAMPOS, C. **ARTSIA: Escalabilidade de Aplicações para Sistemas Inteligentes de Transportes em um Ambiente de Computação em Neblina**. Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais. **Anais**. 2020.

MERKEL, D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. **Linux journal**, v. 2014, p. 2, 2014.

MITRPANONT, J. L. et al. **A Scalable and Low-Cost MQTT Broker Clustering System**. 2017 2nd International Conference on Information Technology (INCIT). **Anais**...2017.

MODESTO, W.; NETO, A.; DO ROSÁRIO, D. CERQUEIRA, E.. **SG2IoT - Uma Arquitetura para Integração de Dispositivos Elétricos Inteligentes de Abordagem Legada em Sistemas Smart Grid Baseados na IoT**. Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP), 2021.

MORAES, J.; LOBATO, E. P. S.; ROSÁRIO, D. BEZZERRA, U.; TOSTES, M. ANTLOGA, A. **Implementação de um cluster Kubernetes com a plataforma Dojot para Aplicações de Internet das Coisas**. Porto Alegre: 2021.

ONU. **Sustainable Development Goal 10: Redução das desigualdades | As Nações Unidas no Brasil**. Disponível em: <https://brasil.un.org/pt-br/sdgs/10>. Acesso em: 14 fev. 2022a.

ONU. **Acordo de Paris sobre o Clima | As Nações Unidas no Brasil**. Disponível em: <https://brasil.un.org/pt-br/node/88191>. Acesso em: 30 jan. 2022b.

PEREIRA FERREIRA, A.; SINNOTT, R. **A performance evaluation of containers running on managed kubernetes services**. Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom. **Anais**...IEEE Computer Society, 1 dez. 2019.

PINELI, J. **Leia isso antes de decidir a plataforma IOT! - Tudo Sobre IoT**. Disponível em: <https://tudosobreiot.com.br/lei-isso-antes-de-decidir-a-plataforma-iot/>. Acesso em: 12 fev. 2022.

PINTO, K. S. et al. **Um Estudo de Caso de Internet das Coisas utilizando o ecossistema Dojot: da configuração a utilização**. Anais X Conferência Nacional em Comunicações, Redes e Segurança da Informação – ENCOM 2020. **Anais**.2020.

PINTO, M. R. **Internet das coisas, cidades inteligentes e mobilidade urbana : um estudo de caso sobre os Smart Parkings em vias públicas e os impactos na qualidade de vida da população**. Niterói, RJ: Trabalho de Conclusão de Curso. Curso de Tecnologia em Sistemas de Computação da Universidade Federal Fluminense., 2017.

PNME. **1º Anuário Brasileiro da Mobilidade Elétrica**. Brasília e Rio de Janeiro: [s.n.].

PROXMOX. **PROXMOX VE ADMINISTRATION GUIDE RELEASE 7.1**. Disponível em: <https://pve.proxmox.com/pve-docs/pve-admin-guide.pdf>. Acesso em: 27 jan. 2022.

RED HAT. **What is virtualization?** Disponível em: <https://www.redhat.com/pt-br/topics/virtualization/what-is-virtualization>. Acesso em: 1 fev. 2022a.

RED HAT. **O que é Kubernetes (K8s)?** Disponível em: <https://www.redhat.com/pt-br/topics/containers/what-is-kubernetes>. Acesso em: 1 fev. 2022b.

RED HAT. **O que é um hipervisor?** Disponível em: <https://www.redhat.com/pt-br/topics/virtualization/what-is-a-hypervisor>. Acesso em: 31 jan. 2022.

SANTOS, B.; SILVA, A.; CELES, C.; BORGES, J.; PERES, B.; VIEIRA, M.; VIEIRA, L; GOUSSEVKAIA, O; LOUREIRO, A. **Internet das coisas: da teoria à prática**. XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2016). **Anais**.2016.

SÁ, J. S. et al. **Estudo de caso sobre a comunicação entre Dojot e Aplicações Android ou Web**. Anais X Conferencia Nacional em Comunicações, Redes e Segurança da Informação – ENCOM 2021. **Anais**.2021.

SANTOS, S. C.; FIRMININO, R. M.; MATTOS, D. M. F; MEDEIROS, D. S. V. **An IoT Rainfall Monitoring Application based on Wireless Communication Technologies**. 2020 4th Conference on Cloud and Internet of Things, CIoT 2020. **Anais**...Institute of Electrical and Electronics Engineers Inc., 7 out. 2020.

SILVA, E. N. et al. **Internet das Coisas com Banco de Dados Relacional e em Tempo Real**. Anais X Conferencia Nacional em Comunicações, Redes e Segurança da Informação – ENCOM 2020. **Anais**.2020.

SILVA, E. N. et al. **Algoritmos para determinação de rotas e previsão de chegada de ônibus em um Smart Campus**. Anais X Conferência Nacional em Comunicações, Redes e Segurança da Informação – ENCOM 2021. **Anais**...2021.

TANEMBAUM. A.S. **Redes de Computadores.** 4ª ed. Brasil: Elsevier, 2003.

VIJAYAKUMAR, K. **Solar Charging Infrastructure for E-vehicles-A Review**. 2021 International Conference on Advance Computing and Innovative Technologies in Engineering, ICACITE 2021. **Anais**.Institute of Electrical and Electronics Engineers Inc., 4 mar. 2021.

VITALINO, J. F. N.; CASTRO, M. A. N. **Descomplicando o Docker**. 2ª Edição ed. [s.l.] Brasport, 2018.

XAVIER, R. F. **Desenvolvimento de uma aplicação de monitoramento utilizando a plataforma IoT Dojot**. Instituto Federal de Educação, Ciência e Tecnologia de Goiás – Campus Inhumas.: Trabalho de Conclusão de Curso. , 2020.

Apêndice A

diesel_<nome>	
type	**attr**
JSON	tensao_rms
float	tensao_bateria_cc
JSON	corrente_rms
float	fator_potencia
float	potencia_ativa
float	potencia_reativa
float	potencia_aparente
float	nivel_tanque
float	eficiencia_gerador
float	consumo_oleo_gerador
float	temperatura
float	frequencia
float	velocidade
float	horas_operacao_total
float	horas_operacao_mes
float	horas_operacao_ultima_manutencao
Texto	endereco
Geo	localizacao

barco_<nome>	
type	**attr**
float	tensao
float	corrente
float	temperatura_bateria
float	energia_final_necessaria
float	autonomia
float	estado_carga
float	consumo
float	potencia_ativa
float	potencia_ativa_por_usuario
float	consumo_por_usuario
float	consumo_por_quilometro
float	quilometragem
float	temperatura_interna
float	temperatura_externa
float	status_carregamento
float	quantidade_usuarios
Texto	endereco
Geo	localizacao

posto_barco_<nome>	
type	**attr**
JSON	tensao_rms
JSON	corrente_rms
JSON	pot_ativa
JSON	pot_reativa
JSON	pot_aparente
JSON	fator_potencia
JSON	consumo
JSON	harm_ind_tensao
JSON	harm_ind_corrente
JSON	harm_total_tensao
JSON	harm_total_corrente
float	quantidade_usuarios
float	potencia_ativa_por_usuario
float	consumo_por_usuario
float	tarifa_por_usuario
float	tarifa_total
Texto	endereco
Geo	localizacao

onibus_<nome>	
type	**attr**
float	tensao
float	corrente
float	temperatura_bateria
float	energia_final_necessaria
float	autonomia
float	estado_carga
float	consumo
float	potencia_ativa
float	consumo_por_usuario
float	consumo_por_quilometro
float	quilometragem
float	temperatura_interna
float	temperatura_externa
float	status_carregamento
float	quantidade_usuarios
Texto	endereco
Geo	localizacao

posto_onibus_<nome>	
type	**attr**
JSON	tensao_rms
JSON	corrente_rms
JSON	pot_ativa
JSON	pot_reativa
JSON	pot_aparente
JSON	fator_potencia
JSON	consumo
JSON	harm_ind_tensao
JSON	harm_ind_corrente
JSON	harm_total_tensao
JSON	harm_total_corrente
float	quantidade_usuarios
float	potencia_ativa_por_usuario
float	consumo_por_usuario
float	tarifa_por_usuario
float	tarifa_total
Texto	endereco
Geo	localizacao

fv_<nome>	
type	**attr**
float	tensao
float	corrente
float	potencia
float	eficiencia_conversao
float	fator_dimensionamento_inversor
float	fator_capacidade
float	energia_especifica
float	produtividade_sistema
float	produtividade_referencia
float	rendimento_global
float	intensidade_instalacao
Texto	endereco
Geo	localizacao

uac_tri_<nome>	
type	**attr**
JSON	tensao_rms
JSON	corrente_rms
JSON	pot_ativa
JSON	pot_reativa
JSON	pot_aparente
JSON	fator_potencia
JSON	consumo
JSON	harm_ind_tensao
JSON	harm_ind_corrente
JSON	harm_total_tensao
JSON	harm_total_corrente
JSON	frequencia
Texto	endereco
Geo	localizacao

bat_<nome>	
type	**attr**
JSON	tensao
float	corrente
float	potencia
float	temperatura
float	energia_final_necessaria
float	energia_consumida
float	energia_fornecida
float	autonomia
float	estado_carga
Texto	endereco
Geo	localizacao

est_meteor_<nome>	
type	**attr**
float	pressao_barometrica
float	temperatura
float	umidade_relativa
float	velocidade_vento
float	velocidade_rajada
float	direcao_vento
float	precipitacao
float	radiacao_total
float	radiacao_refletida
float	iluminancia
float	suv_detector
float	tensao_bateria_cc
Texto	endereco
Geo	localizacao

gateway_<nome>	
type	**attr**
string	phyPayload
JSON	txInfo
JSON	rxInfo
Texto	endereco
Geo	localizacao

endNode_<nome>	
type	**attr**
JSON	enData
JSON	gwData
Texto	endereco
Geo	localizacao