UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

JAQUELINE SEBASTIANY IAKSCH

MODELO SISTÊMICO PARA TRATAMENTO DE DADOS NO CONTEXTO DA MANUTENÇÃO DE EQUIPAMENTOS AGRÍCOLAS EM UMA USINA DE CANA-DE-AÇÚCAR COMO APOIO À IMPLANTAÇÃO DE SISTEMAS DE AGRICULTURA 4.0

CURITIBA
2023

JAQUELINE SEBASTIANY IAKSCH

MODELO SISTÊMICO PARA TRATAMENTO DE DADOS NO CONTEXTO DA MANUTENÇÃO DE EQUIPAMENTOS AGRÍCOLAS EM UMA USINA DE CANA-DE-AÇÚCAR COMO APOIO À IMPLANTAÇÃO DE SISTEMAS DE AGRICULTURA 4.0

Systemic model for data processing in the context of maintenance of agricultural equipment in a sugarcane plant as a support to the implementation of agriculture systems 4.0

Tese apresentada como requisito para obtenção do título de Doutora em Engenharia da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador(a): Milton Borsato.

CURITIBA

2023

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

JAQUELINE SEBASTIANY IAKSCH

MODELO SISTÊMICO PARA TRATAMENTO DE DADOS NO CONTEXTO DA MANUTENÇÃO DE EQUIPAMENTOS AGRÍCOLAS EM UMA USINA DE CANA-DE-AÇÚCAR COMO APOIO À IMPLANTAÇÃO DE SISTEMAS DE AGRICULTURA 4.0

Trabalho de pesquisa de doutorado apresentado como requisito para obtenção do título de Doutora Em Engenharia da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia De Manufatura.

Data de aprovação: 24 de Fevereiro de 2023

Dr. Milton Borsato, Doutorado - Universidade Tecnológica Federal do Paraná

Dra. Carla Cristina Amodio Estorilio, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Eduardo De Freitas Rocha Loures, Doutorado - Pontifícia Universidade Católica do Paraná (Pucpr)

Dr. Jairo Muller Wolf, Doutorado - Universidade Positivo (Up)

Dr. Walter Luis Mikos, Doutorado - Universidade Tecnológica Federal do Paraná

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 24/05/2023.

AGRADECIMENTOS

Agradeço primeiramente a Deus por toda a força concedida na realização desse sonho.

A minha família que me deu total apoio. Especialmente à minha mãe e ao meu noivo pela compreensão e apoio em todas as horas.

Ao meu orientador Prof. Milton Borsato, pelo apoio, orientação e paciência, por ter estado sempre presente em todos os momentos da elaboração desse trabalho.

Aos meus colegas de laboratório que estiveram presentes ajudando direta e indiretamente nessa jornada.

Aos Professores do PPGEM pelos conhecimentos transmitidos em disciplinas por mim cursadas, estendendo meus agradecimentos a todos os funcionários do departamento.

À UTFPR, pela oportunidade de realização do doutorado.

Agradeço também à CNH Industrial, em especial ao Maurício Infantini e Carlos Visconti pela parceria firmada nesta pesquisa.

Ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), na modalidade do programa DAI – Doutorado Acadêmico para Inovação (nº 157258/2019-0), e à CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, na modalidade do PDSE – Programa de Doutorado Sanduíche no Exterior (nº 88881.624353/2021-01), pelo suporte no desenvolvimento dessa pesquisa.

A todos que contribuíram de forma particular e especial, os meus agradecimentos.

“A tarefa não é tanto ver aquilo que ninguém
viu, mas pensar o que ninguém ainda pensou
sobre aquilo que todo mundo vê.”
(Arthur Schopenhauer)

RESUMO

A agricultura vem enfrentando desafios cada vez maiores devido a diversos fatores como o crescimento populacional e mudanças climáticas. Tem-se buscado maneiras de melhorar a rentabilidade e eficiência agrícola através de tomadas de decisões e de um melhor controle gerencial baseado em informações e conhecimentos gerados na fazenda em seu contexto específico. Com a chegada da chamada quarta revolução industrial, tecnologias e soluções inovadores vêm sendo aplicadas como alternativa para a coleta e processamento dessas informações, as quais são geradas por sistemas, equipamentos, mercados e agentes envolvidos na produção agrícola. Transformar esses dados coletados em campo em dados de valor que apoiem uma melhor tomada de decisão é essencial. Para apoiar as tomadas de decisão em nível estratégico, tático e operacional, o presente trabalho desenvolveu inicialmente um modelo a fim de proporcionar uma visão estratégica da Agricultura 4.0. Após foi construído um modelo sistêmico que representa todas as informações ao longo de um cenário agrícola genérico, possibilitando o apoio das decisões táticas relacionadas às Fazendas 4.0. Por fim, desenvolveu-se um modelo de informação com o objetivo de selecionar e interpretar dados originados de sensores presentes em implementos agrícolas, possibilitando auxiliando tomadas de decisão operacionais relativas à necessidade de manutenção de equipamentos, de maneira preditiva. Seguindo a metodologia *Design Science Research* (DSR), a qual objetiva desenvolver conhecimento através da resolução de problemas do mundo real, o presente trabalho é dividido em seis etapas: (*i*) Identificação do problema e motivação; (*ii*) Definição dos objetivos da Solução; (*iii*) Projeto e desenvolvimento da Solução; (*iv*) Demonstração da Solução; (*v*) Avaliação da Solução; e, (*vi*) Comunicação dos resultados. A solução proposta nesta pesquisa foi aplicada e avaliada no contexto de uma empresa de máquinas agrícolas parceira, no cenário de plantações de cana-de-açúcar, e possibilitou auxiliar tomadas de decisão relativas à manutenção de filtros de combustível de colheitadeiras de cana.

Palavras-chave: agricultura 4.0; fazenda 4.0; modelo sistêmico; programação baseada em fluxo.

ABSTRACT

Agriculture is facing increasing challenges due to several factors, such as population growth and climate change. Ways have been sought to improve profitability and agricultural efficiency through decision-making and better management control based on information and knowledge generated on the farm in its specific context. Technologies and innovative solutions have been applied as an alternative for collecting and processing this information with the arrival of the so-called fourth industrial revolution, which is generated by systems, equipment, markets, and agents involved in agricultural production. Transforming this data collected in the field into valuable data that supports better decision-making is essential. This work initially developed a model to provide a strategic vision of Agriculture 4.0 To support decision-making at a strategic, tactical, and operational level. Afterward, a systemic model was built that represents all the information along a generic agricultural scenario, allowing the support of tactical decisions related to Farms 4.0. Finally, an information model was developed with the objective of selecting and interpreting data originating from sensors present in agricultural implements, allowing them to assist in operational decision-making about the need for equipment maintenance in a predictive way. This work follows the Design Science Research (DSR) methodology, which aims to develop knowledge through solving real-world problems. This work is divided into six stages: (i) Identification of the problem and motivation; (ii) Definition of the objectives of the Solution; (iii) Solution design and development; (iv) Demonstration of the Solution; (v) Solution Evaluation; and, (vi) Communication of results. The Solution proposed in this research was applied and evaluated in the context of a partner agricultural machinery company in the scenario of sugar cane plantations. It enabled auxiliary decision-making regarding the maintenance of fuel filters for sugarcane harvesters.

Keywords: agriculture 4.0; farm 4.0; systemic model; flow-based programming.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*	Interface de Programação de Aplicação
BI	*Business Intelligence*	Inteligência de Negócios
CLP	*Programmable Logic Controller*	Controlador Lógico Programável
CPS	*Cyber-Physical Systems*	Sistemas Ciber Físicos
DSR	*Design Science Research*	-
ETL	*Extract-Transform-Load Process*	Processo de Extração-Transformação-Carregamento
FBP	*Flow-based Programming*	Programação Baseada em Fluxo
FMIS	*Farm Management Information System*	Sistemas de informação de gerenciamento de fazenda
GPS	*Global Positioning System*	Sistema de Posicionamento Global
HTTP	*Hyper Text Transfer Protocol*	Protocolo de Transferência de Hipertexto
IFTTT	*"If This and This Then That"*	"Se isso e isso então aquilo".
IIRA	*Industrial Internet Reference Architecture*	Arquitetura de Referência de Internet Industrial
INCOSE	*International Council on Systems Engineering*	-
IoT	*Internet of Things*	Internet das Coisas
MQTT	*Message Queue Telemetry Transport*	-
M2M	*MAchine to Machine*	Máquina para Máquina
QoS	*Quality of Service*	Qualidade de Serviço
RAMI	*Reference Architectural Model*	Modelo de Referência de Agricultura
4.0	*Industry 4.0*	4.0
RFID	*Radio Frequency Identification*	Identificação por Radiofrequência
SGAM	*Smart Grid Architecture Model*	Modelo de Arquitetura de Rede Inteligente
SITAM	*Stuttgart IT Architecture for Manufacturing*	Arquitetura de TI para Fabricação Stuttgart
SQL	*Structured Query Language*	Linguagem de Consulta Estruturada

SYSML	*Systems Modeling Language*	Linguagem de Modelagem de Sistemas
UML	*Unified Language Modeling*	Linguagem de Modelagem Unificada
OMG	*Object Management Group*	-
URL	*Uniform Resource Locator*	Localizador Uniforme de Recursos
UTFPR	*Federal University of Technology - Paraná*	Universidade Tecnológica Federal do Paraná
VM	*Virtual Machine*	Máquina Virtual

SUMÁRIO

1 INTRODUÇÃO

A agricultura sempre teve uma enorme importância no desenvolvimento da civilização, sendo considerada um grande pilar da economia (ABBASI; ISLAM; SHAIKH, 2014). No entanto, com o crescimento populacional e aumento da urbanização, crescimento da degradação ambiental e mudanças climáticas, a agricultura moderna vem enfrentando desafios cada vez maiores (FRESCO; FERRARI, 2018). Para vencer esses desafios, busca-se obter maior produtividade e qualidade, de maneira sustentável e com custos menores (SHAMSHIRI et al., 2018; ZAMORA-IZQUIERDO et al., 2019).

Assim, do ponto de vista comercial, agricultores buscam maneiras de melhorar a rentabilidade e eficiência agrícola para reduzir custos, e assim, obter melhores preços para seu produto. Para isso, é necessário tomar decisões melhores e realizar um controle gerencial eficiente através de informações e conhecimentos gerados na fazenda em seu contexto específico (WOLFERT et al., 2017).

Novas tecnologias e soluções vêm sendo aplicadas para fornecer alternativas que auxiliem na coleta e processamento de informações, contribuindo para o aumento da produtividade agrícola (RAY, 2017). Nos últimos anos, com o desenvolvimento tecnológico foram introduzidas mudanças radicais no ambiente de trabalho agrícola, como a utilização de sistemas eletrônicos e transmissão de dados. Além disso, essas mudanças exigiram o fornecimento de informações atualizadas dos sistemas, equipamentos, mercados e agentes envolvidos na produção para as tomadas de decisão estratégicas e gerenciais envolvidas nos processos (PIVOTO et al., 2018).

Como as indústrias de outros segmentos, o setor agrícola vem sendo aprimorado com a chegada da chamada quarta revolução industrial. Como consequência, os métodos utilizados pela indústria agrícola também vêm sofrendo alterações, como a implementação de tecnologias avançadas como sensores e métodos não poluentes (OZDOGAN; GACAR; AKTAS, 2017).

Com isso, as inúmeras inovações tecnológicas que surgiram nas últimas décadas introduziram o conceito de agricultura de precisão (FRESCO; FERRARI, 2018). A agricultura de precisão busca aprimorar a produção agrícola através do monitoramento de fatores ambientais, com a utilização de tecnologias avançadas, fornecendo informações sobre diferentes aspectos (RAJESWARI; SUTHENDRAN; RAJAKUMAR, 2018).

O desenvolvimento e a utilização de sensores de alta precisão para medir o contexto ambiental possibilitou a adoção de técnicas de irrigação mais eficientes, bem como o uso mais preciso de fertilizantes e pesticidas, na busca por maior produtividade e a lucratividade das fazendas (KAMILARIS et al., 2016). No entanto, a produtividade agrícola é afetada não só pelos aspectos ambientais, como diversos outros ligados à produção e gestão (RAJESWARI; SUTHENDRAN; RAJAKUMAR, 2018).

Os dados coletados no campo são fundamentais para as tomadas de decisão agrícolas, assim torna-se essencial transformá-los em dados de valor que apoiem uma melhor tomada de decisão (XIN; ZAZUETA, 2016). O rápido desenvolvimento da internet das coisas e da computação em nuvem impulsionaram o surgimento da chamada agricultura inteligente (WOLFERT et al., 2017).

Embora a agricultura de precisão esteja levando em consideração apenas a variabilidade no campo, a Agricultura 4.0, por sua vez, surge com o objetivo de permitir a melhoria das operações agrícolas e sua gestão, já que trata da coleta, processamento e análise de dados em tempo real, facilitando os processos de tomada de decisão (KAMILARIS et al., 2016).

A evolução das tecnologias agrícolas segue as inovações do setor industrial. Inicialmente a agricultura operava com potência animal, sendo substituída pela introdução do motor a combustão, passando para a chamada Agricultura 3.0 com a utilização de sistemas de orientação e agricultura de precisão (ZAMBON et al., 2019). Essa evolução teve sua continuidade no surgimento do conceito de Agricultura 4.0, também chamada de agricultura inteligente ou agricultura digital, a qual deriva claramente da Indústria 4.0, e tem como base a implementação do conceito de “mundo digital” (OZDOGAN; GACAR; AKTAS, 2017).

A Agricultura 4.0, assim como a Indústria 4.0, representa a combinação e interação das operações agrícolas internas e externas, possibilitando a utilização de informações digitais detalhadas que orientem as decisões ao longo da cadeia de valor agrícola (ZAMBON et al., 2019). Ainda nesse contexto, surge o conceito de Fazenda 4.0, ou fazenda inteligente, na busca pela garantia de que uma propriedade esteja dotada de equipamentos, processos e pessoas direcionados à agricultura digital (O’GRADY; O’HARE, 2017).

Além da introdução de novas práticas e ferramentas, a Agricultura 4.0 tem como um dos seus principais objetivos a adaptabilidade dos sistemas de produção e a melhoria da eficiência (ANDRITOIU et al., 2018). Esta melhoria de eficiência se dá através do aumento da produtividade, que reside na capacidade de se coletar, utilizar e trocar dados de maneira remota em tempo real (BONNEAU et al., 2017). Assim, a automação e a tomada de decisões

inteligentes tornam-se muito importantes para o cumprimento desses objetivos (RAY, 2017; YANE, 2010).

O desenvolvimento de tecnologias de informação e comunicação, internet das coisas (IoT), computação em nuvem e *Big Data* estão alavancando o desenvolvimento da agricultura inteligente (WOLFERT et al., 2017). Através da combinação de tecnologias que fazem uso da computação em nuvem e da IoT, a agricultura digital está evoluindo, uma vez que é possível aproveitar a imensa quantidade de dados agrícolas gerados pelas operações realizadas nas fazendas (ROTZ et al., 2019).

A utilização de dispositivos e tecnologias de comunicação como sensores, satélites e drones é indispensável para se obter os dados necessários, a fim de apoiar o desenvolvimento da agricultura (ANDRITOIU et al., 2018). Sensores são utilizados para coletar informações sobre atributos físicos e ambientais de maneira que as pessoas envolvidas no controle das operações possam reagir e ter controle sobre diversas situações (ABBASI; ISLAM; SHAIKH, 2014).

Os avanços tecnológicos vêm possibilitando também a utilização de veículos aéreos não tripulados (drones), os quais através de sensores e recursos de imagem, auxiliam no aumento da produtividade ao passo que possibilitam um monitoramento em tempo real das plantações (ABBASI; ISLAM; SHAIKH, 2014). Com o auxílio da internet das coisas é possível enviar dados de equipamentos e sensores através da internet, em tempo real (SHENOY; PINGLE, 2016).

A análise de dados passados pode guiar decisões para que sejam feitas escolhas as quais visam otimizar os resultados das fazendas (SHENOY; PINGLE, 2016). Segundo Vercellis (2009), essas decisões podem ser de três tipos: estratégicas, táticas ou operacionais. As decisões estratégicas afetam a organização como um todo. Já as decisões táticas afetam apenas parte da organização, i.e., um departamento, e estão focadas no processo. Por fim, as decisões operacionais referem-se a alguma atividade específica dentro da organização.

Big Data na agricultura refere-se à uma grande quantidade de dados, os quais representam um desafio para as metodologias e plataformas tradicionais de gerenciamento de dados. O armazenamento e processamento dessa grande quantidade de dados pode ser realizado utilizando computação em nuvem (BENDRE; THOOL; THOOL, 2016). Os proponentes da *Big Data* prometem um alto nível de precisão, armazenamento, processamento e análise de informações que anteriormente eram impossíveis devido a limitações tecnológicas (HIMESH et al., 2018).

As tecnologias de *Big Data* estão sendo utilizadas para fornecer informações preditivas em operações agrícolas, permitindo a condução de decisões operacionais em tempo real e possibilitando o redesenho de processos de negócios. Essas tecnologias desempenham um papel essencial no desenvolvimento de um cenário onde seres humanos estão envolvidos apenas em um nível de inteligência muito alto na análise e planejamento de ações (WOLFERT et al., 2017).

Diante dessas tecnologias que a agricultura inteligente engloba e suas aplicações, fez-se necessária a realização de um estudo que analisasse as publicações científicas relevantes para a pesquisa, a fim de verificar os conhecimentos e pesquisas já realizadas sobre o tema. Assim, para conhecer os trabalhos já realizados, as metodologias, tecnologias e ferramentas utilizadas, e identificar as lacunas existentes na literatura sobre o tema em questão, uma análise bibliométrica e sistêmica foi conduzida.

Diversos autores buscaram entender como os conceitos da agricultura inteligente estão sendo aplicados e quais as tecnologias estão envolvidas. Alguns trabalhos encontrados na literatura propõem soluções baseadas no desenvolvimento de plataformas IoT como o estudo de Abhijith et al. (2017) que buscaram integrar IoT na agricultura. Os autores consideraram uma plantação de cana-de-açúcar como objeto de estudo e buscaram comparar parâmetros ótimos referentes ao solo e a fatores ambientais que devem ser considerados para garantir rendimento máximo da plantação e compará-los com os dados capturados.

Outros autores que buscaram criar soluções baseadas no estudo de fatores ambientais para melhorar o rendimento das culturas. Dewi e Chen (2020), Dan et al. (2017) e Elijah et al. (2018) implementaram tecnologia IoT para capturar dados para o monitoramento de culturas e solo, monitoramento do clima e do ar, bem como o monitoramento do trabalho de máquinas. Segundo Madushanki et al. (2019), a maioria das pesquisas que envolvem aplicações IoT estão voltadas para a gestão da água nas fazendas, devido à falta de sua abundância.

Kamilaris et al. (2016) criaram uma plataforma *online* personalizável, baseada em IoT, a qual permitiu o processamento de dados em larga escala, análise e interpretação de dados provenientes de diversas fontes, como sistemas sensoriais, câmeras de vigilância, previsão do tempo, e informações, avisos e alertas de organizações governamentais. Ainda que a plataforma busque integrar dados provenientes de diversas fontes, ela não é flexível e adaptável a todos os cenários de agricultura, e trata apenas de dados provenientes de fatores ambientais.

Apesar de haver estudos para o desenvolvimento de plataformas IoT voltadas para a agricultura, eles são específicos para determinados cultivos e informações. Assim, faz-se

necessária a criação de plataformas genéricas que possam dar suporte a qualquer tipo de fazenda, independente do seu cultivo, sendo facilmente customizáveis e livres de restrições geográficas (ELIJAH et al., 2018). Ainda, de acordo com os autores, há a necessidade de desenvolver dispositivos IoT que levem em consideração algoritmos com criptografia avançada, para aumentar a segurança dos dados capturados.

Podendo ser considerada a solução para várias aplicações, tecnologias de *Big Data* podem ser utilizadas em tomada de decisão, e para extrair novas ideias e conhecimentos para a agricultura. Os principais desafios são descobrir conhecimentos e correlações de registros históricos e lidar com dados não estruturados (BENDRE; THOOL; THOOL, 2016). Em sua pesquisa, Bendre, Thool e Thool (2016) objetivam, através da utilização de diferentes conjuntos de dados, melhorar a precisão das previsões das chuvas, utilizando diferentes parâmetros.

As aplicações de *Big Data* na agricultura não são estritamente relacionadas à produção primária, mas desempenham um papel importante na melhoria da eficiência de toda a cadeia de suprimentos para minimizar custos de produção (RAJESWARI; SUTHENDRAN; RAJAKUMAR, 2018; WOLFERT et al., 2017). Rajeswari, Suthendran e Rajakumar (2018) utilizaram técnicas de análise preditiva de *Big Data* para analisar e prever o rendimento da colheita e decidir a melhor sequência de colheita com base em informações de rendimento de culturas anteriores na mesma terra, informações atuais sobre nutrientes do solo, bem como prever o custo envolvido com fertilizantes.

Agricultores estão interessados em modelos de negócios que deem suporte a geração de receita a partir de dados capturados através de tecnologias IoT (ELIJAH et al., 2018). A partir dos dados coletados e gerados pelos diversos dispositivos e tecnologias de comunicação, é possível construir bases de conhecimento que armazenem informações complexas e não estruturadas. No entanto, contar com um modelo que disponha de informações de maneira organizada e completa, que forneçam conhecimentos relevantes, que possa ser utilizado de maneira universal e que implemente de fato a agricultura inteligente, ainda é um desafio (MOHANRAJ; ASHOKUMAR; NAREN, 2016; SHAMSHIRI et al., 2018).

Observando-se os trabalhos propostos e as lacunas de pesquisa apresentadas pela literatura, pode-se perceber a inexistência de trabalhos voltados para a área agrícola que considerem a criação de um modelo que caracterize a Agricultura 4.0. Também pode-se notar a ausência de um modelo que retrate todos os agentes de uma fazenda orientados a Agricultura 4.0, com todas as tecnologias e relações entre eles, de maneira integrada e

universal. Além disso, identifica-se a necessidade da criação de um modelo de informação ao longo de um cenário agrícola genérico possibilitando a análise de dados e a predição de eventos para antecipar tomadas de decisão.

Dado o problema, a seguinte pergunta de pesquisa se coloca: **como a criação de um modelo sistêmico de fazenda 4.0 poderia ser utilizado para facilitar a adoção dos conceitos da quarta revolução industrial no contexto agrícola, de tal forma que decisões pudessem ser tomadas de maneira mais ágil e preditiva, antecipando assim, tomadas de decisão?**

Esta pesquisa faz parte do Programa de Manufatura Inteligente, integrante do Programa de Pós-Graduação em Engenharia Mecânica e de Materiais da Universidade Tecnológica Federal do Paraná. Esse programa corresponde a um grupo de pesquisa focado no desenvolvimento de possíveis soluções baseadas em sistemas de operação baseados em modelos, dentre outras demandas, em regime de parceria com empresas brasileiras e com aplicação direta em seus respectivos contextos, com a finalidade última de melhorar suas condições de competitividade no mercado. Essa demanda busca caracterizar completamente processos e equipamentos, capturar todas as operações críticas, integrar funções e implementar a automatização de tomadas de decisão.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho consiste em desenvolver e aplicar um modelo sistêmico de três níveis: estratégico, tático e operacional, que permita a implantação do conceito de Agricultura 4.0 e respectivas tecnologias habilitadoras em um cenário de manutenção de equipamentos utilizados em culturas de cana-de-açúcar.

1.1.2 Objetivos Específicos

Para alcançar o objetivo geral proposto, o estudo deverá satisfazer os seguintes objetivos específicos:

O1. Identificar o contexto e entender os princípios da Agricultura 4.0 e a relação entre as tecnologias envolvidas e a Indústria 4.0;

O2. Propor um modelo que proporcione uma visão estratégica da Agricultura 4.0 (decisões estratégicas);

O3. Construir um modelo sistêmico que retrata uma fazenda genérica orientada à Agricultura 4.0, representando todos seus agentes, todas as tecnologias envolvidas e suas relações (decisões táticas);

O4. Selecionar um dos agentes do modelo sistêmico da fazenda genérica;

O5. Criar um modelo de informação de um dos agentes da uma fazenda orientada à Agricultura 4.0, através de sistemas dinâmicos, a fim de possibilitar a análise de dados e a predição de eventos para antecipar tomadas de decisão (decisões operacionais);

O6. Demonstrar a aplicação do artefato por meio de provas de conceito no contexto da empresa parceira;

O7. Avaliar o modelo proposto quanto a sua generalidade, eficiência e aplicabilidade.

1.2 Justificativa

É esperado que o papel dos humanos em analisar dados e planejar ações seja cada vez mais auxiliado por máquinas, fazendo com que o ciclo ciber físico se torne quase autônomo. Os seres humanos continuarão envolvidos nesses processos, mas em níveis cada vez mais altos de inteligência, deixando para as máquinas as atividades operacionais (WOLFERT et al., 2017). Pesquisas relacionadas à agricultura digital vêm sendo consideradas estratégicas para as organizações. Espera-se que a agricultura digital mude efetivamente a estrutura da indústria rural e aumente seu rendimento (YANE, 2010).

Empresas que fornecem produtos agrícolas vêm buscando oferecer alternativas para coletar e processar informações. Segundo Ray (2017), a tomada de decisão inteligente e automatizada está se tornando muito importante para o aumento efetivo da produtividade no domínio agrícola. Weltzien (2016) comenta a importância do desenvolvimento de sistemas que analisam e processam informações na produção de culturas. Dessa forma, a necessidade de sistemas inteligentes capazes de tomarem decisões e recomendarem as ações a serem tomadas em determinadas situações é reforçada.

Tecnologias de *Big Data* desempenham um papel essencial nesse desenvolvimento. Máquinas são equipadas com sensores que medem dados em seu ambiente e avaliam o comportamento de máquinas. No entanto, para isso, é necessário um conjunto de técnicas e tecnologias capazes de integrar todos os dados para gerar informações de valor, complexas e em larga escala (KAMILARIS et al., 2016; WOLFERT et al., 2017).

Um sistema integrado de gerenciamento permite que uma fazenda inteira seja monitorada, possibilitando evitar riscos desnecessários e a implementação de medidas preventivas que melhorem a produtividade. É possível ainda, conectar diversas fazendas em uma única plataforma, onde informações sobre produção, gestão e recomendações sejam disseminadas para maximizar a produtividade, rendimento e a receita (ELIJAH, OLAKUNLE et al., 2018).

1.3 Delimitações

Este trabalho não pretende desenvolver qualquer tipo de sensor, bem como aplicações de *software*, restringindo-se apenas a criação de um modelo sistêmico de um dos agentes de uma fazenda orientada à Agricultura 4.0, capaz de organizar e compor parte de uma camada de arquitetura de dados.

1.4 Estrutura do trabalho

O presente trabalho está estruturado em cinco capítulos. No primeiro capítulo, é apresentada uma introdução sobre o contexto de agricultura inteligente e as tecnologias envolvidas que vêm sendo utilizadas nesse cenário. Nesse capítulo, ainda são apresentados os objetivos a serem alcançados com a realização da pesquisa, a justificativa que motivou o desenvolvimento desse trabalho e suas delimitações.

Em seguida, o Capítulo 2 consiste na fundamentação teórica necessária para que o estudo seja compreendido. O capítulo seguinte apresenta os aspectos metodológicos utilizados no desenvolvimento desta pesquisa. O desenvolvimento da pesquisa encontra-se no Capítulo 4. Por fim, o Capítulo 5 apresenta as considerações finais sobre a pesquisa, resume o trabalho, expõe as dificuldades encontradas durante o desenvolvimento da pesquisa e sugere recomendações para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo, que apresenta a fundamentação teórica sobre o tema da presente pesquisa, está dividido em onze seções. A primeira e segunda seção introduzem conceitos ligados, respectivamente, a Indústria 4.0 e a Agricultura 4.0. Na sequência, a Seção 2.3 apresenta as principais tecnologias 4.0 presentes na agricultura. As seções seguintes caracterizam processos de tomada de decisão, modelagem de sistemas e ETL. A sétima seção apresenta conceitos de programação baseada em fluxos, seguida das seções 2.7 e 2.8 que apresentam a ferramenta Node-RED e o protocolo MQTT, respectivamente. A décima seção trazem conceitos sobre virtualização e conteinerização. Por fim, na Seção 2.11, destaca-se noções sobre *business intelligence*. Dessa maneira, é cumprido o objetivo de retratar todos os conceitos e definições necessários para a compreensão do presente trabalho.

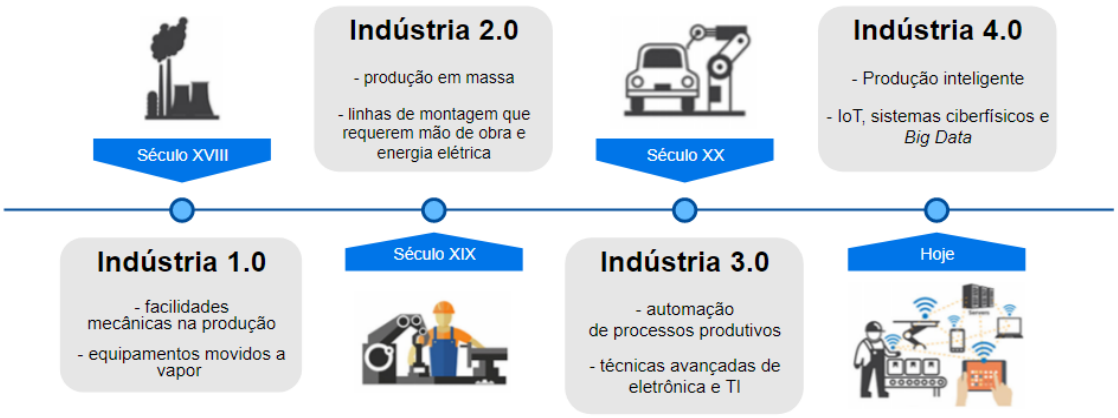
2.1 Indústria 4.0

Todas as revoluções industriais foram causadas por mudanças tecnológicas, sociais e econômicas. Desde a primeira revolução industrial, o setor vem incorporando mais tecnologias ao processo (SANDERS; ELANGESWARAN; WULFSBERG, 2016). Assim, o rápido desenvolvimento de novas tecnologias proporcionou o surgimento da quarta revolução industrial, também chamada de Indústria 4.0 (NIKOLIC et al., 2017).

A quarta revolução industrial foi precedida por outras três revoluções. A primeira revolução industrial deu-se com a introdução de facilidades mecânicas na produção, sendo sucedida pela segunda revolução através da adoção da eletricidade e da divisão do trabalho na indústria. A terceira revolução industrial, ou revolução digital, por sua vez, possibilitou a automação de processos produtivos com a incorporação de técnicas avançadas de eletrônica e de tecnologia da informação (HERMANN; PENTEK; OTTO, 2016).

A Indústria 4.0 ou indústria inteligente, incorpora os últimos avanços das tecnologias de informação e comunicação, além de tecnologia industrial, visando a melhoria da produtividade e eficiência do setor industrial (POSADA et al., 2015). A indústria inteligente está focada na melhoria contínua da eficiência, segurança, produtividade das operações e no retorno de investimento. Para isso, conta com o apoio de diversas tecnologias, também chamadas de pilares da indústria inteligente (i.e., internet das coisas, sistemas ciber físicos e *Big Data*) (COELHO, 2016). A Figura 1 apresenta os progressos ocorridos nas indústrias decorrentes das principais revoluções industriais.

Figura 1 - Perspectiva histórica dos avanços na indústria



Fonte: Adaptado de Oztemel e Gursev (2020)

Kamble et al. (2018), consideram a Indústria 4.0 uma revolução na fabricação que apresenta uma nova perspectiva para a indústria de como combinar novas tecnologias para se obter rendimento máximo com utilização mínima de recursos. Esse movimento vem fazendo com que as empresas repensem a forma como gerenciam seus negócios e processos, permitindo o planejamento e controle de produção em tempo real (SANDERS et al., 2016). Para a implementação da Indústria 4.0 são necessárias integrações verticais e horizontais de todas as suas funções principais, fazendo com que surja uma nova maneira das empresas se posicionarem na cadeia de valor e de gerenciarem sua produção. Essa integração combinada com o surgimento de novas tecnologias permite que sistemas produtivos sejam mais eficientes devido ao fato de serem mais responsivos e preditivos. Assim, a Indústria 4.0 não é somente um movimento de digitalização, mas sim uma combinação de diversas tecnologias baseadas em inovação (COELHO, 2016).

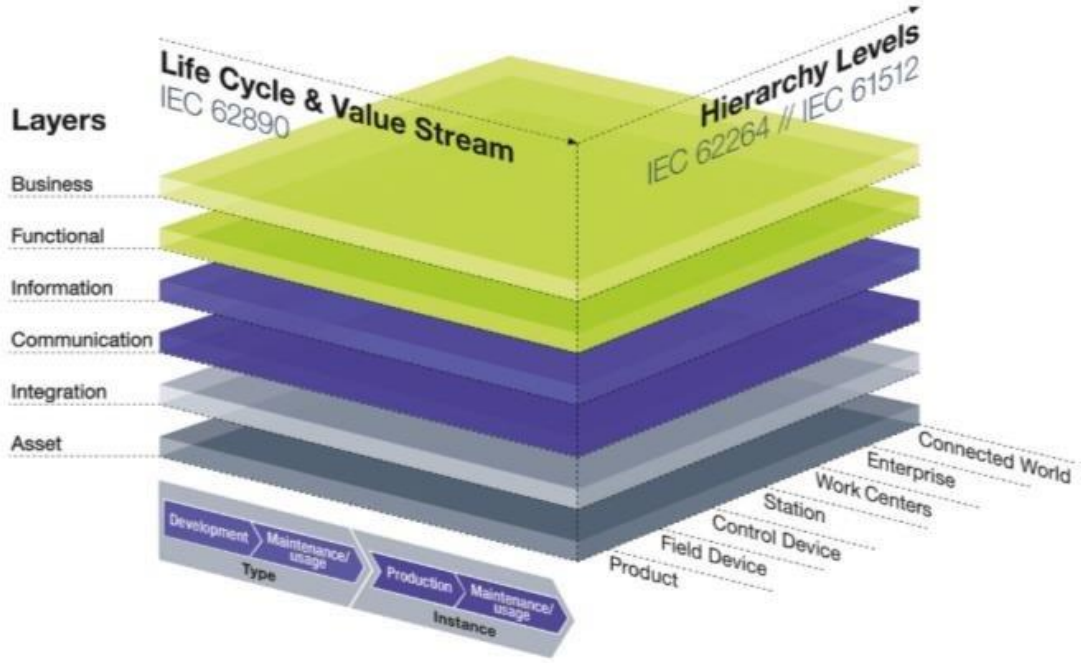
Vários modelos de referência representam a indústria 4.0, como *Industrial Internet Reference Architecture* (IIRA), *Stuttgart IT - Architecture for Manufacturing* (SITAM), *Reference Architectural Model Industry 4.0* (RAMI 4.0), entre outros. Modelos de referência são conjuntos de conceitos interconectados e claramente definidos desenvolvidos por especialistas para entender as interações entre entidades em um ambiente (PISCHING et al., 2018).

Dentre os modelos de referência de arquitetura existentes para a indústria 4.0, o mais conhecido e utilizado é o RAMI 4.0 (RESMAN et al., 2019).

2.1.1 *The Reference Architectural Model Industry* 4.0 (RAMI 4.0)

O RAMI 4.0 visa agrupar e representar diferentes aspectos da indústria em um modelo único, descrevendo os aspectos mais importantes dos sistemas de produção inteligentes com uma visão arquitetônica da Indústria 4.0 (ALAM; EL SADDIK, 2017; RESMAN et al., 2019; ROJKO, 2017). Segundo Pisching et al. (2018), este modelo fornece uma referência para um sistema de arquitetura básica 4.0, servindo como base para a análise de seus relacionamentos e detalhes. Ilustrado na Figura 2, este modelo, desenvolvido na Alemanha (RESMAN et al., 2019), é uma adaptação e expansão do *Smart Grid Architecture Model* (SGAM) para atender aos requisitos 4.0 (PISCHING et al., 2018).

Figura 2 - RAMI 4.0



Fonte: Suakanto et al. (2017)

O RAMI 4.0 é um modelo tridimensional desenvolvido com base na integração horizontal e vertical e na engenharia de ponta a ponta. O eixo horizontal (à direita da Figura 2) representa os níveis hierárquicos, ou seja, as camadas de integração de um sistema de controle empresarial (i.e., empresa, centros de trabalho, estação e dispositivo de controle) e outras três camadas para dar suporte ao conceito inteligente. A camada *Field Device* permite controlar equipamentos e sistemas de forma inteligente (i.e., utilização de sensores). A

camada *Product* representa a interdependência entre os produtos a serem fabricados e os equipamentos de produção. Já a camada *Connected World* permite a representação da expansão dos limites fabris, com a instituição de parcerias e redes colaborativas (BEDENBENDER et al., 2015; PISCHING et al., 2018; ROJKO, 2017).

O eixo horizontal esquerdo - *Life Cycle and Value Stream* - mostra o ciclo de produção e o ciclo de vida do produto (ROJKO, 2017). Por fim, o eixo vertical (*Layers*) representa as camadas hierárquicas de informação, decompondo e representando perspectivas (i.e., mapas de dados, comportamentos) de entidades físicas. Assim, define a estrutura das tecnologias de informação e comunicação para a indústria 4.0 (PISCHING et al., 2018; ROJKO, 2017).

Assim, segundo Bedenbender et al. (2015), o RAMI 4.0 integra conceitos vertical e horizontalmente, engenharia de ponta a ponta e ciclo de vida. Além disso, o modelo permite a divisão dos processos em pacotes, facilitando a comunicação e o processamento dos dados. Este modelo subdivide um sistema complexo com conexões internas em subsistemas menores e mais fáceis de lidar (RESMAN et al., 2019).

A combinação de tecnologias resulta no desenvolvimento de fábricas inteligentes que utilizam recursos de forma altamente eficiente e podem se adaptar rapidamente a diferentes cenários de mercado (KAMBLE; GUNASEKARAN; GAWANKAR, 2018). Nesse contexto, as mudanças e tecnologias presentes na Indústria 4.0 influenciam na forma como as empresas do setor agrícola caminham no mesmo sentido. Assim, com grande potencial de crescimento e produtividade, o setor agrícola vem adotando e adaptando essas mesmas tecnologias, dando origem à chamada Agricultura 4.0 (HUSTI; DAROCZI; KOVACS, 2017)

2.2 Agricultura 4.0

A agricultura tradicional baseada em habilidades está se transformando rapidamente em agricultura digital, baseando-se no conhecimento, com a utilização de *Big Data*, e desempenhando um papel primordial no aumento da produtividade do campo (HIMESH et al., 2018).

As revoluções na agricultura andaram de mãos dadas com as inovações industriais. A revolução agrícola iniciou-se com a Agricultura 1.0, no início do século XX, através de um sistema de trabalho intensivo, o qual utilizava potência animal. Esse sistema foi capaz de alimentar a população, apesar de apresentar baixa produtividade, no entanto havia um grande

número de pequenas propriedades e um terço da população era ativa no processo de produção agrícola (HUSTI et al., 2017; KOVÁCS; HUSTI, 2018).

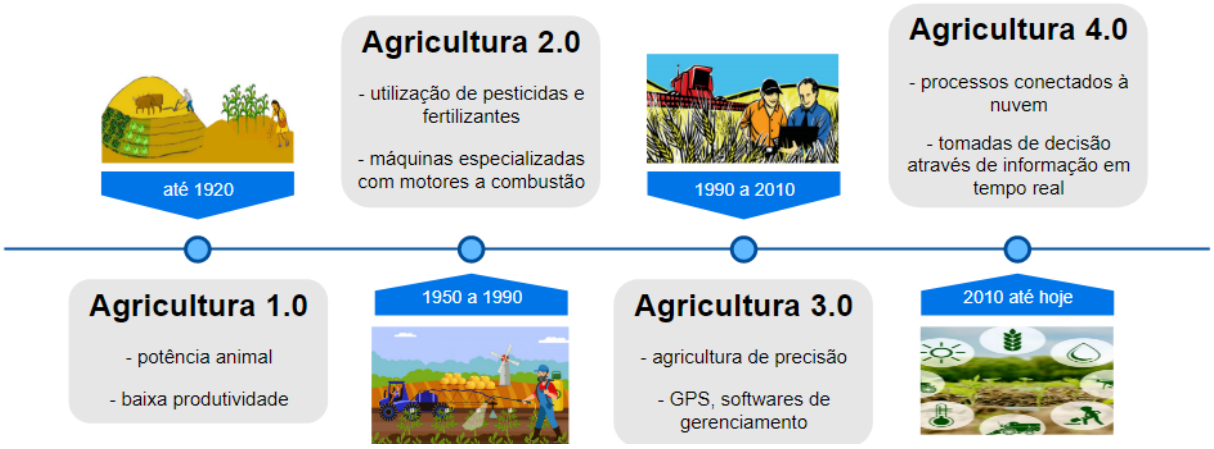
A Agricultura 1.0 foi sucedida pela Agricultura 2.0, no final da década de 1950, com a utilização de novas práticas de gerenciamento e novas ferramentas como pesticidas, fertilizantes, máquinas especializadas e mais eficientes com motores a combustão. Assim, através desses avanços foi possível aumentar drasticamente o rendimento das plantações. Nos últimos anos, a Agricultura 3.0 surgiu com os sistemas de orientação e a agricultura de precisão (HUH; KIM, 2018; KOVÁCS; HUSTI, 2018; STROZZI et al., 2017; ZAMBON et al., 2019).

A Agricultura 3.0, também chamada de agricultura de precisão, teve início quando os sinais GPS militares foram disponibilizados para utilização da população, em meados de 1990. Assim, no final dos anos 90 soluções de direção automática e com maior precisão da orientação possibilitaram o desenvolvimento de equipamentos dotados de monitoramento de rendimento com base na localização do GPS. Além disso, foi nessa época também que inspirada na indústria de transporte, passou-se a utilizar a telemática para monitorar e otimizar os processos logísticos nas fazendas. Ainda durante a chamada Agricultura 3.0, *softwares* de gerenciamento de dados agrícolas tornaram-se amplamente disponíveis. Assim, passou-se a focar ainda mais na redução de custos e aumento da rentabilidade das fazendas (HUH; KIM, 2018; KOVÁCS; HUSTI, 2018; ZAMBON et al., 2019).

Atualmente, a Agricultura 4.0, assim como a Indústria 4.0, está presente com todos os processos conectados à nuvem. Assim, a Agricultura 4.0, também chamada de agricultura inteligente ou agricultura digital, é definida como a Indústria 4.0 do setor primário (STROZZI et al., 2017; HUH; KIM, 2018; KOVÁCS; HUSTI, 2018; ZAMBON et al., 2019).

O desenvolvimento de tecnologias como sistemas e transmissão de dados, gerou mudanças radicais no ambiente de trabalho agrícola nos últimos anos. Essas mudanças fazem com que informações atualizadas dos sistemas, dos mercados e agentes envolvidos na produção sejam necessárias para fornecer informações para tomadas de decisão ligadas à produção e a questões estratégicas e gerenciais (PIVOTO et al., 2018). A Figura 3 apresenta os progressos ocorridos nas indústrias decorrentes das principais revoluções industriais.

Figura 3 - Perspectiva histórica dos avanços na agricultura



Fonte: a própria autora

Sensores mais baratos e aprimorados, comunicação em banda larga, sistemas de informação e comunicação baseados em nuvem e análises de *Big Data* proporcionaram que a partir do ano de 2010 tecnologias inteligentes estivessem cada vez mais adaptadas e presentes em tratores, colheitadeiras e outros equipamentos agrícolas. Computadores de bordo, sensores para acompanhamento da operação das máquinas e do processo agronômico, recursos de automação dos processos de plantação e pulverização, e a telemática incorporada aos veículos são alguns dos exemplos da evolução que ocorreu na agricultura, paralelamente à quarta revolução industrial (KOVÁCS; HUSTI, 2018).

Em virtude da grande dependência do clima e das condições ambientais (i.e., chuvas, temperatura, umidade), eventos imprevisíveis (i.e., doenças animais, pragas), e da volatilidade dos preços do mercado, a agricultura é considerada altamente imprevisível. Por esse motivo, há uma grande necessidade de serem utilizadas tecnologias e análise de dados que ajudem na informação de agricultores sobre as condições e riscos de suas fazendas, a fim de se tomar medidas adequadas e em tempo hábil para proteger suas culturas (KAMILARIS et al., 2017).

Com o surgimento de máquinas e sensores inteligentes, a quantidade e tipos de dados cresceu vertiginosamente. Os processos agrícolas, por sua vez, estão cada vez mais orientados por esses dados (WOLFERT et al., 2017). Soluções tecnológicas (i.e., *Bluetooth*, Sistema de Posicionamento Global – GPS, identificação por radiofrequência – RFID) combinadas com a comunicação entre operadores e máquinas agrícolas em todos os níveis de colaboração tornam viável a criação de uma estrutura auto otimizável (ZAMBON et al., 2019). Segundo Zamora-Izquierdo et al. (2019), os sensores atuais oferecem dados muito precisos, e os atuadores são capazes de realizar o gerenciamento da irrigação, alterar fatores climáticos, ou até mesmo, enriquecer o solo com os nutrientes necessários.

Nos últimos anos houve um aumento da capacidade de controle e monitoramento de sistemas agronômicos devido o uso de tecnologias de informação e comunicação, assim como na indústria, o que permitiu o surgimento da agricultura digital (ZAMORA-IZQUIERDO et al., 2019). É importante ainda ressaltar que apesar da agricultura inteligente e a agricultura de precisão muitas vezes serem tratadas como algo semelhante, na verdade, são bastante diferentes (SUAKANTO et al., 2017). A agricultura de precisão permitiu o surgimento da agricultura inteligente, que trata da coleta, processamento e análise de dados em tempo real, bem como de tecnologias de automação dos procedimentos agrícolas, permitindo, assim, a melhoria das operações, gestão e tomadas de decisão (KAMILARIS et al., 2017).

Ao contrário da agricultura de precisão, que leva em consideração apenas a variabilidade do campo, a agricultura inteligente vai além, incluindo todo o ciclo de produção agrícola, objetivando aprimorar a consciência do contexto e da situação, acionada por eventos em tempo real (HIMESH et al., 2018). Através da combinação de tecnologias da agricultura digital e da agricultura de precisão será possível reduzir custos e maximizar rendimentos e lucros (SUAKANTO et al., 2017).

De acordo com Rotz et al. (2019), a agricultura digital é um meio para enfrentar desafios como mudanças climáticas, escassez de água e problemas ambientais, os quais tornam a produção de alimentos mais difícil e cara. A tomada de decisão e o planejamento agrícola podem ser apoiados ainda mais, levando a uma agricultura mais eficaz e eficiente (KÖKSAL; TEKINERDOGAN, 2019).

Agricultura 4.0 significa a utilização de tecnologias para aumentar a lucratividade e a sustentabilidade na agricultura (OZDOGAN et al., 2017). Wolfert et al. (2017) afirmam que a agricultura digital vem sendo impulsionada pelo rápido desenvolvimento da computação em nuvem e da internet das coisas, utilizando a tecnologia da informação e comunicação no gerenciamento ciber físico das fazendas. Pode-se dizer ainda que a Agricultura 4.0 é a aplicação de sistemas de tecnologia de grande volume de dados e precisão na agricultura (OZDOGAN et al., 2017; ROTZ et al., 2019).

Diversas melhorias foram proporcionadas pela agricultura digital, como o aumento de eficiência de fazendas e da qualidade da colheita, minimização de riscos, redução de custos e aumento da lucratividade, e melhoria na gestão de tomadas de decisão (KÖKSAL; TEKINERDOGAN, 2019).

A agricultura inteligente representa a aplicação de modernas tecnologias de informação e comunicação (ICT) na agricultura com o objetivo de aumentar a produção e o retorno financeiro, e ainda, reduzir o impacto causado ao meio ambiente (RAINS; THOMAS,

2009). De acordo com (KÖKSAL; TEKINERDOGAN, 2019) a agricultura inteligente baseia-se na utilização de tecnologias como computação em nuvem, sensoriamento remoto, internet das coisas, agricultura orientada por dados e análise de *Big Data*.

2.3 Tecnologias 4.0 na agricultura

Avanços tecnológicos vêm impulsionando mudanças drásticas nas indústrias desde que as revoluções industriais se iniciaram (RÜSSMANN et al., 2015). Diversas tecnologias estão disponíveis para que a Indústria 4.0 atinja seu objetivo relacionado a melhoria contínua, maior eficiência, segurança e produtividade de operações (RÜSSMANN et al., 2015; COELHO, 2016).

Nove avanços tecnológicos, também chamados de nove pilares, são considerados fundamentais para a Indústria 4.0. Eles são responsáveis por transformar a produção em uma produção totalmente integrada, automatizada e otimizada. Além disso, essas tecnologias estão sendo responsáveis pela mudança no relacionamento entre produção, fornecedores e clientes (ENNIS et al., 2018; ROJKO, 2017; RÜSSMANN et al., 2015).

Assim como na indústria, o setor agrícola está sendo transformado com a utilização dessas novas tecnologias. A Agricultura 4.0 oferece novas oportunidades por meio da disponibilidade de tecnologias interconectadas e intensivas, que surgem a partir da Indústria 4.0 (KOVÁCS; HUSTI, 2018).

Mudanças radicais na execução das operações ocorreram com a utilização de Sistemas Ciber Físicos (do inglês *Cyber-Physical Systems – CPS*), os quais integram e permitem a interação entre computadores, redes de comunicação e processos físicos (COELHO, 2016). Através da utilização de CPS, o planejamento dos planos de produção, que eram baseados em previsões, passaram a poder ser realizados em tempo real e a serem auto otimizáveis. A introdução de sistemas de informação e comunicação também levou a um aumento no grau de automação, possibilitando sincronizar toda a linha de produção das empresas com a cadeia de valor (SANDERS et al., 2016).

Atualmente, a internet desempenha um papel vital em todas as áreas (RAJESWARI; SUTHENDRAN; RAJAKUMAR, 2018). A IoT visa conectar objetos físicos à internet usando conectividade de rede incorporada a sensores (NUKALA et al., 2016). De acordo com Rüssmann et al. (2015) a IoT tem como principal função conectar objetos físicos à internet visando a coleta de dados. Por meio desta coleta, computadores e dispositivos podem auxiliar em tomadas de decisão sobre operações. Assim, com seu uso, operações de negócios vem se

tornando mais ágeis e integradas, e consequentemente aumentam a competitividade das empresas.

Na indústria IoT possibilita a criação de um ambiente inteligente através da conexão entre todos os equipamentos das fábricas. A integração de informações entre máquinas inteligentes, sistemas de armazenagem e instalações fabris permite a comunicação e troca de informações de ponta a ponta no processo produtivo das empresas (SANDERS et al., 2016).

Assim como nas indústrias, no contexto da agricultura digital, várias aplicações vêm utilizando IoT para monitorar o crescimento das plantações, para selecionar os fertilizantes mais adequados, bem como em sistemas que apoiem decisões ligadas à irrigação. Os dispositivos IoT fornecem informações precisas sobre diversos parâmetros para melhorar os métodos de cultivo (i.e., fatores ambientais, solo, equipamentos agrícolas, irrigação, pragas e fertilizantes), podendo ser usadas para determinar o momento ótimo de colheita, qual cultivo é mais adequado para determinada localização, detecção de pragas e controle de maquinários (MUANGPRATHUB et al., 2019; NUKALA et al., 2016; SHENOY; PINGLE, 2016). Assim, os dispositivos IoT são utilizados para detectar os dados agrícolas, os quais são armazenados em bancos de dados em nuvem (RAJESWARI; SUTHENDRAN; RAJAKUMAR, 2018).

Já a computação em nuvem, do inglês *cloud computing*, permite o armazenamento e o compartilhamento de dados em larga escala a baixo custo, sendo utilizado em conjunto com IoT, contando ainda com alocação de memória sob demanda (DONZIA; KIM; HWANG, 2019; RAJESWARI; SUTHENDRAN; RAJAKUMAR, 2018). A utilização da IoT, computação em nuvem e a utilização de sensores acarreta a geração de um grande volume de dados chamados de *Big Data* (ELIJAH et al., 2018; MUANGPRATHUB et al., 2019).

Big Data refere-se a quantidades muito grandes de dados que são gerados em tempo real por sistemas ligados a rede (IoT) (COELHO, 2016; RAJESWARI; SUTHENDRAN; RAJAKUMAR, 2018). Wolfert et al. (2017) definem *Big Data* como conjuntos de dados muito grande e complexos que não seria possível processá-los com métodos tradicionais. Seu processamento necessita de técnicas e tecnologias capazes de integrar e revelar padrões ocultos nesses conjuntos de dados complexos e em grande escala. Dessa forma, *Big Data* vem sendo utilizado para fornecer informações preditivas em operações agrícolas, conduzir decisões em tempo real e redesenhar processos de negócios (ELIJAH et al., 2018; ZHOU et al., 2018).

Outro elemento chave da agricultura inteligente é o sistema de informações de gerenciamento agrícola, do inglês *farm management information systems* (FMIS), o qual é utilizado para coletar e processar os dados utilizados na gestão das operações das fazendas.

Esse tipo de sistema dá suporte à automatização da aquisição de dados, processamento, monitoramento, planejamento, tomadas de decisão, documentação e gestão das operações (KÖKSAL; TEKINERDOGAN, 2019).

2.4 Processos de tomada de decisão

Segundo Chiavenato (2003), as decisões tomadas em uma organização são realizadas de acordo com a percepção das pessoas diante das situações. Para Vercellis (2009), tomada de decisão é o processo no qual um indivíduo busca solucionar uma diferença existente entre o estado operacional atual de um sistema e suas condições desejadas.

As decisões devem ser tomadas levando em consideração a maior quantidade de informações referentes ao processo. Assim, deve-se estudar o problema a partir de dados levantados, estabelecer possíveis soluções, escolher a melhor solução, viabilizá-la e então, implementá-la e analisar os resultados obtidos (CHIAVENATO, 2007; GUIMARÃES; ÉVORA, 2004).

De acordo com Chiavenato (2007), seis elementos básicos estão envolvidos nos processos de tomada de decisão:

(i) o tomador de decisão;
(ii) os objetivos a serem alcançados;
(iii) critérios de escolha;
(iv) estratégia adotada e sequência de ações;
(v) aspectos ambientais vivenciados pelo tomador de decisão; e,
(vi) o resultado proveniente da estratégia adotada.

As decisões ainda podem ser relacionadas aos níveis organizacionais, sendo elas decisões estratégicas, as quais afetam a organização como um todo, decisões táticas, as quais estão focadas na gestão de processos de uma parte da organização, e decisões operacionais, as quais referem-se a atividades específicas (VERCELLIS, 2009).

Devido ao cenário competitivo que as organizações enfrentam, se faz necessário tomar decisões de maneira rápida, segura e estratégica para obter vantagem no mercado. Para isso é necessário contar com sistemas que apoiem a análise de dados e as tomadas de decisão (RAUTENBERG; DO CARMO, 2019).

2.5 Modelagem de sistemas

Modelagem é o processo de produzir um modelo, ou seja, gerar uma representação da estrutura e funcionamento de algum sistema de interesse. Um modelo tem como objetivo permitir análises para prever efeitos de mudanças em um sistema, no entanto é mais simples do que o sistema que ele representa (MARIA, 1997).

Segundo Traoré (2019), modelos são criados para dar suporte ao projeto, análise, verificação e validação de projetos e requisitos de sistemas. Os modelos fornecem perspectivas formalizadas sobre o cenário o qual representam, expressas de forma a possibilitar a coleta de informações específicas sobre o assunto. Dessa maneira, seu objetivo é tornar possível entender, verificar, construir, comunicar e aplicar parte do mundo real (FLANDERS; JANNIDIS, 2015).

Já o termo sistema refere-se a um conjunto de entidades, que pressupõem interação de causa e efeito entre as partes que os compõem, e que atuam com o mesmo fim. Assim, pode-se definir modelagem de sistemas como sendo um processo de apoio a tomadas de decisão que necessitam análises mais robustas e elaboradas para solucionar um determinado problema (MILLER; KOSSIK; VOSS, 2003).

Os sistemas podem ser analisados por meio de modelos reais, os chamados modelos físicos, os quais podem ser protótipos do objeto de estudo, ou por meio de modelos que representem o sistema real. Esse tipo de modelo é denominado modelo matemático e representa o sistema por meio de relações lógicas-quantitativas, as quais são manipuladas e modificadas para realizar-se a análise do comportamento do sistema (LAW; KELTON; KELTON, 2000).

Modelos matemáticos de um sistema podem ser simulados para que seja possível avaliar mudanças e seus impactos no sistema antes de uma tomada de decisão. Assim, pode-se considerar um modelo de simulação como um conjunto de regras que representam como ir do estado atual de um sistema para um estado futuro (GRIGORYEV, 2015).

A complexidade de um sistema depende da quantidade de relações existentes entre seus elementos, quanto mais conexões houver entre seus elementos, mais complexo será o sistema. As informações representadas no modelo e o formalismo adequado para escrever o modelo dependem de fatores como a questão que se deseja responder sobre o sistema, e as propriedades do sistema a ser representado (TRAORÉ, 2019).

O processo de modelagem de um sistema é composto pelo desenvolvimento de diferentes modelos que representam diferentes perspectivas do sistema. Os modelos podem

ser criados por meio de notações gráficas como forma de facilitar a discussão sobre um sistema, para documentar um sistema, ou ainda para descrever um sistema para que sua implementação seja possível (FLANDERS; JANNIDIS, 2015).

As linguagens gráficas de modelagem surgiram há bastante tempo no campo de desenvolvimento de *software*, uma vez que as linguagens de programação não possuíam o nível de abstração necessário para que os projetos fossem discutidos. Uma das linguagens de modelagem gráfica mais conhecida é a UML (*Unified Modeling Language*) (BOOCH, 1999).

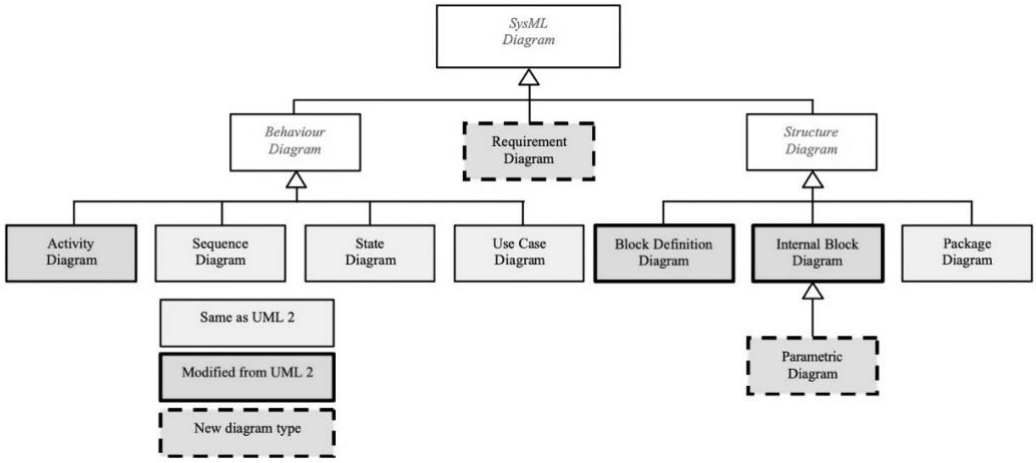
A UML surgiu da unificação de diversas linguagens gráficas, quando James Rumbaugh e Grady Booch começaram a unificar suas notações já bastante conhecidas (WAZLAWICK, 2016, p.3), em meados do ano de 1997, e tornou-se uma linguagem padrão para projetos de *software* pelo *Object Management Group* (OMG) (FOWLER, 2014, p.25; YARLAGADDA, 2016). Como a linguagem UML foi projetada para o desenvolvimento de *software*, entendendo a necessidade de modelar outros sistemas, em 2001 a OMG, juntamente com o *International Council on Systems Engineering* (INCOSE), decidiu desenvolver a *Systems Modeling Language* (SysML), um padrão de linguagem UML específica para a engenharia de sistemas (FRIEDENTHAL; MOORE; STEINER, 2014; WOLNY et al., 2020)

2.5.1 *Systems Modeling Language*

A *Systems Modeling Language* (SysML) é uma linguagem de modelagem gráfica, publicada pela OMG em 2006, que permite especificar, analisar, desenvolver e validar sistemas complexos. Esses sistemas podem ser *hardwares, softwares*, dados, procedimentos, pessoas e elementos de sistemas naturais (FRIEDENTHAL; MOORE; STEINER, 2014; HAUSE, 2006; HUANG; RAMAMURTHY; MCGINNIS, 2007).

A linguagem SysML é amplamente utilizada para dar suporte a projetos e análises de diferentes tipos de sistemas, e garantir a consistência entre o projeto do sistema e sua documentação. Isso é possível devido a SysML utilizar um modelo orientado a objetos, o que facilita a representação da arquitetura do sistema e do seu fluxo de informações. Ainda, a SysML possibilita aos usuários enxergar muitos aspectos da arquitetura de um sistema, pois fornece um conjunto de diagramas padronizados que descrevem os muitos elementos que o campo de engenharia de sistemas requer (HOSSAIN et al., 2022). Essa linguagem, para dar suporte à engenharia de sistemas, reutiliza um subconjunto de diagramas de UML e adiciona outros, conforme pode ser observado na Figura 4.

Figura 4 - Diagramas da SysML



Fonte: Hause (2006)

Os diagramas SysML são divididos em três grupos, são eles: diagrama de requisitos, diagramas comportamentais e diagramas estruturais. O diagrama de requisitos representa a relação hierárquica entre requisitos e suas relações com outros elementos do modelo (MANN, 2009; WOLNY et al., 2020).

Já os diagramas de comportamento especificam as partes dinâmicas dos elementos representados, podendo ser diagrama de atividades, de sequência, de máquina de estados e de casos de uso. O diagrama de caso de uso descreve a funcionalidade do sistema (alto nível). O diagrama de atividades, por sua vez, representa o fluxo de dados e o controle entre atividades, por último, o diagrama de sequência representa a interação entre as partes do sistema, e o diagrama de requisitos (FRIEDENTHAL; MOORE; STEINER, 2014). Por fim, os diagramas estruturais definem os elementos estruturais e estáticos, assim como os diagramas estruturais da UML, podendo ser diagrama de pacotes, de blocos de definição de blocos internos e paramétricos. (HAUSE, 2006).

Os diagramas de pacote, do inglês *package diagram*, são utilizados quando deseja-se mostrar a organização do modelo do sistema, ou seja, transmitir informações sobre a estrutura do modelo. Esses diagramas são úteis para proporcionar uma visão simplificada do modelo, uma vez que é possível representar pacotes aninhados dentro de outros pacotes para representar a hierarquia contida no modelo (DELLIGATTI, 2013 p.189; HART, 2015).

Segundo Delligatti (2013 p.190), ao criar um diagrama de pacotes é comum realizar alterações, criar pacotes e reorganizar o diagrama, uma vez que à medida que o design da estrutura do modelo vai evoluindo com o passar do tempo. Nos diagramas de pacotes é

comum utilizar o relacionamento de importação de pacote, o qual retrata que um pacote importa o conteúdo de outro. A notação é uma flecha tracejada com a palavra <<*importar*>>.

O diagrama de blocos de definição é o tipo mais comum de diagrama SysML. Ele é semelhante a um diagrama de classes tradicional, e é utilizado para descrever as relações existentes entre os blocos. Muitas vezes esse tipo de diagrama é criado em conjunto com outros diagramas SysML para fornecer uma visão complementar a algum ponto de interesse do sistema que está sendo retratado (DELLIGATTI, 2013 p.24; HART, 2015).

O diagrama retrata a estrutura do sistema, a hierarquia do sistema e os classifica, descrevendo as relações entre cada subsistema, bem como o fluxo de troca de material, energia e informação. Os elementos representados neste diagrama podem ser blocos, atores, especificações de fluxo e interfaces, e são chamados de elementos de definição, uma vez que formam a base de todo o modelo. Ainda, as relações estruturais entre os elementos presentes no diagrama de definição de blocos podem ser: associações, generalizações e dependências (DELLIGATTI, 2013 p.23).

Já o diagrama de blocos internos descreve a estrutura interna dos sistemas (i.e., partes, portas e conexões). Por sua vez, o diagrama paramétrico representa as restrições nos valores de desempenho de um sistema (i.e., confiabilidade e propriedades que dão suporte às análises de engenharia) (HAUSE, 2006).

Para realizar a modelagem de sistemas complexos utilizando o SysML é fundamental contar com o apoio de ferramentas computacionais que auxiliem na construção dos diagramas (HART, 2015). Diversas ferramentas foram criadas para auxiliar no desenvolvimento de modelos de sistemas como os *softwares Modelio System Architecture, IBM Rational Rhapsody e Enterprise Architect*.

A ferramenta *Modelio System Architecture*, desenvolvido pela empresa *ModelioSoft,* suporta a construção de todos os diagramas e conta com uma interface de usuário dedicada para cada diagrama. Permite também a rastreabilidade de informações na criação de novos elementos no modelo, facilitando o entendimento da modelagem por todos envolvidos. Não é uma ferramenta *open source*, no entanto uma versão *trial* é disponibilizada por dez dias pelo fabricante (MODELIOSOFT, 2023).

Já a ferramenta *IBM Rhapsody*, desenvolvida pela IBM, é uma ferramenta que dá suporte a análise, desenvolvimento, testes e simulações de sistemas. Uma vantagem dessa ferramenta é possuir uma interface amigável para o usuário. Assim como a ferramenta *Modelio*, esse *software* não é *open source*, porém sua versão *trial* possibilita sua utilização

durante trinta dias. Para realizar simulações dos modelos é necessário utilizá-la em conjunto com o *software Simulink* (IBM, 2023).

O *software* desenvolvido pela empresa *Sparx Systems* denominado *Enterprise Architect*, proporciona a modelagem completa do ciclo de vida para sistemas de negócios e TI, engenharia de *software* e sistemas, e desenvolvimento em tempo real e integrado. É utilizado por mais de oitocentos e cinquenta mil usuários em todo o mundo, é a principal plataforma de modelagem da *Sparx Systems*. Assim como o *software IBM Rational Rhapsody*, possibilita a modelagem, projeto e construção de sistemas utilizando a linguagem SysML. Uma vantagem dessa ferramenta é a possibilidade de gerar códigos automaticamente em Python, criar relatórios, gerenciar e simular testes. Assim como todas as outras ferramentas, ela não é *open source* e possui uma versão *trial* que fica disponível por trinta dias (SPARX SYSTEMS, 2023).

2.6 Processo ETL

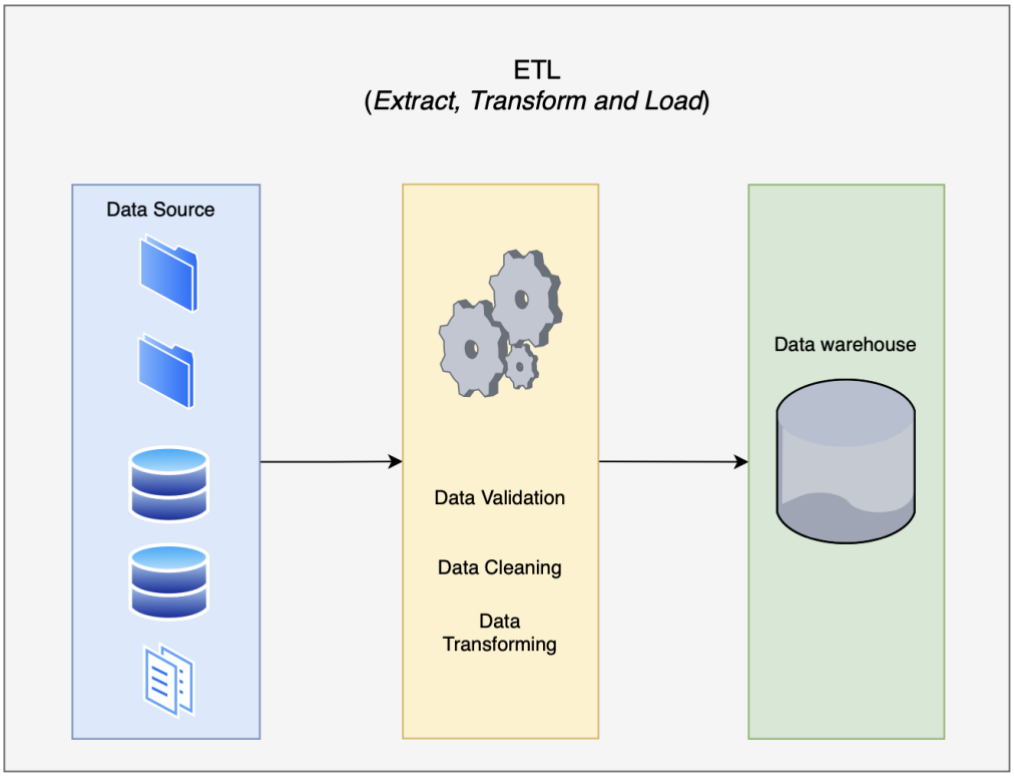
O processo ETL (*Extract-Transform-Load*) é responsável por integrar dados em um *data warehouse* (do inglês, armazém de dados). Nesse processo, dados são extraídos de diversas fontes, são transformados e carregados no armazém (DIOUF; BOLY; NDIAYE, 2018; MUÑOZ; MAZÓN; TRUJILLO, 2009).

O ETL é utilizado para integrar dados advindos de diversos domínios, extraindo esses dados, transformando-os para atender a determinadas necessidades operacionais - podendo incluir filtragem de dados e inspeção da qualidade dos dados, e carregamento desses dados em uma base de dados, e vem sendo aplicado na integração de dados na indústria (BANSAL, 2014). De acordo com Bergamaschi et al. (2011), o processo ETL apoia a identificação de informações relevantes na fonte de dados, a extração dessas informações, a customização e integração de informações advindas de diversas fontes em um formato comum, a limpeza do conjunto de dados, e a propagação dos dados para a base de dados. Ainda, segundo Vassiliadis, Simitsis e Baikousi (2009), as atividades do ETL são módulos de *software* responsáveis por popular um *data warehouse* com dados operacionais que passam por uma série de transformações até chegarem a esses armazéns de dados.

Os fluxos de trabalho do ETL são considerados complexos pelo grande volume de diferentes atividades que estão contidas nesse processo (VASSILIADIS; SIMITSIS; BAIKOUSI, 2009). O processo ETL possui três fases (Figura 5), a primeira é denominada *Extract* (do inglês, extração) e envolve a extração de dados de diversas fontes. Isso ocorre,

pois os sistemas de origens dos dados podem ter formatos diferentes de acordo com as suas necessidades operacionais (TALIB et al., 2016). Comumente os dados extraídos estão em formatos como *xls*, *txt* e *csv* (BANSAL, 2014). As empresas geralmente não possuem um documento que contém um registro formal indicando quais são as fontes de dados da empresa e onde as informações estão armazenadas, por isso é necessário analisar os processos da organização antes de iniciar essa etapa. Essa análise é realizada por pessoas que conhecem os processos da empresa e que entendem as restrições e requisitos desses dados (BERGAMASCHI et al., 2011).

Figura 5 - Processo ETL



Fonte: Adaptado de Talib et al. (2016)

Transform (do inglês, transformar) é a segunda fase do processo ETL, considerada a fase crucial do processo (TALIB et al., 2016). Segundo Bansal (2014), nessa fase é comum realizar atividades como normalização de dados, remover dados duplicados, checar a integridade dos dados, e filtrar, ordenar e agrupar dados. Atividades típicas de transformação dos dados são: transformações do esquema, atividades de limpeza, filtros, ordenamento, agrupamentos e aplicações de funções (VASSILIADIS; SIMITSIS; BAIKOUSI, 2009).

Por fim, a última fase do processo ETL, a qual envolve a propagação dos dados extraídos e transformados em bases de dados ou armazéns de dados, é chamada *Load* (do inglês, carregar) (BANSAL, 2014). Essa etapa garante que os dados sejam convertidos da estrutura de origem para a estrutura de dados de destino do armazém de dados, através de funções como agregação e instanciação (TALIB et al., 2016).

ETL é um processo longo e custoso, pois envolve muitos recursos humanos e de tecnologia de informação. No entanto, o processo ETL é um passo obrigatório nos processos de tomada de decisão (DIOUF; BOLY; NDIAYE, 2018). A velocidade desse processo vem sendo um fator decisivo para a competitividade das empresas no mercado. Para agilizar esse processo, a utilização de ferramentas que estruturam esse fluxo de informações e plataformas computação em nuvem vem sendo bastante utilizadas, uma vez que possui muitos recursos computacionais e espaço de armazenagem ilimitado (DIOUF; BOLY; NDIAYE, 2018).

2.7 Programação Baseada em Fluxo

A Programação Baseada em Fluxo - do inglês, *Flow-based Programming* (FBP) – é um paradigma de programação que foi criado por J. Paul Morrison por volta do ano de 1970, e acabou ganhando popularidade pela utilização cada vez maior de aplicações orientadas a dados, uma vez que essas aplicações podem ser criadas como uma rede de processos assíncronos que trocam e aplicam transformações sobre dados (ORTIZ et al., 2022).

Belsa et al. (2018) definem a FBP como um paradigma que estabelece aplicações como processos “caixa-preta”, que trocam dados por meio de conexões pré-definidas, sendo eles conectados para criar diferentes soluções sem a necessidade de serem modificados internamente. Ainda, para Hagino (2021, p.5) o FBP é uma mistura entre fluxo de trabalho e fluxo de dados, onde a aplicação é vista como uma rede de processos assíncronos que começam em um determinado ponto, realizam um processo sequencial único o qual realiza uma operação por vez até que esta termine.

O FBP modela sistemas com “nós” de processamento, que são executados de maneira assíncrona e se comunicam entre si através da passagem de dados entre eles (ORTIZ et al., 2022). Um nó precisa de parâmetros de entrada e oferece informações de saída que são o resultado de uma série de processos internos (BELSA et al., 2018). As tarefas são executadas em um conjunto de diversos “nós” de processamento interligados, essa ligação entre eles deve ser bem definida, criando a relação de dependências do fluxo, pois cada nó é responsável por realizar uma parte das atividades necessárias para a aplicação (ORTIZ et al., 2022).

Uma das principais vantagens de se utilizar FBP é a modularidade que esse tipo de programação permite, pois o FBP permite que os componentes de um sistema sejam combinados, separados e reutilizados apenas alterando as conexões entre os "nós" (ORTIZ et al., 2022). Além disso, aplicações FBP são executadas mais rapidamente do que os programas tradicionais, não é necessária nenhuma programação especial e otimizam o uso de todos os processadores de máquina (HAGINO, 2021, p.5). Ainda segundo o autor, esse tipo de método de programação é fácil de utilizar, cria um modelo visual e possibilita que qualquer pessoa entenda o fluxo.

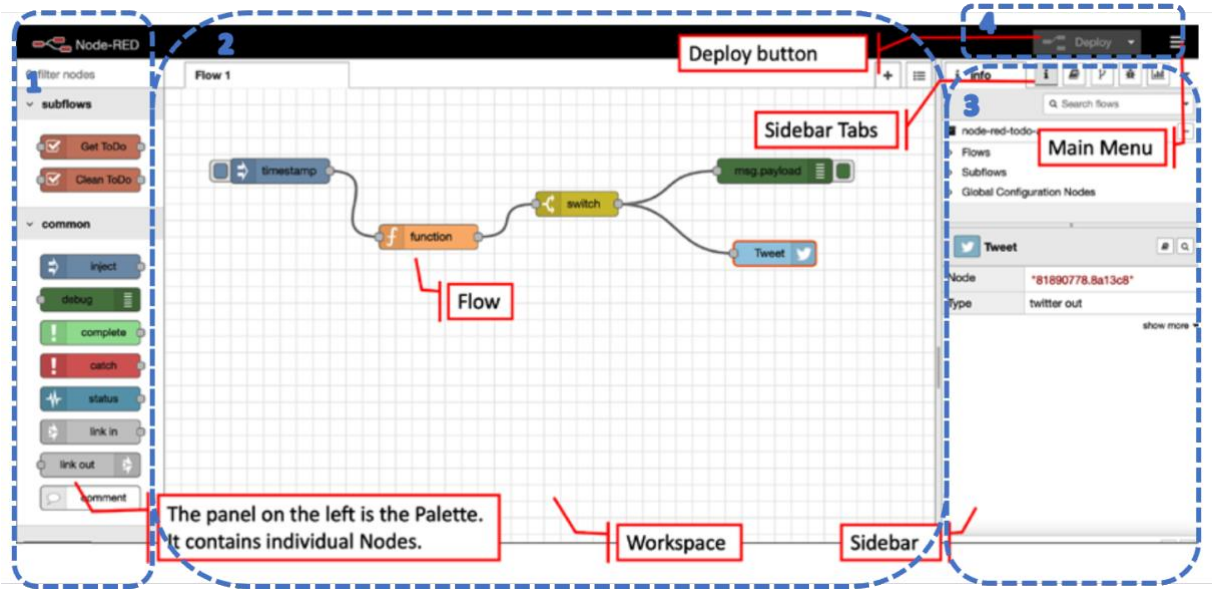
Há diversas ferramentas de FBP, uma das mais conhecidas é a IFTTT. Essa ferramenta cria fluxos baseados no conceito *If This Then That* e seu principal objetivo é criar repositórios de tarefas automáticas. A IFTT é voltada para a interligação de serviços *web*, no entanto não é capaz de gerar tarefas complexas (BELSA et al., 2018). Segundo Ortiz et al. (2022), a ferramenta de programação *flow-based* que oferece mais funcionalidades é a Node-RED.

2.8 Node-RED

Node-RED é uma aplicação *web*, *open source* criada pela IBM que oferece uma abordagem de programação visual para facilitar a criação de sistemas desenvolvidos em Node.js. Ela possui diversas funcionalidades que são definidas através de blocos, os quais trocam mensagens contendo dados através de ligações representadas por meio de fluxos (BELSA et al., 2018).

O Node-RED pode ser executado utilizando-se um navegador via *localhost* (http://localhost:1880). Neste programa utilizam-se blocos, que são conjuntos de códigos, denominados *Nodes* ("nós"), os quais criam a programação ao serem conectados por meio de fluxos. Através da utilização da sua interface é possível selecionar os "nós" que se deseja utilizar, além de ser possível instalar novos blocos compartilhados por outros desenvolvedores, criar blocos e programar blocos já existentes. A Figura 6 ilustra a interface de desenvolvimento do Node-RED, sendo (1) onde encontram-se os "nós" que podem ser utilizados para a construção dos fluxos, (2) o local onde é feito o desenvolvimento dos fluxos utilizando-se o "nós" em (1), e (3) onde as configurações são realizadas. Após sua criação, o fluxo pode ser implementado através do botão *Deploy* (4).

Figura 6 – Interface de desenvolvimento do Node-RED



Fonte: Adaptado de Hagino (p.7, 2021)

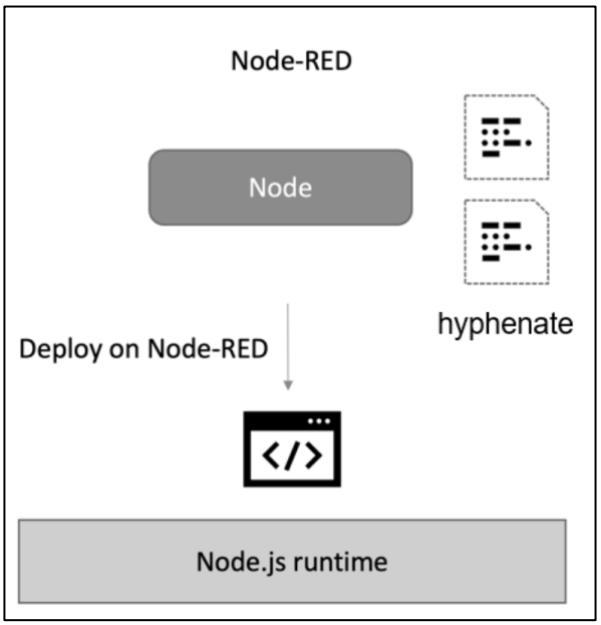
Segundo Chanthakit, Keeratiwintakorn e Rattanapoka (2019), Node-RED é uma ferramenta para desenvolvedores de programa conectarem dispositivos com API's, através do *browser*. Ela permite conectar serviços *online*, APIs, funções de código e dispositivos de *hardware* relacionados à IoT (HAGINO, 2021 p.7). Nessa ferramenta os "nós" são utilizados principalmente para processar eventos de entrada, enviar eventos de saída, manipular mensagens e *payloads*, e criar códigos que adaptem a informação de dados de saída de um nó em dados de entrada de outro (BELSA et al., 2018).

A ferramenta permite que seus usuários substituam tarefas comuns de codificação por meio de sua interface visual de arrastar e soltar (KODALI; ANJUM, 2018). Ao utilizar o Node-RED, raramente será necessário o desenvolvedor escrever um código, pois a programação pode ser feita apenas selecionando um nó e conectando-o, no entanto, funções podem ser criadas em linguagem *JavaScript* utilizando o editor de texto presente na ferramenta (CHANTHAKIT; KEERATIWINTAKORN; RATTANAPOKA, 2019). Os fluxos no Node-RED podem ser representados em diferentes abas ou páginas para auxiliar na organização do desenvolvedor, no entanto não são separados, existe apenas um fluxo para todo o sistema (BLACKSTOCK; LEA, 2014).

Normalmente o desenvolvimento de aplicações em Node.js deve-se escrever o código-fonte utilizando um editor de código, após um arquivo executável é gerado construindo o que foi criado no código-fonte. Já no Node-RED, por ser utilizada uma programação visual em

que os “nós” já correspondem a partes programadas, a codificação é realizada ao inserir os “nós” no editor, conforme apresenta a Figura 7 .

Figura 7 - Programação no Node-RED



Fonte: Adaptado de Hagino (p.55, 2021)

Por ter sido criada inicialmente para auxiliar na visualização e manipulação de tópicos de protocolo MQTT (*Message Queue Telemetry Transport*), a ferramenta Node-RED é muito útil e adequada para ser utilizada em projetos e serviços baseados em IoT (i.e., controle de dados em dispositivos e ligar dispositivos com serviços de nuvem). (HAGINO, 2021 p.11).

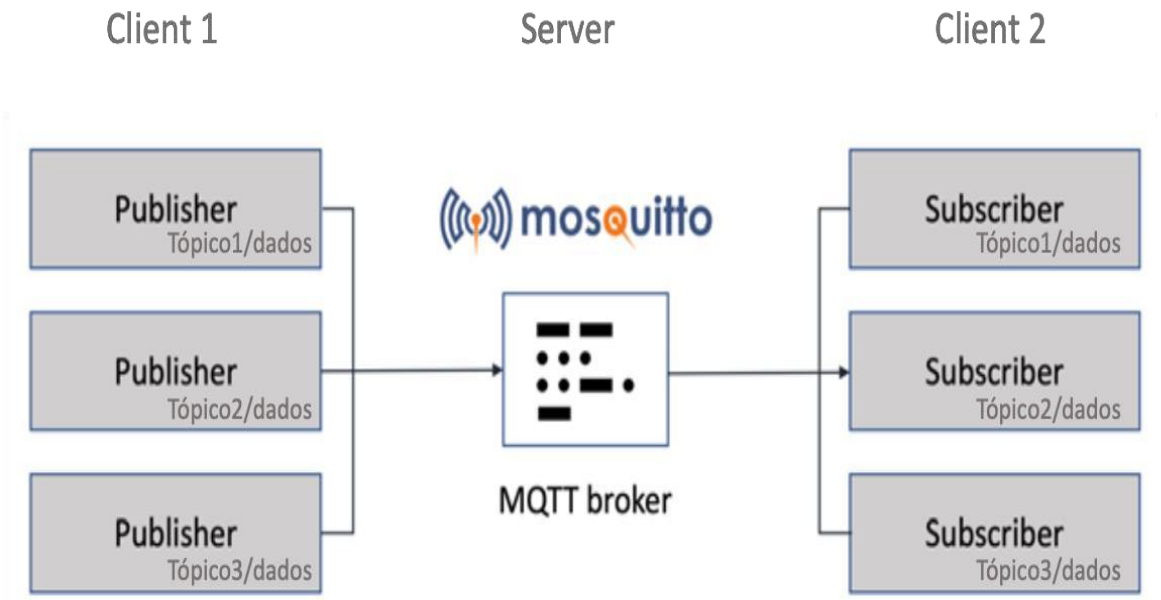
2.9 *Message Queue Telemetry Transport*

O protocolo *Message Queue Telemetry Transport* (MQTT) foi criado pela IBM em 1999. É um formato baseado no modelo *Publisher/Subscriber,* que proporciona uma comunicação confiável por apresentar nativamente três tipos de garantia de entrega de mensagem, em inglês *Quality of Service* (QoS) (LAMPKIN et al, 2012, p.212).

O protocolo MQTT é o padrão para IoT - assim como o HTTP é o protocolo padrão da *web* – e baseia-se na conexão IoT M2M (*machine-to-machine*), por realizar a transmissão de mensagens através de sensoriamento remoto e filas. Esse protocolo foi criado para transferir mensagens *request*/*response* de maneira leve, sem necessitar de muita memória ou conexão de banda larga (ALANSARI et al., 2018; HAGINO, 2021, p.204,).

O MQTT contém quatro elementos: um tópico, dois clientes e um servidor (*broker*), conforme ilustra a Figura 8. O primeiro cliente publica as informações e as envia em forma de mensagens para o segundo cliente através de um tópico, que é um canal por onde as mensagens são enviadas. O segundo cliente, por sua vez, recebe os dados no tópico específico, podendo também interagir e enviar mensagens para o primeiro cliente. O *broker* é o elemento que garante essa troca de informação entre os clientes e mantém os dispositivos conectados com a nuvem. Portanto, clientes inscritos em um tópico podem enviar e receber mensagens via *broker*. Ainda é possível observar que um *broker* pode estar conectado a diversos clientes e receber e enviar informações ao mesmo tempo (MEDINA-PÉREZ; SÁNCHEZ-RODRÍGUEZ; ALONSO-GONZÁLEZ, 2021).

Figura 8 - Exemplo de estrutura do MQTT



Fonte: Adaptado de (HAGINO, 2021)

O *broker* é o servidor por onde os dados chegam e são enviados, sendo o ponto principal de funcionamento deste protocolo. Ele pode ser tanto *online* quanto local e é o componente intermediário que repassa as informações de sensores de diferentes tópicos para seus clientes (LAMPKIN et al., 2012, p.3). O *broker* MQTT mais conhecido é o Mosquitto (https://mosquitto.org), um *broker open source*, desenvolvido pela Eclipse. É um servidor que pode ser utilizado em diversos ambientes e por diversos sistemas, sendo compatível também

com dispositivos móveis e microcontroladores (ALANSARI et al., 2018; MEDINA-PÉREZ; SÁNCHEZ-RODRÍGUEZ; ALONSO-GONZÁLEZ, 2021).

2.10 Virtualização e Containerização

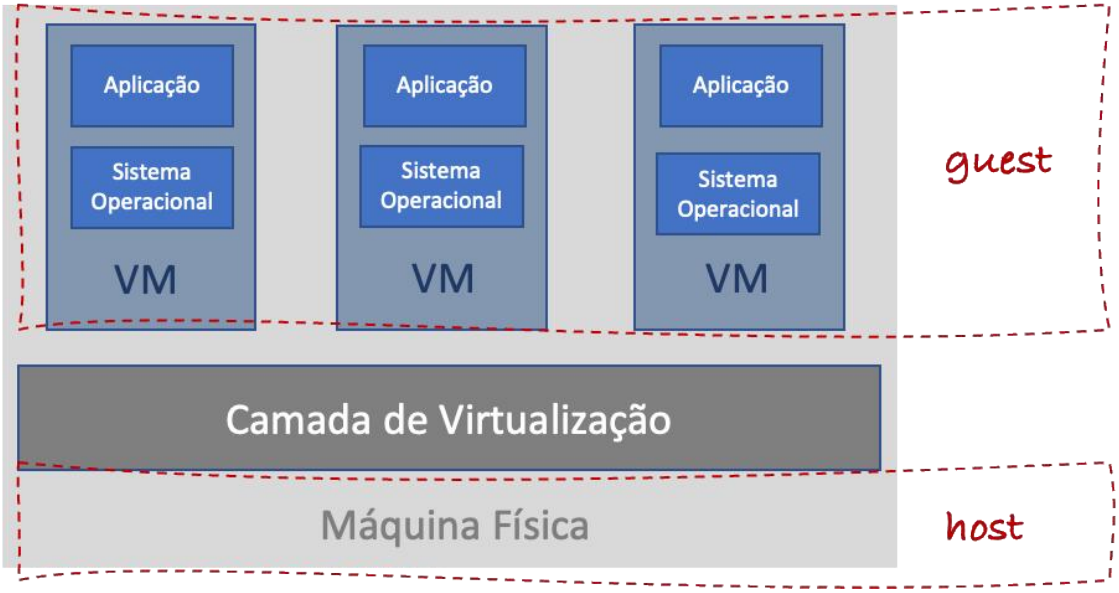
Virtualização é uma tecnologia, que foi introduzida no início dos anos 70 pela IBM, para ajudar os administradores e usuários de sistemas a enfrentarem desafios ligados ao desempenho e fornecimento de recursos e infraestrutura de computação. Foi por meio da virtualização que surgiram a computação em nuvem e os *data centers* (BENTALEB et al., 2022).

A virtualização é definida por Bermejo; Juiz; Guerrero (2019) como sendo um conjunto de tecnologias e conceitos que fornecem abstração e isolamento de ambientes de execução de aplicativos. Ela é utilizada para incorporar servidores em infraestruturas de informática e de dados, permitindo melhorar a utilização e alocação de recursos. Assim, por meio da virtualização é possível reduzir a complexidade e aumentar a flexibilidade de um sistema (DUA; RAJA; KAKADIA, 2014; GHATREHSAMANI et al., 2020; MONDESIRE et al., 2019).

A virtualização começou a ser mais utilizadas devido a necessidade de aumento dos recursos de servidores e as limitações de espaço físico nos *data centers*. O uso mais comum da virtualização está associado a *hardware*, como a solução de infraestrutura para computação em nuvem (GHATREHSAMANI et al., 2020). Segundo Bentaleb et al. (2022), o maior benefício trazido pela virtualização foi a abstração do *hardware*, fornecendo um ambiente isolado para aplicativo e criando um *pool* com recursos lógicos (i.e., CPU, memória, rede e discos).

Um ambiente de virtualização é composto por três entidades, são elas: convidado (*guest*), *host* e camada de virtualização (*virtualization layer*), as quais estão representadas na Figura 9. O convidado interage com a camada de virtualização e representa os componentes do sistema. O *host* corresponde ao sistema original, que é capaz de alocar e gerenciar o convidado. A camada de virtualização (ou gerenciador de máquina virtual), por sua vez, é um pedaço de *software* alocado entre os recursos físicos e de aplicação, que comanda as máquinas virtuais, sendo capaz de criá-las, realizar migrações, copiar e apagá-las. Caso se tratar de uma virtualização de *hardware*, os convidados são representados por uma imagem do sistema (BERMEJO; JUIZ; GUERRERO, 2019).

Figura 9 - Entidades de um ambiente de virtualização



Fonte: A autora.

Implementar a tecnologia de virtualização faz com que as restrições de *hardware* sejam abstraídas e a flexibilidade do sistema aumente, devido a criação de um ambiente abstrato e isolado para executar aplicações (MONDESIRE et al., 2019). Devido a utilização cada vez maior de servidores em nuvem, a virtualização ganhou mais importância na execução de vários servidores em uma única infraestrutura compartilhada, ficando conhecida como consolidação de máquina virtual (BENTALEB et al., 2022).

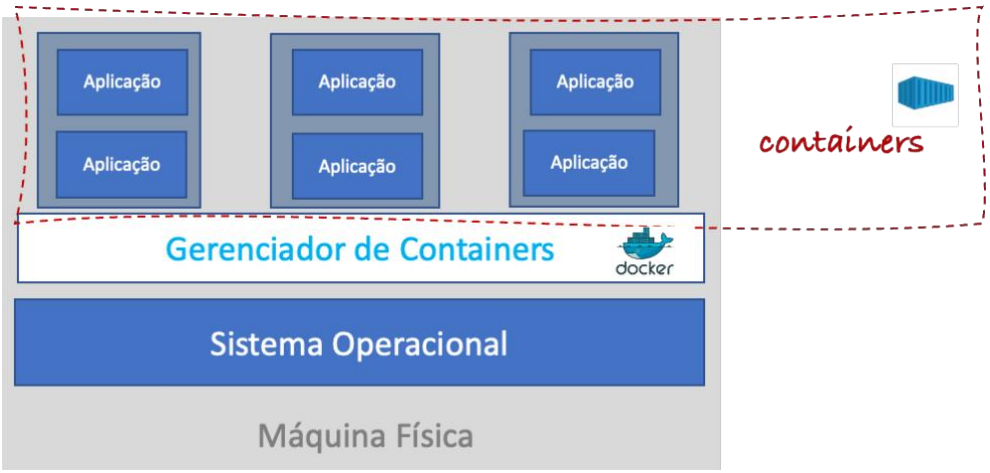
Apesar da diminuição de custos proporcionada pela virtualização, sua utilização gera um aumento da complexidade dos sistemas, e como a instância de um sistema operacional em uma máquina virtual (VM) roda como um único processo, as máquinas virtuais acabam apresentando desempenho inferior ao de máquinas físicas equivalentes, já que (ABUABDO; AL-SHARIF, 2019; BENTALEB et al., 2022; MONDESIRE et al., 2019).

A virtualização mais utilizada é a nível do sistema operacional, que permite o uso de mecanismos de isolamento, fornecendo aos usuários ambientes virtuais semelhantes a um servidor dedicado. Este ambiente virtual isolado é denominado de *container*, fazendo alusão ao método padrão de transporte de cargas (DUA; RAJA; KAKADIA, 2014).

A conteinerização concentra-se em abstrair o nível do sistema operacional, ao invés de virtualizar uma *stack* (do inglês, pilha) de *hardware* com máquinas virtuais (Figura 10). Dessa maneira, a conteinerização é uma maneira de executar aplicativos, sendo possível desenvolvê-los, testá-los e implantá-los dentro dos *containers*. Além disso, a eficiência da interconexão

entre *containers*, sua leveza e portabilidade são algumas das vantagens de utilizar essa tecnologia (BENTALEB et al., 2022).

Figura 10 - Entidades de um ambiente de conteinerização



Fonte: A autora.

Por meio da utilização de *containers* é possível isolar e gerenciar recursos no ambiente Linux. Assim, um *container* com sistema operacional permite isolar um processo do restante do sistema, fazendo com que o tempo de inicialização dos *containers* seja menor se comparada com uma VM, afinal, é possível compartilhar recursos com as máquinas *host* (DUA; RAJA; KAKADIA, 2014). Ademais, os containers tratam cada instância em execução de maneira independente, concedendo isolamento espacial, de processos, de sistemas de arquivos, nomes e recursos de hardware (MONDESIRE et al., 2019).

Devido a capacidade de promover a implantação rápida de aplicações, a conteinerização criou uma mudança na maneira como as operações computacionais são realizadas. Os *containers* contêm o código de uma aplicação e suas dependências empacotados em um ambiente encapsulado, podendo processar aplicações em larga escala (ABUABDO; AL-SHARIF, 2019; MONDESIRE et al., 2019). Algumas das plataformas baseadas em *containers* são: Singularity, uDocker e Docker. Singularity, desenvolvido pelo Lawrence Berkeley National Laboratory (Sylabs Inc.) e a uDocker se concentram em abordagens de *containers* no sistema operacional Linux, já a plataforma Docker trabalha com sistemas operacionais Linux e Windows, sendo a mais popular entre elas (BENTALEB et al., 2022).

2.10.1 Docker

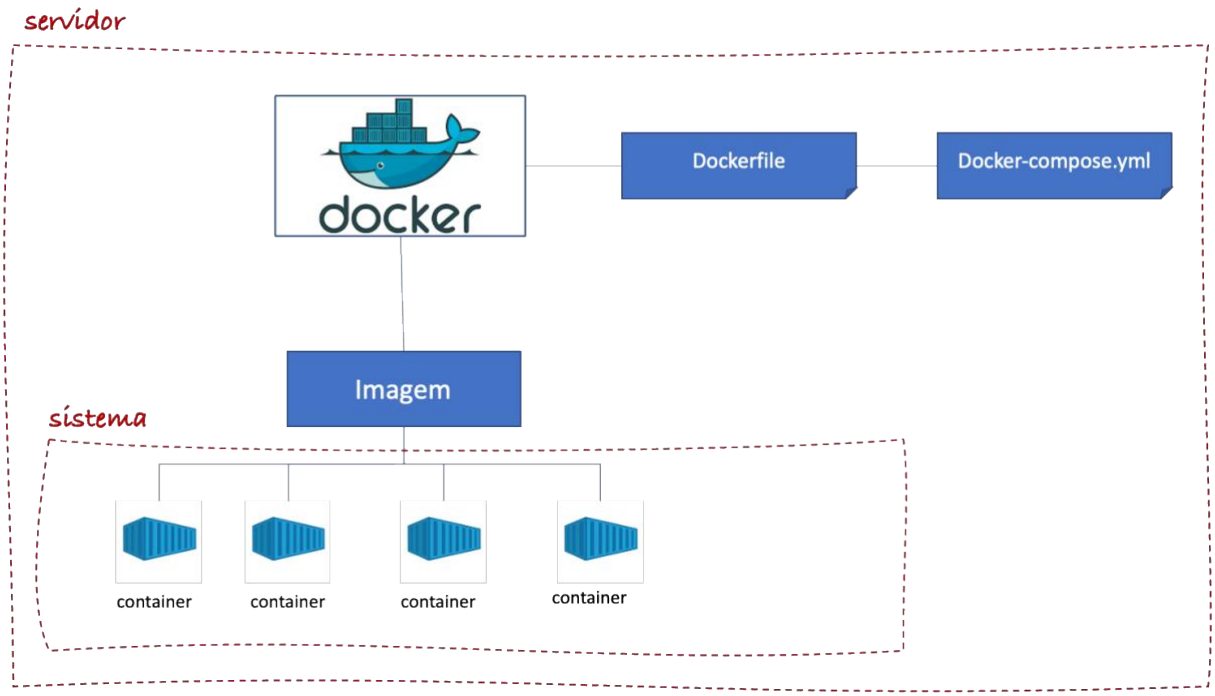
Docker é uma plataforma *open source* utilizada para o desenvolvimento e execução de pacotes de aplicação em *containers* e foi desenvolvida para ser um ambiente leve e rápido (BENTALEB et al., 2022; PARAISO et al., 2016). Para Scheepers (2014), o Docker facilita o empacotamento e uma aplicação e todas as duas dependências em um *container*. Ainda, é possível enviar, testar, implantar códigos mais rapidamente, além de encurtar o tempo entre a escrita e a execução do código com a utilização dessa plataforma (PREETH et al., 2015; TURNBULL, 2016, p.7).

O Docker é composto por diversos recursos necessários para seu funcionamento, como imagens, *containers* e volumes (Figura 11). As imagens são arquivos que contém as instruções para instanciar um *container*, podendo ser consideradas o código-fonte dos *containers*. Por ser uma plataforma *open source*, diversos arquivos *dockerfile*, os quais contém todas as etapas de construção de uma imagem, estão disponíveis no *DockerHub* (https://hub.docker.com), podendo ser facilmente baixadas e utilizadas. Ainda, é possível compartilhar, armazenar e atualizar as *Docker Images* (BOETTIGER, 2015; MERKEL, 2014).

Os *containers* são instâncias em execução que tem como base imagens que podem conter um ou mais processos. Eles são um ambiente de execução composto por um conjunto de operações padrão, iniciados a partir de imagens, e podem conter um ou mais processos em execução. Já os volumes representam o espaço reservado para armazenar o conteúdo que um *container* produz ou recebe, sendo esse espaço isolado dos arquivos principais do *host*. Um volume pode ainda trabalhar mantendo os dados dos *containers* mesmo após sua parada ou apenas mantê-los enquanto estiver sendo executado (BOETTIGER, 2015; RAD; BHATTI; AHMADI, 2017).

A plataforma Docker conta ainda com um mecanismo que fornece uma eficiente camada para executar micro serviços e possibilita a criação de grupos de *containers* a partir de um arquivo YAML contendo todas as configurações de suas aplicações. Esse mecanismo é denominado *Docker Compose*, ao utilizá-lo, com apenas um comando é possível criar e iniciar todas as aplicações contidas no arquivo YAML (BENTALEB et al., 2022; MARTIN et al., 2018; PARAISO et al., 2016).

Figura 11 - Recursos Docker



Fonte: A autora.

Miell e Sayers (2019, p.7) e Turnbull (2016, p.13) afirmam que a plataforma Docker pode ser utilizada para diversos fins, como:

- Ajudar a tornar o fluxo de desenvolvimento local mais rápido, eficiente e leve, uma vez que os desenvolvedores podem criar, executar e compartilhar *containers*;
- Executar serviços e aplicativos autônomos em vários ambientes, facilitando a arquitetura e implantação de micro serviços;
- Criar instâncias isoladas para executar testes; e,
- Construir e testar arquiteturas e aplicativos complexos em um *host* local, antes de implementá-los.

2.11 *Business Intelligence*

Business Intelligence (BI) é o processo de extrair, transformar, gerir e analisar dados de negócios, para apoiar uma melhor tomada de decisão. Esse processo é geralmente baseado em grandes conjuntos de dados, com o objetivo de disseminar conhecimento em uma organização, desde o nível estratégico até o nível operacional, proporcionando que gestores tomem decisões baseadas em dados (NIU; LU; ZHANG, 2009). Para Lim, Chen e Chen

(2013), o termo BI tem sido utilizado para descrever conceitos e métodos que tenham como objetivo a melhoria das tomadas de decisão dos negócios, e que utilizem sistemas de suporte baseados em fatos.

Segundo Lim, Chen e Chen (2013), o principal objetivo de BI é possibilitar o acesso fácil e interativo de diversos dados, a manipulação e transformação desses dados, para que analistas e gestores de negócios possam conduzir análises apropriadas e tomar medidas apropriadas. Por esse motivo, sistemas de BI também podem ser chamados de sistema de suporte à decisão (DSS) (ELENA, 2011).

Segundo Waas et al. (2013), as funções mais comuns das tecnologias de BI são relatórios, processamento analítico, mineração de dados, gerenciamento de negócios, *benchmarking* e análise preditiva. De acordo com Elena (2011), as tecnologias BI podem ser aplicadas para os seguintes propósitos: medir progresso do negócio em relação às metas traçadas, analisar processos quantitativamente para alcançar decisões ótimas, reportar informações consideradas estratégicas para um negócio, unir informações de diferentes áreas para apoiar tomadas de decisão e criar *insights* através de informações para guiar a empresa no alcance de seus objetivos.

Analistas de BI inspecionam e consultam os dados disponibilizados nos armazéns de dados para obterem informações, por isso, um sistema de BI ideal possibilita ter as informações certas no momento correto para que a melhor decisão possível seja tomada (WAAS et al., 2013). Isso é possível graças aos recursos de BI que incluem painéis personalizados, alertas automatizados, tabelas, gráficos e outras opções de visualização que permitem uma exibição clara e concisa de dados por meio da capacidade analítica (POPEANGĂ; LUNGU, 2012). Há diversas ferramentas de apoio para BI, como Tableau, Looker, Google Data Studio e Metabase (FATHONI et al., 2022).

O Metabase é uma aplicação *web*, *open source*, compatível com diversos sistemas operacionais, que pode ser utilizada em *Data Science* e para a visualização de conceitos e abordagens de BI (LIMA; CRUZ, 2019). A ferramenta permite, por meio da criação de *dashboards*, apresentar, visualizar e monitorar dados. Esse *dashboards* são criados através de buscas (*queries*), que podem ser realizadas em linguagem SQL, permitindo que os usuários processem os dados e os exibam de no formato de gráficos e tabelas de fácil compreensão (JOVANOSKA; NECHKOSKA; MANCHESKI, 2021).

2.12 Manutenção de equipamentos

O sucesso da produção agrícola depende, em grande parte, da manutenção dos equipamentos utilizados nas operações de cultivo, uma vez que garante a segurança das operações e disponibilidade de máquinas e equipamentos. Com o aumento da competitividade no setor, houve a necessidade de melhorar as abordagens utilizadas e buscar a redução dos custos relacionados a manutenção dos equipamentos agrícolas (KHODABAKHSHIAN, 2013).

A manutenção refere-se a todas as tecnologias e gerenciamento realizados para garantir ou restaurar o desempenho das funções de equipamentos, garantindo sua segurança e confiabilidade (HAO; ZHAO; WANG, 2018). Lee, Kim e Yoe (2017) definem manutenção como todas as ações necessárias para manter ou restaurar um sistema, de tal modo que ele cumpra sua função pretendida, controlando o custo advindo das atividades de manutenção e a potencial perda de produção.

De acordo com Lüttenberg, Bartelheimer e Beverungen (2018), há três estratégias de manutenção que podem ser adotadas: corretiva, preventiva e preditiva. A manutenção corretiva é realizada apenas quando ocorrem falhas e as máquinas deixam de funcionar. Assim, nessa estratégia a manutenção não é planejada nem agendada e os componentes são utilizados o máximo possível, resultando na minimização dos custos com componentes, mas tornando as máquinas mais vulneráveis a paradas (BEN-DAYA et al., 2009).

Já a manutenção preventiva busca substituir ou reparar componentes antes da ocorrência de falhas. Esse tipo de manutenção de equipamentos é realizado periodicamente e tem como base as recomendações dos fabricantes de seus componentes (BEN-DAYA et al., 2009). Considerando essa regularidade na troca de componentes, pode ocorrer a sua substituição antes do final da sua vida útil, ou seja, acaba-se por não esgotar totalmente a vida útil dos componentes que já estão em serviço gerando, assim, um aumento de custos em comparação à manutenção corretiva. Por outro lado, a manutenção preventiva reduz os tempos de inatividade dos equipamentos, uma vez que as atividades de manutenção são realizadas antes que ocorra um defeito (LÜTTENBERG; BARTELHEIMER; BEVERUNGEN, 2018).

Por sua vez, na manutenção preditiva, programa-se a troca ou reparo de componentes de equipamentos com base em suas condições. Assim, os componentes são utilizados o maior tempo possível, mas são substituídos antes que apresentem algum defeito (MOBLEY, 2002). É possível realizar essa estratégia de manutenção devido ao monitoramento e análise de dados

provenientes de sensores instalados nos equipamentos. Quanto aos custos envolvidos, essa estratégia permite programar as atividades e paradas de equipamento para manutenção de forma eficiente, e ao mesmo, reduzir os custos relativos à substituição de componentes (LIPS; BUROSE, 2012; NADJ et al., 2016). Segundo Lüttenberg, Bartelheimer e Beverungen (2018), a estratégia de manutenção preditiva é a melhor escolha para minimizar os custos totais com manutenção de equipamentos.

3 ASPECTOS METODOLÓGICOS

Neste capítulo a caracterização da pesquisa proposta e os procedimentos empregados são apresentados. A Seção 3.1 aborda a caracterização da metodologia de pesquisa utilizada neste trabalho, seguida pela Seção 3.2, onde a abordagem metodológica é apresentada. Na Seção 3.3 o procedimento metodológico adotado é descrito, e por fim, a Seção 3.4 apresenta as limitações da pesquisa.

3.1 Caracterização da pesquisa

Pesquisas podem ser classificadas quanto aos seus objetivos como prescritivas e descritivas. As pesquisas prescritivas buscam compreender e descrever determinados fenômenos, analisando valores reais para o desenvolvimento de teorias ou hipóteses, objetivando propor melhorias e soluções para um problema (HEVNER; CHATTERJEE, 2010). Enquanto pesquisas descritivas visam fornecer diagnóstico sobre um problema, direcionando o objeto da pesquisa para o problema e não para a solução (BONAT, 2009). Sendo assim, de acordo com os objetivos dessa pesquisa, a mesma pode ser caracterizada como uma pesquisa prospectiva, uma vez que ela pretende desenvolver um modelo (artefato) capaz de selecionar e interpretar dados para auxiliar nas tomadas de decisão.

Propõe-se a adoção do *framework* metodológico *Design Science Research* (DSR) como método de pesquisa para pesquisas prescritivas, posto que essa abordagem objetiva desenvolver e projetar artefatos para melhorar sistemas, solucionar problemas ou apresentar novos artefatos que contribuam alguma mudança em um sistema, seja melhorando seu desempenho ou resolvendo um problema (LACERDA et al., 2013).

3.2 Abordagem metodológica

A condução da presente pesquisa é baseada na abordagem *Design Science Research* (DSR), a qual visa auxiliar no desenvolvimento de artefatos que busquem resolver os problemas observados, realizar contribuições de pesquisa, avaliar as soluções e comunicar os resultados (PEFFERS et al., 2007; SIMON, 2019). Por conseguinte, essa abordagem objetiva aperfeiçoar a percepção de profissionais em seus campos de atuação para que problemas sejam solucionados (SIMON, 2019).

Segundo Simon (2019), artefatos opõem-se a algo natural, sendo definidos como algo construído por seres humanos. Artefatos são projetados com o objetivo de alterar algo em um sistema, objetivando resolução de problemas ou melhoria de desempenho, sendo definidos como: métodos (i.e., boas práticas e algoritmos); modelos (i.e.; representações e abstrações); construções (i.e., símbolo e vocabulário); e instâncias (i.e., protótipos e implementações de sistemas) (LACERDA et al., 2013).

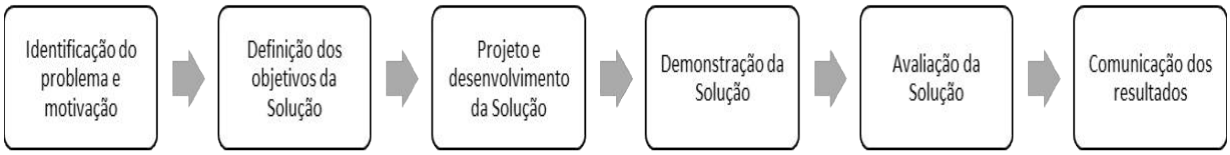
Para Von Alan et al. (2004), o conhecimento e compreensão de um problema e sua solução são adquiridos na construção e aplicação de um artefato. Assim, propõe-se a construção de modelos para o desenvolvimento do presente trabalho, a fim de que o objetivo proposto seja atingido. Para Lacerda et al. (2013), modelos são conjuntos de proposições ou declarações que descrevem, ou seja, representam um cenário real. Baseando-se na DSR, a atenção na construção do modelo deve estar voltada à sua utilidade.

De acordo com Peffers et al. (2007), para a criação de um artefato consolidado, a abordagem DSR define as seguintes etapas que devem ser seguidas:

i) Identificação do problema e motivação;
ii) Definição dos objetivos da Solução;
iii) Projeto e desenvolvimento da Solução;
iv) Demonstração da Solução;
v) Avaliação da Solução; e
vi) Comunicação da Solução.

A seguir, cada uma das fases que compõem a estrutura metodológica da abordagem é explicada. A Figura 12 apresenta um fluxograma com essas fases.

Figura 12 - Estrutura metodológica da abordagem DSR



Fonte: Adaptado de Peffers et al. (2007)

Fase 1 - Identificação do problema e motivação: Essa etapa é constituída da definição do problema de pesquisa específico e da justificativa para a solução proposta. Nessa fase é imprescindível conhecer o estado do problema e a importância de sua solução.

Fase 2 – Definição dos objetivos da Solução: Deve-se nessa fase inferir os objetivos da Solução a partir da definição do problema e do entendimento do que é possível e viável e da definição do problema. Para tanto é necessário conhecer o estado dos problemas e das soluções atuais e sua eficácia.

Fase 3 – Projeto e desenvolvimento da Solução: Consiste em criar o artefato, determinar a sua funcionalidade desejada e planejar sua arquitetura.

Fase 4 – Demonstração da Solução: Utilização do artefato para resolver uma ou mais instâncias do problema, i.e. experimentação, simulação, prova, estudo de caso ou outra atividade apropriada.

Fase 5 – Avaliação da Solução: Nessa fase deve-se comparar os objetivos da solução com os resultados obtidos através da sua demonstração, observando e medindo quão bem o artefato auxilia na solução do problema. De acordo com Von Alan et al. (2004) e Hevner e Chatterjee (2010), a avaliação pode ser:

- Observacional (estudo de caso e estudo de campo);
- Analítica (análise estática, análise de arquitetura, otimização e análise dinâmica);
- Experimental (experimento controlado e simulação);
- Teste (funcional – caixa preta e estrutural – caixa branca); e
- Descritiva (argumento informado e cenários).

Fase 6 – Comunicação dos resultados: Nessa etapa deve-se comunicar o problema e sua importância, o artefato, a utilidade e a novidade, e sua eficácia para os pesquisadores e público relevante.

Cada uma dessas atividades está detalhada na seção 3.3, a fim de apresentar como e o que foi realizado para a construção do modelo.

3.3 Procedimento metodológico

3.3.1 Identificação do problema e motivação

A Identificação do problema e motivação, primeira etapa da DSR, relaciona-se com a identificação do contexto de aplicação do modelo na empresa parceira. De maneira geral, se tem como objetivo principal nessa etapa direcionar o pesquisador para a resolução de um

problema existente. Executamos essa etapa com base em reuniões com membros da empresa parceira e através de uma busca realizada na literatura.

Realizamos a busca na literatura por meio de uma análise bibliométrica e sistêmica acerca do tema de pesquisa, a fim de obtermos uma base conceitual sobre o assunto, auxiliar na identificação de pesquisas já realizadas e seus resultados alcançados, bem como detectar tendências e lacunas de pesquisa sobre o tema em questão.

A busca na literatura que norteou a proposta apresentada nesta pesquisa foi realizada por meio de uma análise bibliométrica e sistêmica. Nesta busca adotamos o *Knowledge Development Process-Constructivist* (*Proknow-C*) como procedimento metodológico. Este instrumento define um fluxograma para a revisão bibliométrica e foi desenvolvido para auxiliar os pesquisadores no gerenciamento de informações e conhecimentos de conteúdos relevantes em uma determinada questão científica (TASCA et al., 2010).

O processo de revisão estruturada de literatura é uma ferramenta fundamental, já que há uma imensa disponibilidade de informações na atualidade, sendo utilizado para gerenciar a diversidade de conhecimentos para uma pesquisa acadêmica. Ao realizar uma revisão estruturada da literatura é possível mapear e avaliar o conhecimento existente e definir uma questão de pesquisa para desenvolver ainda mais o conhecimento existente (TRANFIELD; DENYER; SMART, 2003).

O *ProKnow*-C é composto pelos seguintes passos: seleção do portfólio bibliográfico, seguido da análise bibliométrica, análise sistêmica do portfólio e por fim, definição da pergunta e objetivo da pesquisa respectivamente. Na etapa "definição do portfólio bibliográfico" define-se os eixos de pesquisa e as palavras-chave, bem como as bases de dados, e então realiza-se as buscas. Após, inicia-se a etapa de análise bibliométrica, na qual define-se os autores, artigos, periódicos e as palavras-chave relevantes à questão de pesquisa. Por fim, na análise sistêmica, é realizada a interpretação dos artigos do portfólio bibliográfico (ENSSLIN et al., 2010).

Para a realização desta etapa, todos os passos propostos pelo *Proknow*-C foram seguidos e identificamos as seguintes oportunidades de pesquisa:

i) caracterização uma Fazenda 4.0, com todas as tecnologias e fluxos de informações presentes;

ii) criação de plataformas IoT genéricas voltadas para a agricultura;

iii) desenvolvimento de dispositivos IoT que levem em consideração algoritmos com criptografia avançada;

iv) elaboração de um modelo de informações universal para implementar a agricultura inteligente;

v) predição de eventos considerando todos os fatores envolvidos no cenário agrícola.

Desse modo, foi possível perceber que as pesquisas relacionadas à Agricultura 4.0 não apresentam uma visão sistêmica, ou seja, tratam do tema de maneira restrita, buscando encontrar soluções isoladas e pontuais. A análise bibliométrica e sistêmica realizada nessa pesquisa pode ser vista em detalhes por meio dos seguintes estudos publicados: "*Digitalization and Big Data in smart farming–a review*" (IAKSCH; FERNANDES; BORSATO, 2021) e "*Digitalization and Big Data in Smart Farming – Bibliometric and Systemic Analysis*" (IAKSCH; FERNANDES; BORSATO, 2020).

Nas reuniões com membros da empresa parceira, responsáveis pela área de inovação, foram identificados os níveis de integração das diversas tecnologias adotadas pela empresa em seus produtos com o contexto de uma fazenda inteligente. Foi possível diagnosticar que, apesar de contarem com diversos sensores IoT instalados nos equipamentos, capazes de gerar dados que auxiliem em tomadas de decisões, poucos dados por eles coletados são utilizados. Isso ocorre não só pela falta de conectividade no campo, mas também por não haver nenhuma solução capaz de unir esses dados e fornecer informações úteis, ou seja, empresas que fornecem equipamentos agrícolas estão equipando seus produtos com sensores capazes de coletar diversos tipos de dados, contudo esses dados não são utilizados de maneira a gerar informações úteis, tão pouco para melhorar a produtividade global das fazendas.

Ao realizarmos essas reuniões, também percebemos que os engenheiros de produto e de manutenção previam e programavam a manutenção de equipamentos agrícolas considerando cenários que se aproximavam das condições de trabalho reais em campo e de informações sobre vida útil de componentes fornecidos pelos fabricantes. No entanto, na prática, esses equipamentos agrícolas eram utilizados em regiões com climas muito distintos, as quais possuíam particularidades relacionadas à temperatura do ambiente e umidade, o que fazia com que muitas vezes essa programação de manutenção fosse imprecisa.

3.3.2 Definição dos objetivos da Solução

Na segunda etapa da DSR, Definição de objetivos da Solução, deve-se elaborar os objetivos esperados para a resolução do problema. Portanto, de acordo com os objetivos geral e específicos, foram definidos os resultados esperados para esta pesquisa.

Após a realização da revisão de literatura e do diagnóstico na empresa parceira, definimos que o artefato desenvolvido teria seu enfoque na melhoraria da programação e em antever a necessidade de manutenção de componentes de equipamentos agrícolas. Assim, seria possível auxiliar na predição da necessidade de manutenção, e consequentemente promoveria o aumento da disponibilidade dos equipamentos.

Portanto, a solução proposta teve como objetivo guiar as decisões relativas ao momento de troca de um determinado componente do equipamento, uma vez que o modelo apresentaria todos os dados necessários para apoio à tomada de decisão. Então, a solução deveria considerar todas as informações relevantes às trocas de manutenção, i.e., dados de sensores de máquina, aspectos climáticos da localidade onde encontrava-se o equipamento e estoque disponível.

Isto posto, estabelecemos que o artefato desenvolvido neste trabalho corresponderia a um modelo sistêmico ao longo de um cenário agrícola, com o objetivo de capturar dados relacionados à manutenção em bases de dados já existentes e alimentadas por IoT, para controlar e prever a necessidade de manutenção dos equipamentos, auxiliando nas tomadas de decisão de maneira preditiva. Para que fosse possível antever a necessidade de manutenção, e como consequência aumentar a disponibilidade de equipamentos, o artefato desenvolvido teve seu enfoque em dados capturados por sensores.

3.3.3 Projeto e desenvolvimento da Solução

As atividades realizadas na etapa de Projeto e desenvolvimento da Solução são relacionadas à construção do artefato. Portanto, essa fase contou com a utilização de ferramentas para a construção do modelo de informação.

Para atingir o objetivo definido na segunda etapa do DSR, optamos por desenvolver uma solução composta por três artefatos. Os artefatos foram criados considerando-se os níveis organizacionais envolvidos nos processos de tomada de decisão das organizações. Diante disso, tem-se cada artefato que compõe a Solução representa os níveis organizacionais aos

quais as decisões podem ser relacionadas, desde as decisões organizacionais gerais, que afetam as empresas como um todo, até as decisões operacionais que se referem a decisões específicas.

Primeiramente criamos um modelo que proporcionou uma visão sistêmica e caracterizou a Agricultura 4.0. Esse Modelo de Referência de Agricultura 4.0 foi proposto com base no modelo de referência da Indústria 4.0 denominado RAMI. O primeiro modelo desenvolvido trata-se de uma representação gráfica que incorporou os princípios apresentados no RAMI e adaptou-os à realidade agrícola.

Após, iniciamos a criação do Modelo de Referência, desenvolvendo assim, o segundo modelo, o qual retratou todos os agentes de uma fazenda orientada à Agricultura 4.0 - Fazenda 4.0, representando todas as tecnologias e relações entre eles de maneira integrada e universal. Esse artefato contou com a representação de todos os agentes que compõem uma fazenda (i.e., equipamentos, estruturas, recursos humanos) e representou todos os fluxos de informações que ocorrem entre eles.

Para desenvolver esse modelo, escolhemos utilizar a linguagem de modelagem de sistemas SysML, já que esta linguagem fornece uma abordagem gráfica, facilitando a modelagem de sistemas complexos. O desenvolvimento desse modelo foi apoiado pela utilização do *software Enterprise Architect* 16.1, por ser uma ferramenta capaz de criar todos os tipos de diagramas SysML, e por haver a disponibilidade de uma licença acadêmica do programa. Ainda, antes de criar o modelo no EA, elaboramos um mapa mental para auxiliar a guiar o desenvolvimento do modelo, o qual foi criado utilizando-se o *software XMind*. Com base no mapa mental, criamos dois diagramas estruturais, o diagrama de pacotes que organizou o modelo e serviu como base para a criação do diagrama de blocos, que representou a fazenda 4.0 e todas as relações entre seus agentes. Com o modelo criado, viabilizou-se operar uma visão sistêmica da Fazenda 4.0.

Após a modelagem da Fazenda 4.0, elaboramos um modelo de informação ao longo de um cenário agrícola genérico, o qual possibilitasse a análise de dados e a predição de eventos para antecipar tomadas de decisão. Esse modelo de informação capturou os dados gerados em um dos fluxos de informação representados no modelo da Fazenda 4.0, a fim de, controlar e prever a necessidade de manutenção dos equipamentos, auxiliando nas tomadas de decisão de maneira preditiva.

Escolhemos para ser representado no modelo de informação uma colheitadeira de cana-de-açúcar. A escolha da colheitadeira justificou-se pelo cultivo de cana ser de extrema importância econômica para o país. De acordo com Silva et al., 2008), a agroindústria

canavieira é uma das culturas mais importantes do agronegócio brasileiro devido a sua utilização na produção de açúcar, etanol e energia elétrica, proveniente do seu bagaço e resíduos da colheita. Ainda, segundo Narimoto e Belussi (2018), no setor agrícola, uma das fases consideradas mais críticas, para qualquer cultura, é a colheita. Além disso, a empresa estava realizando estudos para implantação de novos sensores nas colheitadeiras de cana-de-açúcar no momento do início dessa pesquisa.

Para o caso de aplicação desse estudo foi utilizado o filtro de combustível, por ser um componente considerado de simples complexidade, e suas características e parâmetros serem conhecidos. No entanto, qualquer outro componente ou equipamento agrícola poderiam ter sido escolhidos como objeto de aplicação da solução.

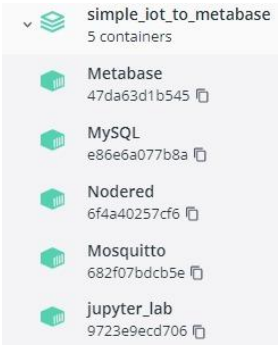
Como a gestão das fazendas e seus processos são influenciados pelas condições climáticas da área de cultivo, foram consideradas três fazendas em três cidades distintas, possibilitando representar regiões com diferentes condições climáticas. Assim, as fazendas foram representadas no modelo como *Field_01*, *Field_02* e *Field_03*, respectivamente. Ainda, os aspectos meteorológicos de cada região como temperatura, umidade, pressão atmosférica e velocidade do vento, também foram ponderados pelo Modelo.

Quanto aos equipamentos, considerou-se dois tipos de colheitadeira de cana para fazerem parte do modelo. Dessa maneira, foi definido que haveria duas colheitadeiras em cada uma das três fazendas, ou seja, ao todo seis equipamentos foram representados. Como optamos por utilizar o filtro de combustível como componente de aplicação da Solução, os dados provenientes dos sensores que possuíam relação com ele foram considerados. Desse modo, alcançamos o quarto objetivo específico deste trabalho o qual era selecionar um dos agentes do modelo sistêmico da fazenda genérica.

Levando em conta que, o desenvolvimento desse terceiro artefato seria realizado localmente, no computador da autora dessa pesquisa, buscamos uma solução para que eventualmente, caso a empresa optasse por utilizar o modelo após sua concepção, fosse possível sua aplicação sem a necessidade de programar e instalar todas as ferramentas necessárias novamente. Escolhemos então pela utilização da plataforma Docker, a qual facilita a criação e administração de ambientes isolados, criando arquivos dentro de *containers* virtuais. Outro motivo de utilizar a plataforma, é devido a possibilidade de implantação dos *containers* criados em soluções de computação em nuvem (i.e., *Azure* e *Amazon AWS*) (RAD; BHATTI; AHMADI, 2017). Dessa maneira, foi instalado o *Docker Desktop* (versão 4.11.1).

Iniciamos o desenvolvimento do Modelo de Informação com a criação de *containers* no Docker. Esses *containers* foram gerados através da utilização da ferramenta *compose*, a qual permite gerar um arquivo único para definir e compartilhar arquivos. Assim, criou-se um arquivo único, em linguagem *YAML*, contendo todo o código para executar e instalar a aplicação (Apêndice A). Esse código deu origem a uma *stack*, do inglês pilha, a qual continha cinco *containers*, cada um com uma das ferramentas utilizadas no desenvolvimento do modelo de informação, são elas: Metabase, MySQL, Node-RED, Mosquitto e Jupyter. A Figura 13 apresenta a *stack* e seus cinco *containers*.

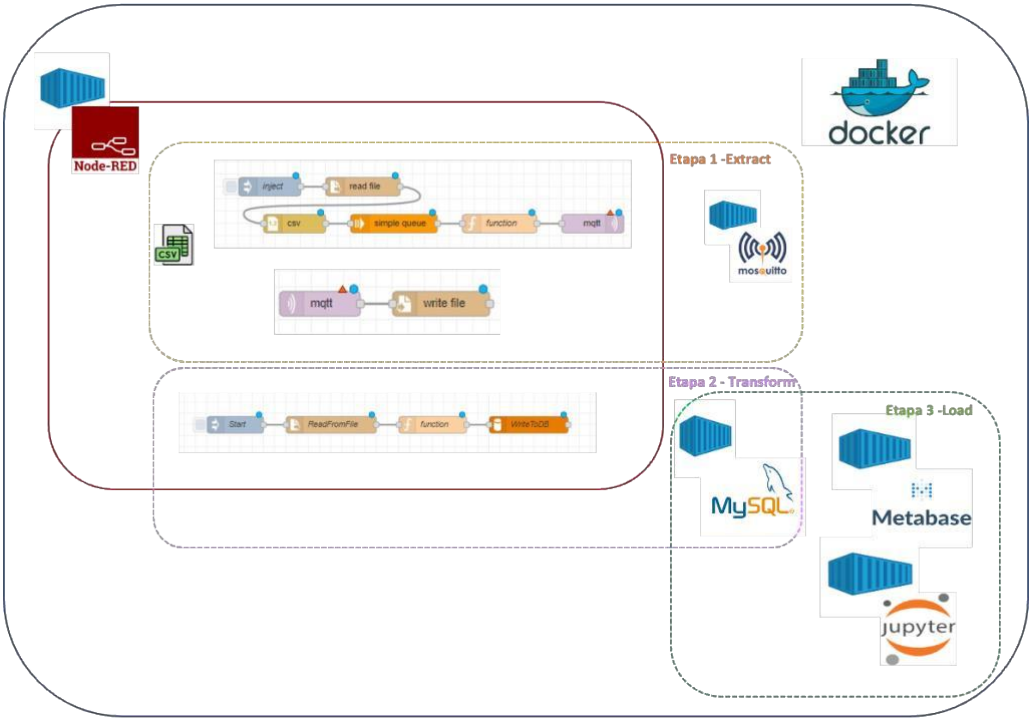
Figura 13 - Docker – *Stacks* e *Containers*



Fonte: A autora

Com a criação dos *containers* no Docker, as ferramentas estavam disponíveis e prontas para serem utilizadas. Então iniciamos a condução do desenvolvimento desse Modelo, que foi dividida em três etapas, conforme o processo ETL de integração de dados: *Extract*, *Transform* e *Load*. A Figura 14 apresenta de maneira esquemática as ferramentas utilizadas e as três etapas realizadas na construção do Modelo.

Figura 14 - Etapas da construção do Modelo de Informação



Fonte: A autora

Na primeira etapa de desenvolvimento do Modelo (*Extract*) utilizamos a ferramenta Node-RED e o *broker* Mosquitto. Nessa etapa, a Node-RED (versão 3.0.2) foi utilizada para programar os fluxos de injeção de informações do Modelo. Os "nós" utilizados nessa pesquisa são apresentados no Apêndice B. A utilização dessa ferramenta justifica-se pelo fato dela ser de fácil entendimento, uma vez que é uma ferramenta de programação visual. Outro motivo de ter sido utilizada nessa pesquisa é o fato que ela é de código aberto, gratuita, e permite interligar dispositivos físicos, ambientes de *software* e serviços em nuvem (FERENCZ; DOMOKOS, 2019). Empregamos o *broker* Mosquitto (versão 2.0) no Modelo pois utiliza o protocolo MQTT o qual é excelente para comunicação de sensores e compatível com qualquer tipo de equipamento, principalmente daqueles que são dotados de tecnologia IoT. Ademais, é extremamente seguro, protegido de falhas, e de código aberto (DA SILVA et al., 2016).

A criação do primeiro fluxo foi necessária, pois no caso do cenário de aplicação dessa pesquisa, como não havia conectividade nas fazendas de cana-de-açúcar, não havia o envio dos dados por meio dos sensores IoT. Dessa forma, foi necessário criar um fluxo que injetasse os dados emulando os dispositivos IoT. Assim, esse primeiro fluxo poderia ser retirado do Modelo de Informação caso o cenário de aplicação contasse com conectividade. Ainda nesse fluxo, para que fosse possível realizar a injeção dos dados, os arquivos em formato *Excel* (*xls*)

tiveram que ser convertidos em arquivos de dados com extensão *csv*. Por fim, o segundo fluxo contido na etapa de injeção de dados, foi criado para receber os dados e criar registros dos dados (*dataset*).

É importante salientar que, nos casos em que os dados que estavam sendo inseridos correspondiam a informações de condições de clima, utilizamos juntamente com o Node-RED e o Mosquitto, a API *One Call API 3.0*, a qual é disponibilizada na internet pela *OpenWeather* (https://openweathermap.org). Essa API foi utilizada nesse trabalho devido a possibilidade de realizar até um milhão de chamadas por mês de maneira gratuita, e por fornecer todos os dados de clima necessários ao contexto de aplicação.

Com os registros de dados criados, iniciamos a segunda etapa de desenvolvimento do Modelo de Informação (*Transform*). Nessa etapa as transformações dos dados foram realizadas e o *dataset* foi salvo em um banco de dados. Para cada arquivo de registros criado na etapa um, uma tabela correspondente foi adicionada à base onde as informações ficaram armazenadas. Escolhemos o banco de dados MySQL (versão 5.7) para ser utilizado no Modelo, já que ele é um banco de dados relacional, de código aberto, e possui alta flexibilidade e escalabilidade (EYADA et al., 2020).

Por fim, na terceira etapa (*Load*) realizamos a conexão entre o banco de dados MySQL e a ferramenta de *business intelligence* denominada Metabase (versão 0.45), permitindo analisar e gerar *insights* sobre os dados contidos nas tabelas da base de dados. Essa ferramenta possui uma interface simples e interativa, nela é possível realizar conexões entre tabelas, explorar dados e criar *dashboards*, além de ser uma ferramenta *open source* e possuir integração com o MySQL (JOVANOSKA; NECHKOSKA; MANCHESKI, 2021).

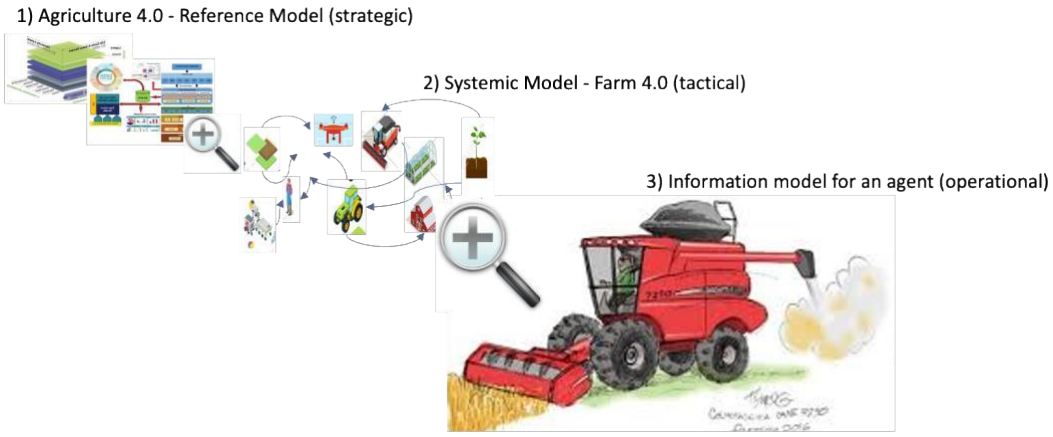
Ainda, na última etapa de desenvolvimento do Modelo de Informação, utilizamos a ferramenta *JupyterLab* (versão 3.6.0) para possibilitar a utilização de algoritmos de *machine learning* na análise dos dados do modelo. Na ferramenta foram utilizadas a linguagem *Python* de programação e o algoritmo de *machine learning RandomForest*.

Ainda, salienta-se que todos os modelos foram desenvolvidos utilizando-se um *notebook* HP Intel® Core™ i5-6300U CPU 2.40GHz – 16GB, com sistema operacional Windows 10, cedido pela empresa parceira para a realização da pesquisa.

Portanto, o primeiro artefato criado foi um Modelo de Referência para Agricultura 4.0, o qual baseou-se em um modelo de referência amplamente utilizado e consolidado na Indústria 4.0 - artefato correspondente à camada estratégica do processo decisório. Em seguida, desenvolveu-se o segundo artefato: Modelo Sistêmico de Fazenda 4.0, o qual buscou representar a camada de decisões táticas. Este modelo representou todas as trocas de

informação entre os principais agentes de uma fazenda orientada à Agricultura 4.0. Por fim, o terceiro artefato foi desenvolvido um modelo de informação aplicado a um dos agentes representados no segundo artefato - contemplando a camada decisória operacional. A Figura 15 apresenta a relação entre os modelos que compõem a solução desenvolvida e as camadas de decisão.

Figura 15 - Modelos *versus* camadas de decisão



Fonte: A autora

3.3.4 Demonstração da Solução

A quarta etapa, Demonstração da Solução, visa atestar que o artefato funciona para o fim ao qual foi proposto. Dessa forma, realizamos a aplicação do modelo proposto para que a solução concebida fosse demonstrada. Essa etapa, por sua vez, está relacionada ao objetivo específico desta pesquisa de demonstrar a aplicação do artefato por meio de provas de conceito no contexto da empresa parceira.

Levando-se em consideração o contexto da empresa parceira desta pesquisa, consideramos na demonstração do Modelo de Informação, os dados coletados pelos sensores de uma colheitadeira de cana e os aspectos externos relacionados ao ambiente de trabalho delas.

Para a demonstração da solução definimos que as cidades das três fazendas representadas no Modelo de Informação seriam: Campos Gestal, Morro Agudo e Rio Brilhante, representadas no modelo como *Field_01*, *Field_02* e *Field_03*, respectivamente. Em razão de alguns dados serem considerados sigilosos pela empresa parceira, na demonstração da solução, alguns nomes, códigos de componentes, e informações dos

equipamentos foram modificados. A relação dos nomes das colheitadeiras presentes em cada fazenda, do modelo das colheitadeiras e dos filtros considerados na demonstração estão apresentados no Quadro 1.

Quadro 1 - Relação de Fazendas, Colheitadeiras e Filtros de Combustível

City	State	Field	Machine	Machine_ID	Machine_Model	Fuel_Filter_Part_Number	Fuel_Filter_ID
Campos Gestal	SP	Field_01	Harvester_01	11AA	A9900	FPN9900	F1
			Harvester_03	13AA	A8800	FPN8800	F3
Morro Agudo	SP	Field_02	Harvester_02	12AA	A9900	FPN9900	F2
			Harvester_04	14AA	A8800	FPN8800	F4
Rio Brilhante	MG	Field_03	Harvester_05	15AA	A9900	FPN9900	F5
			Harvester_06	16AA	A8800	FPN8800	F6

Fonte: A autora

Para realizarmos a demonstração, utilizamos a ferramenta Metabase, onde foram criados *dashboards*, do inglês painéis de controle, para apoiar tomadas de decisão relativas ao cenário de aplicação. Construímos os *dashboards* a partir de perguntas (*queries*) feitas utilizando a base de dados criada na fase de desenvolvimento da solução. Ao todo criamos setenta e cinco *queries*, as quais distribuímos em três *dashboards*.

Ainda, na fase de demonstração do Modelo de Informação, importamos as tabelas de dados do MySQL para a ferramenta JupyterLab e desenvolvemos um código para demonstrar que através da estrutura de informação criada pelo modelo, utilizando algoritmos de *machine learning*, seria possível predizer dados para o modelo.

3.3.5 Avaliação da Solução

A etapa de Avaliação da Solução objetiva responder à pergunta: A solução funciona bem? Ou seja, essa etapa se correlaciona ao objetivo específico de avaliar o modelo proposto.

Segundo Lacerda et al. (2013), modelos são avaliados quanto a sua fidelidade com os fenômenos do mundo real, completude, nível de detalhe, robustez e consistência interna. De maneira geral, a solução será avaliada de maneira descritiva quanto a sua generalidade, eficiência e aplicabilidade.

Nesta etapa realizamos a observação e medição do comportamento do artefato proposto, avaliando a sua relevância. Para tal, utilizamos o cenário de aplicação inserido no contexto da empresa parceira, a fim de verificar se a proposta da solução atendia de fato as

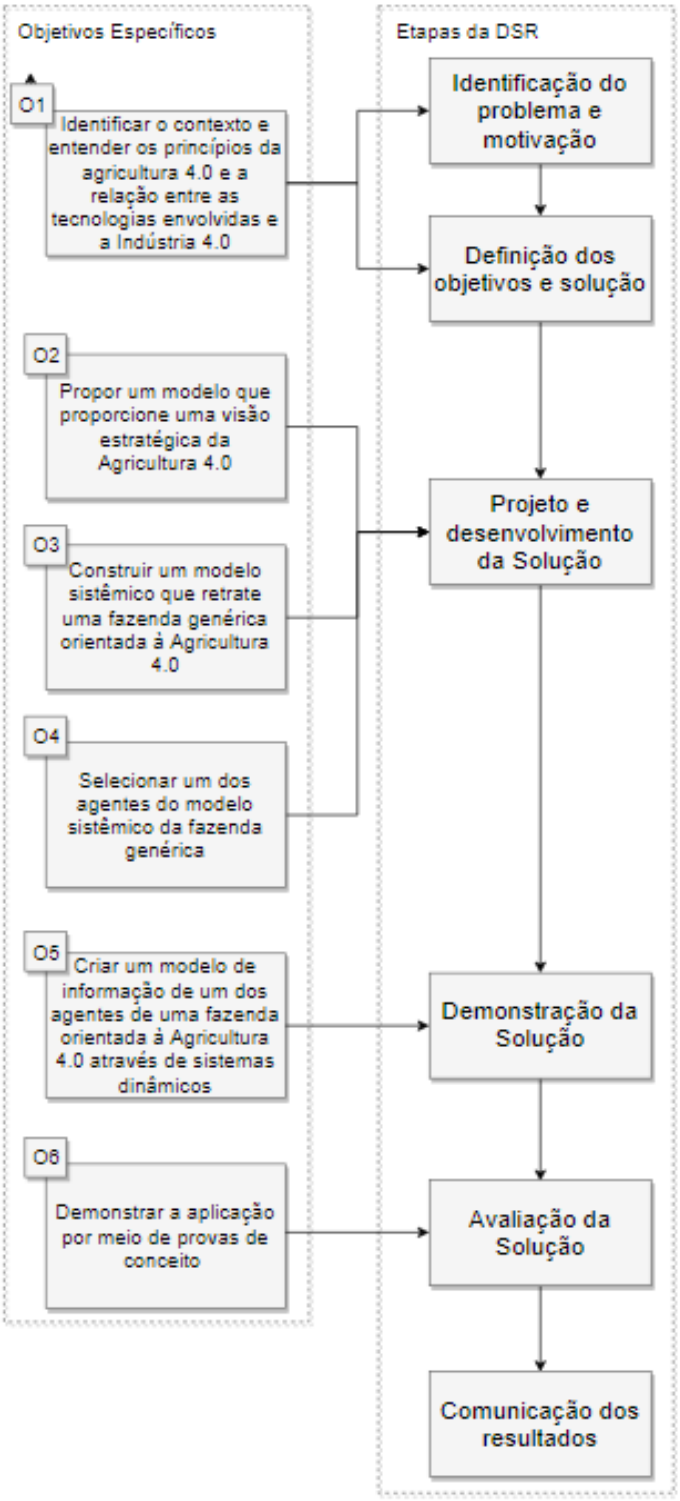
necessidades levantadas na primeira etapa da metodologia, bem como se ajustes necessitavam ser feitos. Dessa forma, realizamos a avaliação do artefato ao longo do desenvolvimento e demonstração da solução, por meio de reuniões e discussões com os membros da empresa. Assim, ajustes foram sendo realizados ao longo do desenvolvimento da solução conforme os *feedbacks* recebidos.

3.3.6 Comunicação dos resultados

A última etapa da DSR consiste em comunicar as análises e os principais resultados obtidos por meio desta pesquisa. Essa comunicação se deu através da elaboração e publicação de três artigos ao longo do desenvolvimento desse trabalho e da escrita dessa tese de doutorado. Esperamos ainda publicar um artigo referente aos resultados encontrados, ao modo como o artefato foi construído e sua aplicação no mundo real.

Através de reuniões e apresentações a presente pesquisa foi exposta aos membros interessados e envolvidos da empresa parceira. Além disso, um relatório técnico será elaborado e entregue à empresa parceira. A Figura 16 ilustra os objetivos específicos, apresentados na Seção 1.2.2, correlacionados às etapas da metodologia DSR.

Figura 16 - Procedimento metodológico



Fonte: A autora.

3.4 Limitações do trabalho

Este trabalho, apesar de ter seguido todas as etapas propostas pela abordagem DSR, apresentadas na seção 3.2 possui algumas limitações relacionadas à etapa de demonstração e avaliação.

Na etapa de demonstração, não foi possível apresentar os dados reais das colheitadeiras de cana-de-açúcar, uma vez que foi necessário criar mais dados e aumentar o *dataset*, já que a quantidade de dados fornecida pela empresa continha um intervalo pequeno de dados. Assim, os dados utilizados como *input* para as *queries* não podem ser considerados consistentes e confiáveis, consequentemente as respostas também não. Ademais, alguns dados (i.e., nomes, códigos de componentes, e informações dos equipamentos) foram modificados por serem considerados sigilosos pela empresa parceira.

Na etapa de avaliação, a solução proposta foi avaliada apenas de maneira descritiva, por meio da observação do comportamento do Modelo de Informação. Além disso, o modelo de informação não foi implementado, apenas foi avaliado por alguns membros da empresa, o que impediu uma avaliação mais aprofundada.

A próxima seção apresenta os resultados obtidos a partir da implementação dos aspectos metodológicos no desenvolvimento da solução proposta por esta pesquisa.

4 RESULTADOS E DISCUSSÃO

Este capítulo mostra os resultados obtidos nesta pesquisa, os quais seguem as etapas do *Design Science Research* (DSR) descritas no capítulo de aspectos metodológicos, apresentando as entregas obtidas em cada uma das etapas dessa abordagem.

Na primeira etapa do DSR, Identificação do problema e motivação, realizamos diversas reuniões com membros da empresa parceira, além de uma busca na literatura. Na etapa Definição dos objetivos da Solução, segunda etapa do DSR, definimos que a solução proposta deveria ser capaz de auxiliar na condução de atividades de manutenção, guiando as decisões relativas ao momento de troca de um determinado componente do equipamento, uma vez que o modelo apresentaria todos os dados necessários para apoio à tomada de decisão.

A primeira e segunda etapas foram apresentadas no primeiro capítulo deste trabalho, onde os principais aspectos sobre o tema deste trabalho foram contextualizados, a oportunidade de pesquisa foi definida, e os objetivos da solução foram apresentados. Assim, diante dos objetivos da solução e do seu contexto de aplicação, foi possível iniciar a terceira etapa do DSR, o Projeto e desenvolvimento da Solução proposta.

4.1 Projeto e desenvolvimento da solução

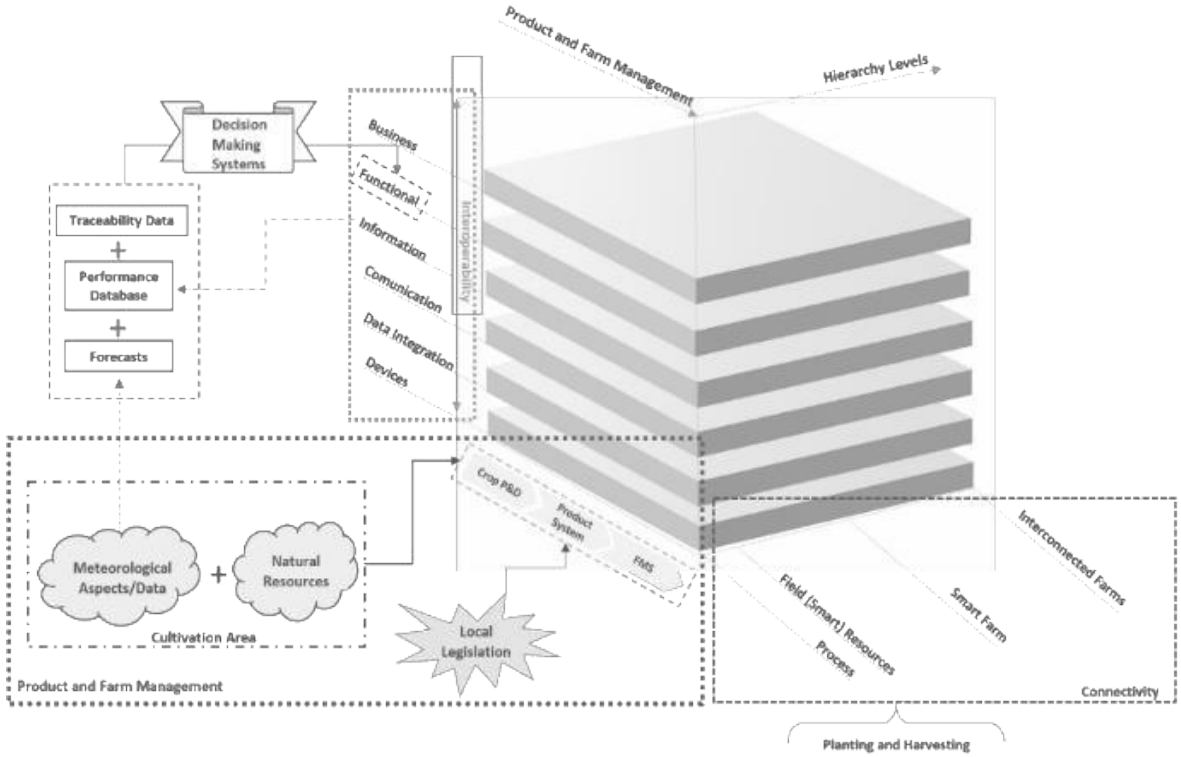
O processo de construção dos três artefatos, que em conjunto compõem a Solução proposta, é apresentado com detalhes no decorrer desta seção.

4.1.1 Modelo de Referência para Agricultura 4.0

O primeiro artefato desenvolvido para compor a solução dessa pesquisa foi um Modelo de Referência para Agricultura 4.0. Este Modelo foi criado com base no RAMI 4.0, uma referência amplamente utilizada na Indústria 4.0.

Desenvolvemos o Modelo através da incorporação dos princípios da Agricultura 4.0 e da adaptação do RAMI 4.0 ao cenário agrícola. Assim como o RAMI 4.0, o Modelo proposto é tridimensional, sendo desenvolvido com foco na integração horizontal e vertical constituído por dois eixos horizontais e um eixo vertical. A Figura 17 apresenta o Modelo proposto.

Figura 17 - Modelo de Referência para Agricultura 4.0



Fonte: A autora.

Os eixos horizontais do Modelo - *Hierarchy Levels* e *Product and Farm Management*- representam a conexão de todos os elementos do processo produtivo agrícola e o fluxo de valor da produção agrícola e sua gestão, respectivamente.

No RAMI 4.0: um dos eixos horizontais representa os níveis hierárquicos, ou seja, as camadas de integração de um sistema de controle empresarial (empresa, centros de trabalho, estação e dispositivo de controle) e as outras três camadas dão suporte ao conceito inteligente. Assim, no Modelo proposto, o eixo horizontal denominado *Hierarchy Level* (à direita da Figura 17) representou a integração entre os níveis hierárquicos presentes no cenário agrícola. Este eixo retratou a conexão de todos os agentes que compõem os processos (*i.e.*, pessoas, dados e equipamentos) e as etapas necessárias para a produção agrícola, portanto, representamos nesse eixo do Modelo todas as atividades envolvidas nesse processo, desde a plantação até a colheita.

A importância da comunicação e conexão entre sistemas e máquinas flexíveis ao longo dos processos de uma fazenda inteligente também foi representada no eixo *Hierarchy Levels*, sendo enfatizada por meio de um retângulo pontilhado e da palavra *Connectivity.* Portanto, reforçamos a importância de haver conectividade ao longo de todo o processo de produção

agrícola. Além disso, esta camada se relaciona ao conceito de "mundo conectado" presente no modelo RAMI 4.0 e buscou representar a expansão dos limites de uma fazenda, a qual ocorre através de parcerias firmadas e relações de colaboração entre diferentes propriedades.

Já no eixo horizontal, à esquerda do Modelo denominado de *Product and Farm Management*, representamos o desenvolvimento e produção das plantações e seus sistemas de gestão. No RAMI 4.0 esse eixo representa o ciclo de produção industrial, portanto no Modelo ele retratou o ciclo de produção agrícola. Ainda nesse eixo, a pesquisa e desenvolvimento de plantações representou tanto os processos de desenvolvimento e pesquisa em sementes, como novas técnicas de plantio, os quais são de extrema importância para o tempo de crescimento da plantação e da produtividade esperada na época de colheita.

Além disso, no eixo *Product and Farm Management*, representamos os sistemas de produção logo após o processo de pesquisa e desenvolvimento de plantações. Esse aspecto do Modelo englobou todos os equipamentos, implementos agrícolas, sensores e tecnologias necessários para o cultivo de uma determinada área, bem como toda a estrutura de manuais de operação, planos de manutenção e *softwares* que auxiliam no controle produtivo de fazendas.

Os Sistemas de Gestão de Fazenda, do inglês *Farm Management Systems* (FMS), também foram representados no eixo *Product and Farm Management* do Modelo. Estes sistemas computacionais de gestão são extremamente importantes na gestão das fazendas, uma vez que facilitam a administração das terras ao fornecer relatórios de campo sobre as fases de desenvolvimento da cultura, prever e controlar a colheita, além de realizar o controle e gestão financeira da fazenda.

Ademais, nesse eixo, consideramos dois outros aspectos importantes relacionados à produtividade das lavouras: Dados meteorológicos (i.e., precipitação de chuvas, temperatura) e dados de recursos naturais (i.e., tipo de solo), uma vez que devem ser combinados para apoiar a decisão de escolha da cultura de cultivo e sua produção. Estes dois aspectos estão representados na Figura 17 dentro do retângulo *Cultivation Area* e serviram ao Modelo como possíveis entradas de informações no início do processo de pesquisa e desenvolvimento de plantações.

A legislação vigente no local de cultivo também foi outro aspecto ponderado no Modelo, já que as normativas da região onde as fazendas estão localizadas devem ser consideradas no planejamento da produção agrícola. Esse aspecto se liga ao Modelo pelo processo de sistemas de produção apontado neste eixo (*Product and Farm Management*).

No RAMI 4.0, o eixo vertical é denominado *Layers* e representa as camadas hierárquicas de informação, decompondo e representando perspectivas de entidades físicas

(i.e., mapas de dados, comportamentos). Dessa maneira, assim como no modelo RAMI 4.0, representamos no eixo vertical do Modelo seis camadas hierárquicas de informação. A primeira camada foi denominada *Business* e representou os processos de negócios organizacionais seguida da camada *Functional,* a qual trouxe as informações funcionais da organização. Na sequência, representamos as camadas *Information* e *Communication*, as quais retratam os dados necessários e o acesso às informações dos processos organizacionais, respectivamente. Por fim, as últimas duas camadas: *Integration* e *Asset* indicaram a integração entre o mundo digital e real.

Ainda, destacamos no Modelo proposto a interoperabilidade entre essas camadas de informação. Este destaque pode ser observado na Figura 17 pela flecha bidirecional presente no Modelo, a qual reforça a necessidade da interoperabilidade da comunicação entre todos os atores envolvidos. Ainda no mesmo eixo representamos os sistemas de apoio à tomada de decisão, os quais dependem da rastreabilidade de dados, desempenho de equipamentos e previsões derivadas de bancos de dados meteorológicos. Devido a isso, as camadas hierárquicas Funcionais e de Informação foram consideradas pelo Modelo.

O Modelo de Referência para Agricultura 4.0 criado integrou os conceitos 4.0 e suas tecnologias ao longo de todo ciclo de produção agrícola, agrupou e representou diferentes aspectos da agricultura inteligente em um único modelo. Dessa maneira, pode-se afirmar que o Modelo criado é capaz de orientar e auxiliar as tomadas de decisões estratégicas na implementação da Agricultura 4.0.

A criação do Modelo de Referência forneceu uma visão estratégica da Agricultura 4.0 e descreveu os aspectos mais importantes dos sistemas de produção agrícola inteligentes e possibilitou alcançar o atingimento do segundo objetivo específico desta pesquisa. É importante salientar que utilizamos o Modelo como base para o desenvolvimento dos demais artefatos que compõem a solução proposta.

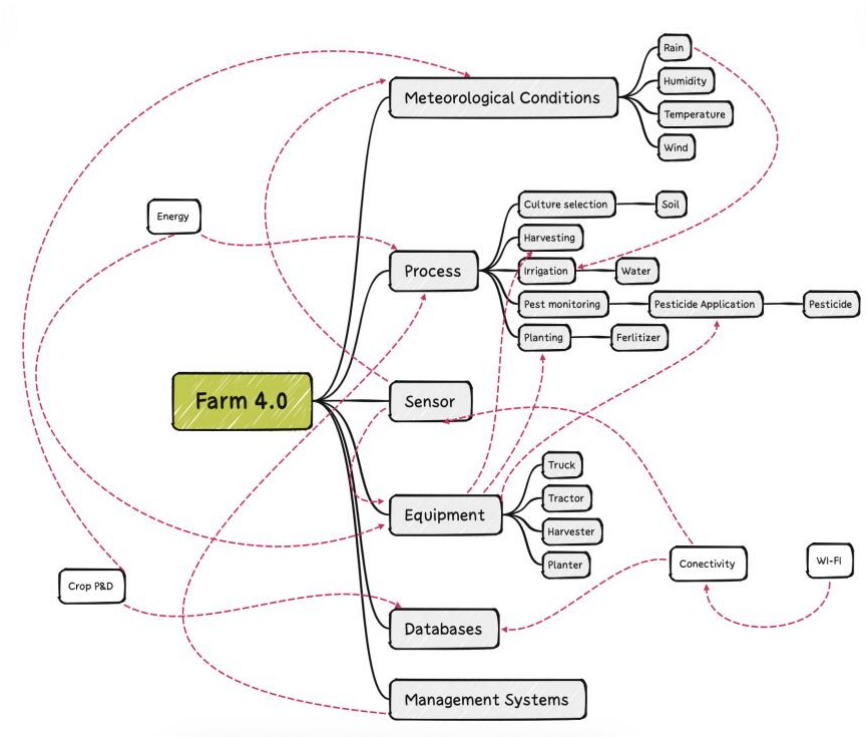
Dessa maneira, após criarmos o Modelo de Referência iniciamos o desenvolvimento do segundo artefato dessa pesquisa, o qual objetivou desenvolver um modelo de informação que retratasse todos os agentes de uma fazenda voltada para a Agricultura 4.0, e representasse todas as tecnologias e relações entre elas de forma integrada e universal.

4.1.2 Modelo Sistêmico de Fazenda 4.0

Após a criação do Modelo de Referência para Agricultura 4.0, foi possível orientar as tomadas de decisões estratégicas para a implantação da agricultura inteligente e deu-se início ao desenvolvimento do segundo artefato - Modelo Sistêmico de Fazenda 4.0, o qual é apresentado nesta seção. Este artefato objetivou representar de maneira gráfica a estrutura de uma fazenda orientada à agricultura 4.0, representando todos os seus agentes e todas as relações entre eles de maneira integrada.

Iniciamos o desenvolvimento desse Modelo com a criação de um mapa mental, o qual levou em consideração os agentes de uma fazenda e os conceitos apresentados pelo Modelo de Referência de Agricultura 4.0, que se relacionavam as fazendas de forma a facilitar a modelagem sistêmica de uma Fazenda Inteligente. A Figura 18 mostra o mapa mental com a representação dos agentes e suas relações presentes em uma Fazenda 4.0, que foi desenvolvido utilizando o software XMind.

Figura 18 - Mapa mental dos agentes de uma fazenda inteligente e suas relações



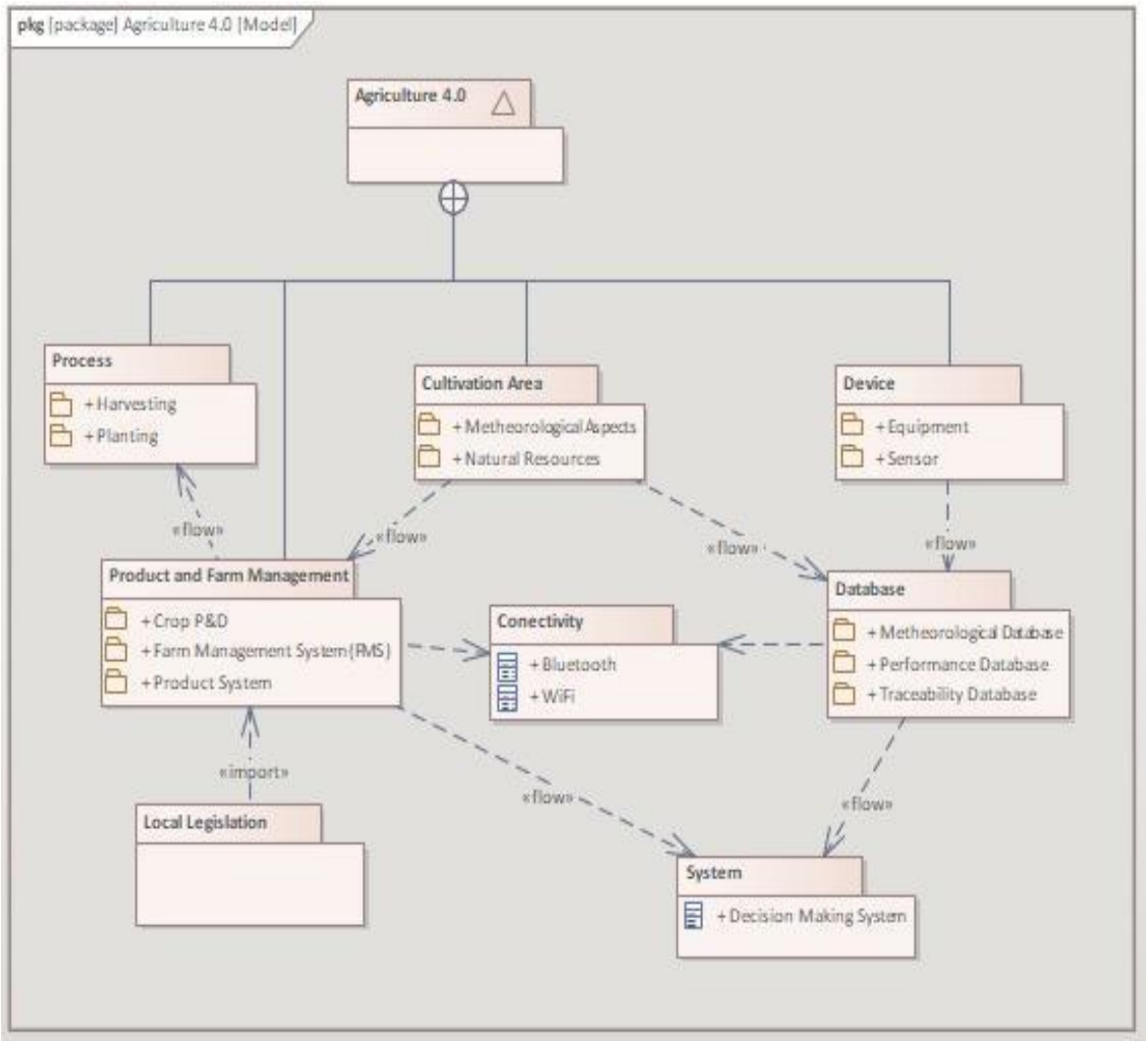
Fonte: A autora

Com os agentes e suas relações explicitadas foi possível iniciarmos o processo de criação do modelo proposto. Nesta etapa, fizemos uso do *software Enterprise Architect*

juntamente com o *plug-in* de extensão para a linguagem SysML, criando os diagramas que compuseram o Modelo Sistêmico. Optamos por utilizar para a representação do Modelo os diagramas de pacotes e blocos, uma vez que buscou-se retratar um modelo estrutural.

O diagrama de pacotes foi o primeiro a ser criado e representou a estrutura do modelo, retratando os agrupamentos lógicos e dependências entre os sistemas representados no Modelo de Referência de Agricultura 4.0. Assim, organizamos todos os eixos e aspectos presentes no Modelo em pacotes e as relações e dependências entre eles foram identificadas, conforme pode-se observar na Figura 19.

Figura 19 - Diagrama de pacotes



Fonte: A autora

O pacote denominado *Agriculture 4.0*, primeiro a ser criado, é um pacote genérico em que todos os outros estão conectados e está no diagrama para representar a ligação do

conceito de Agricultura 4.0 com todos os aspectos e eixos representados no Modelo de Referência. Após isso, adicionamos o pacote *Process* para representar os principais processos presentes na agricultura.

Na sequência, criamos o pacote *Database*. Conforme representado na Figura 19, ele recebe dados oriundos de outros dois pacotes: *Device* e *Cultivation Area*, compartilhando suas informações com o pacote *System*. Neste processo, o pacote *Product and Farm* Management, compartilha seus dados enviando-os para os pacotes *Process e System*.

Neste diagrama, representamos ainda a relação de dependência da utilização de tecnologias que oferecem conectividade para que os pacotes contidos em *Database* e *Product and Farm Management* desempenhem suas funcionalidades. Além disso, o pacote *Product and Farm Management* importa informações do pacote *Local Legislation* sobre a legislação das áreas de cultivo.

Após a criação do diagrama de pacotes, analisamos conceitos mais específicos relacionados às fazendas 4.0, objetivando representar todas as informações necessárias trocadas em uma fazenda inteligente. Dessa maneira, optamos pela criação de um diagrama de blocos, visto que objetivávamos especificar as partes do sistema, suas propriedades e relações.

Ao criarmos o Modelo Sistêmico de Fazenda 4.0, primeiro geramos os blocos que representavam os principais aspectos de uma fazenda inteligente, seguido da criação de relações entre eles. Durante o desenvolvimento do diagrama de blocos, verificamos diversas vezes todas as relações entre os blocos e suas propriedades, além de compará-las com o mapa mental apresentado na Figura 18, buscando assegurar que o Modelo continha todas as informações capazes de guiar na condução da implantação de uma Fazenda 4.0.

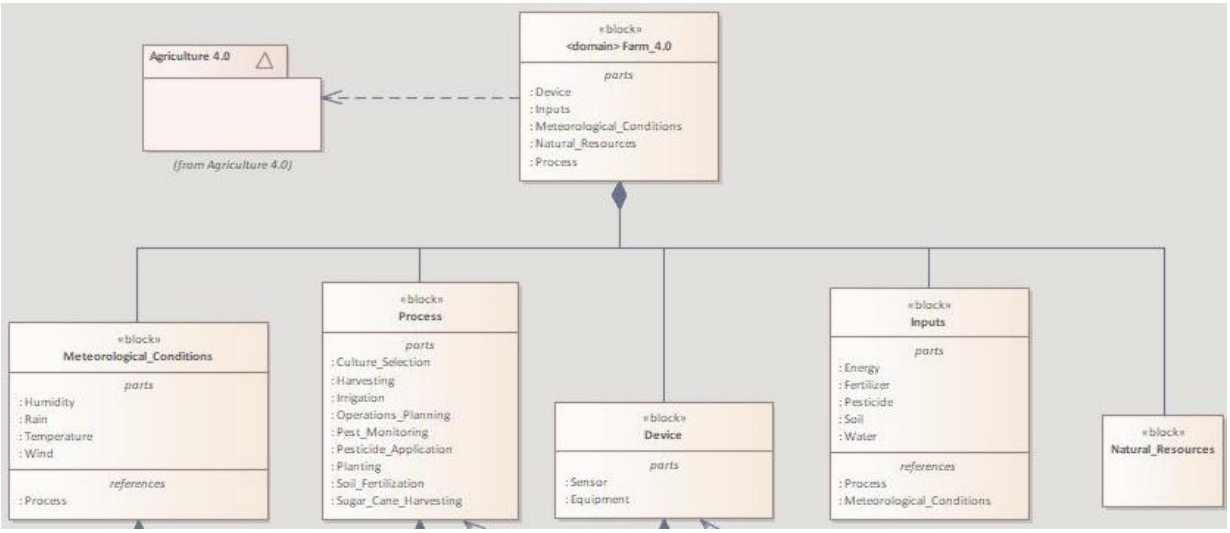
O primeiro bloco que criamos foi denominado *Farm_4.0* e representou as fazendas inteligentes. Ele possui uma relação de dependência com o diagrama de pacotes, a qual foi representada por uma flecha pontilhada que liga o bloco ao pacote *Agriculture 4.0*, conforme pode ser observado na Figura 20. Ao bloco *Farm_4.0* também incluímos a propriedade de associação com os seguintes blocos do diagrama: *Device*, *Inputs*, *Meteorological_Conditions*, *Natural_Resources* e *Process*.

A seguir definimos para cada um dos blocos ilustrados na Figura 20 suas propriedades. A propriedade de associação aparece nos blocos como *parts* e apresenta as trocas de dados e informações entre eles. No bloco *Process* é possível observar todos os processos presentes em uma fazenda, e no bloco *Inputs* todos os insumos desses processos são representados.

Outra propriedade de ligação entre os blocos, denominada *referência*, foi utilizada nessa fase. Pode-se observar essa relação entre os blocos *Process* e

Meteorological_Conditions, os quais servem de referência para o bloco *Inputs*. Por meio dessa ligação, indicamos a relação de dependência existente entre a quantidade de água para irrigação utilizada nas fazendas e o volume de chuvas medido no local, da mesma forma como a demanda por pesticidas depende do plano de aplicação que é posto em prática em cada propriedade.

Figura 20 - Bloco *Farm_4.0*



Fonte: A autora

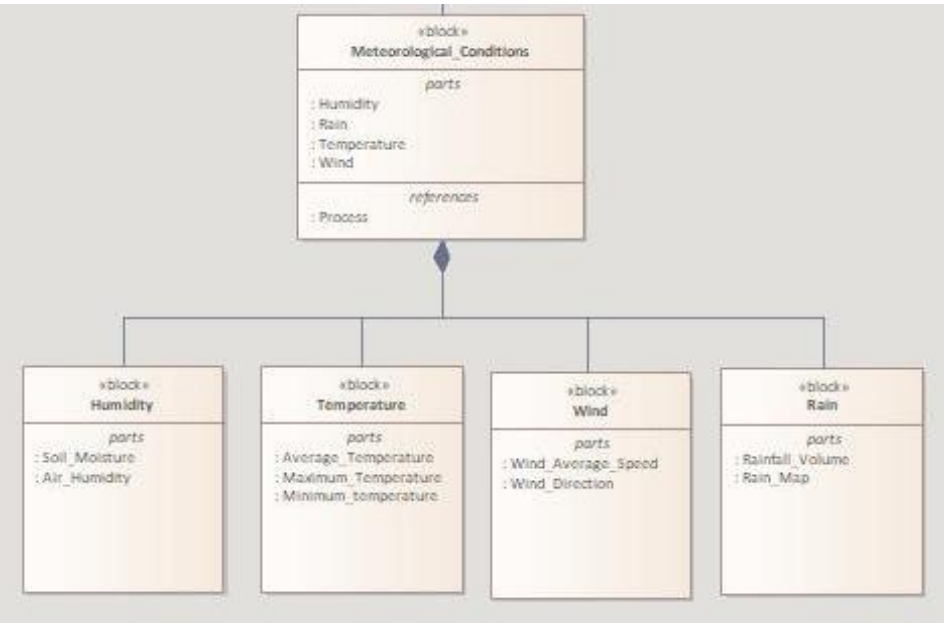
Associamos o bloco *Meteorological_Conditions* com outros quatro blocos (*Humidity*, *Temperature*, *Wind* e *Rain*), os quais definem e se conectam com os aspectos relacionados ao clima que devem ser considerados. Definimos esses aspectos considerando os principais parâmetros climáticos monitorados, como é o caso do bloco *Temperature*. Isto posto, julgamos necessária a representação e a utilização da temperatura média, máxima e mínima - já que são informações bastante utilizadas pelas fazendas. Isso também foi considerado para os blocos *Humidity*, *Wind* e *Rain*, conforme ilustra a Figura 21.

Por sua vez, o bloco *Device* foi criado para representar todos os equipamentos e sensores utilizados pelas fazendas. Assim, criamos os blocos *Equipment* e *Sensor* que foram ligados ao bloco *Device* através da relação de associação. O mesmo ocorreu com o bloco *Equipment*, que teve aspectos de manutenção, performance e subsistemas dos equipamentos a ele relacionados.

É importante salientar que ligamos ainda o bloco *Device* com os pacotes *Product and Farm Management* & Connectivity, oriundos do Modelo de Referência. A Figura 22 ilustra a representação do bloco *Device* e todas as relações que o sucedem no modelo.

Além disso, criamos o bloco *Sensor* para representar todos os sensores existentes em uma fazenda, sejam sensores de equipamentos, sejam meteorológicos. Realizamos também a ligação do pacote *Database* com esse bloco, para ilustrar o envio e armazenamento dos dados provenientes de sensores nas bases de dados.

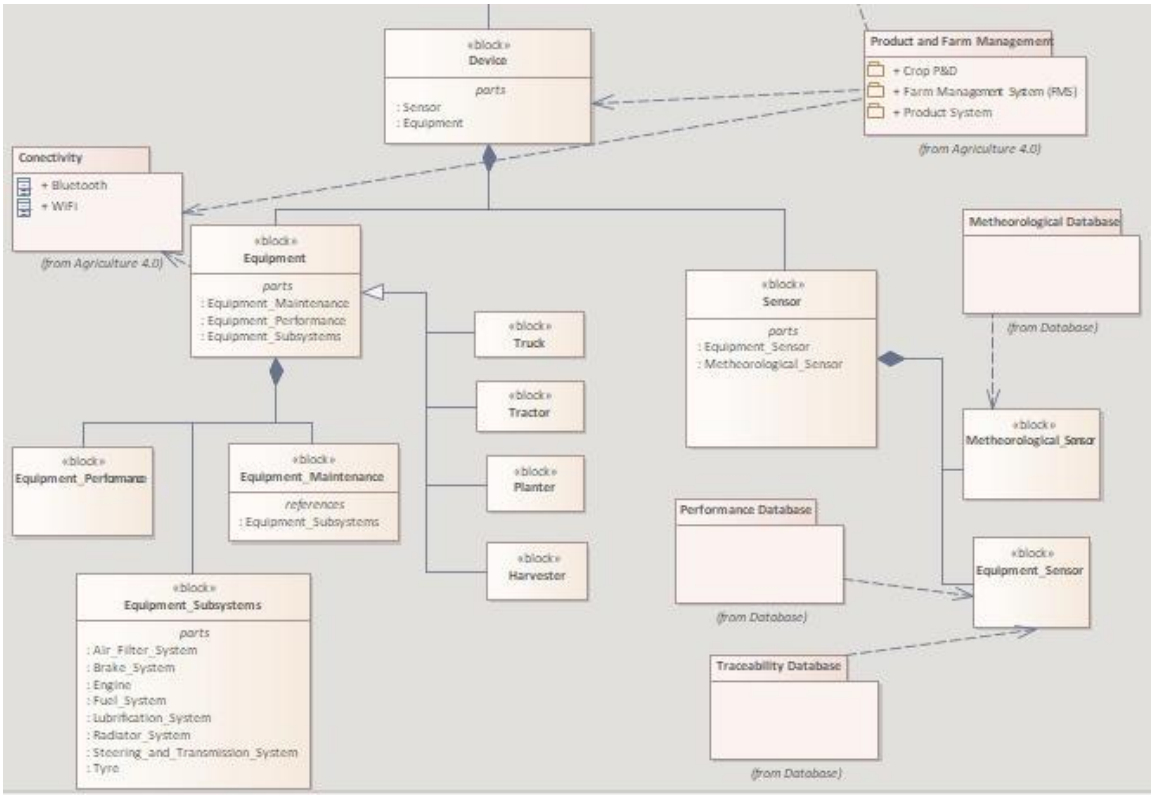
Figura 21 - Bloco *Meteorological_Conditions*



Fonte: A autora

Assim, representamos Modelo Sistêmico com todos os fluxos de informações presentes em uma fazenda inteligente, estruturando todas as trocas de dados e o cruzamento de informações essenciais, considerando tecnologias envolvidas e servindo de suporte para as tomadas de decisão táticas. Dessa forma, o Modelo Sistêmico, o qual teve seu desenvolvimento baseado no Modelo de Referência e no mapa mental apresentado, representou a Fazenda 4.0 de forma estruturada em uma linguagem que facilitou e orientou a criação do terceiro artefato, que compõe o objetivo deste trabalho. Assim, atingimos o terceiro objetivo específico proposto de construir um modelo sistêmico que retratasse uma fazenda genérica orientada à Agricultura 4.0. O modelo completo é apresentado no Apêndice C.

Figura 22 - Bloco *Device*



Fonte: A autora

Com os modelos de referência e sistêmico criados, pode-se iniciar o desenvolvimento do terceiro modelo que compõe a solução proposta neste trabalho.

4.1.3 Modelo de Informação

Nesta seção é apresentada a construção do Modelo de Informação, terceiro artefato que compõe a Solução proposta. Desenvolvemos o Modelo considerando o contexto de aplicação definido nas etapas anteriores do DSR: a manutenção de colheitadeiras de cana-de-açúcar, mais especificamente dos filtros de combustível.

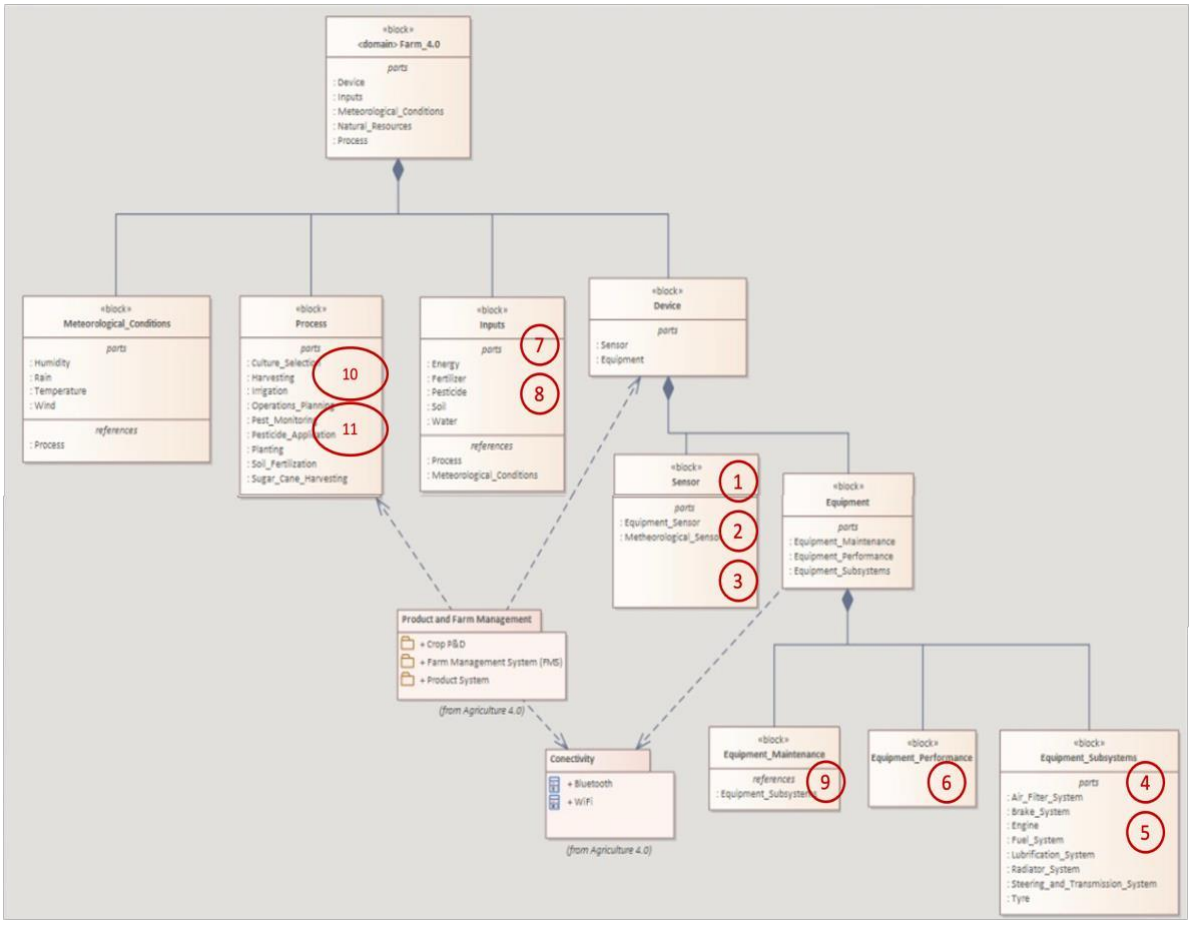
No Brasil poucas são as fazendas de cana-de-açúcar que contam com a conectividade necessária para que os sensores IoT presentes nos equipamentos agrícolas enviem os dados por eles coletados para um arquivo de registros na nuvem.

Portanto, devido a falta de conectividade no campo, a empresa parceira realizou a aquisição dos dados de entrada do modelo extraindo-os diretamente do CLP dos equipamentos e sensores, através de um *pendrive*, sendo exportados em formato de planilha de dados (xls). Por esse motivo, foi necessário elaborar a estrutura do modelo de tal forma que

fosse possível simular a entrada de dados, com os dados estruturados da mesma forma como ocorreria caso houvesse conectividade no campo.

Assim, diversas planilhas de dados contendo informações referentes a fazenda de cana-de-açúcar, seus equipamentos e sensores foram fornecidos pela empresa parceira para servirem de dados de entrada do modelo. Para assegurar que os dados obtidos junto à empresa parceira eram suficientes para representar o contexto de aplicação de uma Fazenda 4.0, relacionamos os dados fornecidos com os fluxos e as ligações entre os agentes representados no Modelo Sistêmico.

Figura 23 - Modelo Sistêmico x Dados fornecidos pela empresa parceira



Fonte: A autora

Um dos arquivos disponibilizados continha dezesseis tipos de dados das colheitadeiras (i.e., pressão de base de corte, altura de base de corte, velocidade e carga do motor). Foram fornecidos também arquivos com dados de sensores presentes no sistema de combustível, mais especificamente, pressão de combustível, e dados de localização dos equipamentos (i.e.,

latitude, longitude e ID de máquina). Esses conjuntos de dados puderam ser relacionados com o bloco *Sensor* do modelo sistêmico e foram representados na Figura 23 pelos números 1, 2 e 3, respectivamente.

Outros arquivos com a lista de componentes (i.e., *part numbers*) e dados específicos do sistema de combustível das colheitadeiras (i.e., capacidade do tanque de combustível e média de consumo de combustível) também foram disponibilizados. Constatamos que essas informações se relacionam com o bloco *Equipment_Subsystems*, conforme pode ser observado na Figura 23 pelos números 4 e 5.

Na mesma figura estão representadas pelo número 6 as informações relativas à performance das colheitadeiras (i.e., média de toneladas de cana colhidas por hectare e média de hectares colhidos por hora) as quais possuíam relação com o bloco *Equipment_Performance* do Modelo Sistêmico. Arquivos contendo o preço médio dos filtros de combustível e dados relativos às análises da qualidade do combustível (i.e., densidade e partículas presentes no diesel) também foram fornecidos e estão relacionados ao bloco *Inputs*. Esses arquivos estão representados na Figura 23 pelos números 7 e 8, respectivamente.

Ainda, ao analisarmos os arquivos recebidos, percebemos que nenhuma das informações fornecidas tinham relação com os blocos *Equipment_Maintenance* e *Process*. Dessa forma, optamos por criar três arquivos de dados: um contendo informações referentes à troca do filtro de combustível, um informações relativas à quantidade de filtros em estoque nas fazendas e outro com informações da quantidade de filtros em estoque nas lojas concessionárias. A criação desses arquivos foi baseada em relatórios de manutenção fornecidos pelas fazendas à empresa parceira. Já os arquivos com informações sobre estoque, tanto das fazendas quanto das lojas concessionárias, foram criados apenas para que os dados fossem representados no modelo, sem ter qualquer tipo de dado real como base. Esses arquivos encontram-se representados na Figura 23 pelos números 9, 10 e 11, respectivamente.

Observamos que o mesmo ocorria com o bloco *Meteorological_Conditions*. No entanto, não foi necessário criar nenhum arquivo de dados, uma vez que com o auxílio de uma interface de programação de aplicações (API) e utilizando-se o arquivo com os dados de localização da colhedora que havia sido fornecido pela empresa, seria possível obter informações relativas aos aspectos meteorológicos.

Ao todo, o conjunto de dados contou com trinta arquivos. Foram consideradas as seis colheitadeiras, portanto, o conjunto de dados de entrada do modelo contou com seis arquivos com dados das colheitadeiras, seis arquivos com dados de filtro de combustível e seis arquivos com a posição geográfica dos equipamentos. Ainda, como foram consideradas três

fazendas distintas pelo modelo, contou-se com três arquivos com dados sobre estoque e três arquivos com informações sobre manutenção das colheitadeiras.

Outros seis arquivos fizeram parte do conjunto de dados de entrada do Modelo: lista de componentes, dados do sistema de combustível das colheitadeiras, dados de performance das colheitadeiras, lista do preço médio dos filtros de combustível, dados relativos às análises da qualidade do combustível, e quantidade de filtros em estoque nas lojas concessionárias.

Por meio das planilhas fornecidas foi possível verificarmos os tipos de dados coletados, no entanto, os arquivos fornecidos eram pequenos, ou seja, continham um intervalo de tempo de coleta dessas informações de apenas um dia, portanto, para tornar possível a criação do Modelo, foi necessário avaliarmos e aumentarmos esse conjunto de dados (*dataset*). Para tal, identificamos, através do cálculo da média e desvio padrão, o comportamento dos dados contidos em cada coluna. Em seguida, considerando esse padrão, utilizamos a função *aleatório* no *software Excel* para criar mais dados e aumentar o *dataset*.

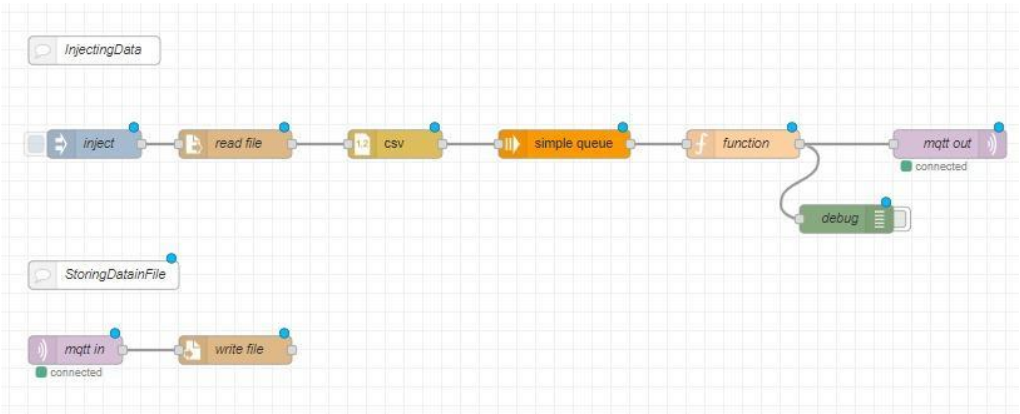
Apesar dos dados de entrada do Modelo não possuírem a mesma acuracidade que teriam caso todos os dados considerados fossem reais, ponderamos que o desenvolvimento do modelo não seria afetado, uma vez que o tipo de informação utilizado era real. Desse modo, constatamos que o conjunto de dados disponível, era suficiente para representar o contexto de aplicação escolhido dentro do conceito de fazenda 4.0 especificado pelo Modelo Sistêmico de Fazenda 4.0. Então, com a definição do conjunto de arquivos de dados que serviriam de entrada, foi possível iniciarmos a construção do modelo propriamente dito. As etapas e ferramentas utilizadas para a criação do Modelo são apresentadas na Figura 14.

A primeira etapa de criação do Modelo foi responsável por injetar os dados de entrada e criar arquivos contendo esses registros. Portanto, nessa etapa criamos dois fluxos no Node-RED: o primeiro injetou os dados na nuvem, o segundo recebeu esses dados e os armazenou em um arquivo como registros.

Considerando que os dados de entrada do modelo estavam distribuídos em trinta arquivos distintos, fez-se necessário organizar a criação desses fluxos, portanto, optamos pela criação de uma guia para cada um dos arquivos de dados. Assim, nessa primeira etapa de construção do Modelo, adicionamos ao Node-RED trinta guias, cada uma contendo dois fluxos cada.

A primeira guia criada foi denominada *Harvester01*. Nessa guia criamos os fluxos responsáveis pela injeção e geração do arquivo de registros contendo dados de uma das colheitadeiras. Esses dois fluxos podem ser observados na Figura 24.

Figura 24 - Fluxos de injeção e criação de arquivo de registro



Fonte: A autora

Os primeiros “nós” adicionados ao fluxo de injeção de dados, foram os “nós” *read file* e *csv*, os quais eram responsáveis por ler os arquivos que continham os dados e transformá-los em arquivos csv. A planilha de dados utilizada como entrada no fluxo continha dezesseis colunas, eram elas: *Time, Prim_Extr_Rpm, Prim_Extr_Rpm_Setp, Basecutter_pressure, Chopper_Hydr_Press, Eng_Load, Basecutter_height, Cruise_Command, Ground_Speed, BaseCutter_Rpm, Chopper_Rpm, Machine_Seconds, Elevator_Seconds, Fuel_Level_Pct, Machine_Model* e *Machine_ID*.

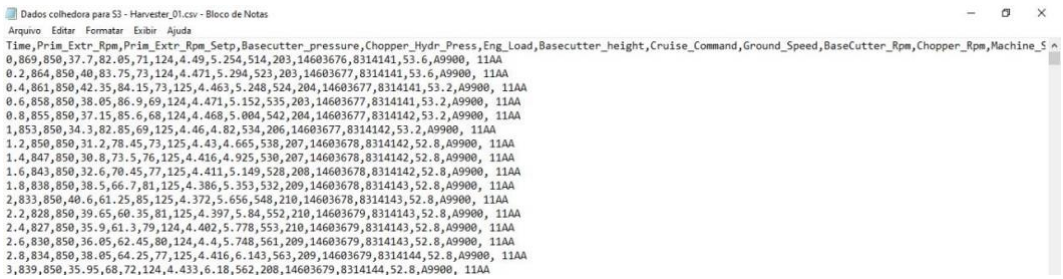
A primeira coluna, apesar de ser denominada *Time*, do inglês tempo, não apresentava nenhuma medida de tempo, apenas uma sequência crescente de números que aumentavam de dois em dois décimos. A segunda e terceira colunas apresentavam as rotações por minuto do extrator primário, sendo a primeira a rotação real e a segunda a rotação programada. Já a quarta coluna apresentava o valor da pressão na base de corte da colheitadeira e a quinta coluna o valor da pressão nas facas de corte.

As colunas *Eng_Load*, *Basecutter_height* trouxeram dados de carga do motor e altura da base de corte, seguidas pelas colunas *Cruise_Command* que apresenta informação da velocidade programada no piloto automático, e *Ground_Speed* que apresenta a velocidade real da colheitadeira. A décima coluna apresenta as rotações por minuto da base de corte, seguida da coluna *Chopper_Rpm,* a qual trouxe as rotações das facas de corte.

O arquivo ainda informa dados de horímetro da colheitadeira, representado pela coluna *Machine_Seconds,* e tempo de funcionamento de elevador (*Elevator_Seconds*). O percentual do nível de combustível também é informado na coluna *Fuel_Level_Pct*. Por fim, as colunas *Machine_Model* e *Machine_ID* apresentam o modelo e o código de identificação da colheitadeira.

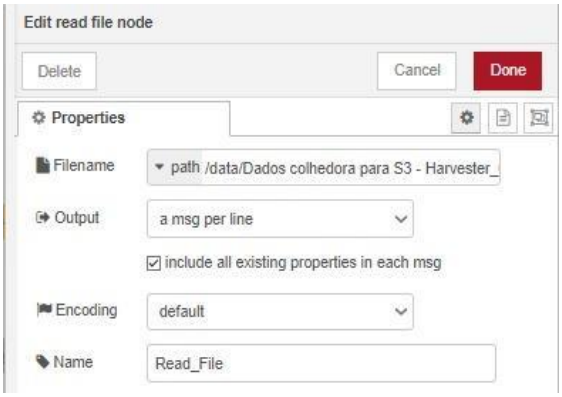
Ao configurar o nó *read file* percebemos que para o Node-RED realizar a injeção dos dados, seria necessário transformar a planilha de dados (*xls*) em um arquivo de dados separados por vírgulas (*csv*). Após essa transformação, especificamos o caminho do arquivo que seria utilizado como entrada de dados no fluxo (i.e., */data/Dados_colheitadeira-Harvester01.csv*) e como a leitura desse arquivo seria realizada. Quanto ao modo como seria realizada a leitura do arquivo, optamos por gerar uma mensagem a cada linha do arquivo que fosse lida. As FigurasFigura 25 e Figura 26 exemplificam respectivamente: o arquivo *csv*, onde a primeira linha corresponde ao nome das colunas e as demais linhas, aos registros, e a configuração do nó *read file*.

Figura 25 - Exemplo de dados de entrada em arquivo *.csv*



Fonte: A autora

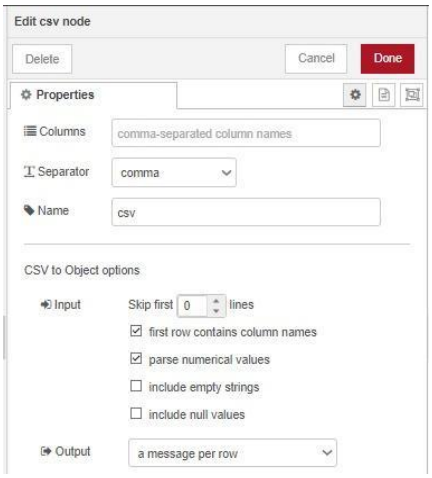
Figura 26 - Exemplo de Configuração do nó *read file*



Fonte: A autora

Já o nó *csv* foi configurado para declarar que os dados das colunas estariam separados por vírgulas e que a primeira linha do arquivo continha o nome das colunas dos valores de dados, conforme mostra a Figura 27.

Figura 27 - Detalhe da configuração do nó *csv*

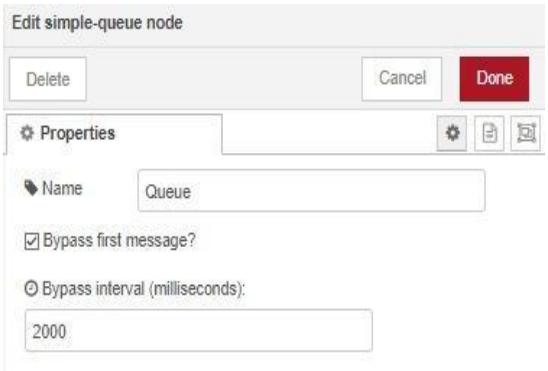


Fonte: A autora

Dando continuidade a criação do fluxo de injeção de dados, por se tratar de dados provenientes de sensores IoT presentes na colheitadeira, acrescentamos o nó *simple queue* entre os "nós" *csv* e *function*. Esse nó foi utilizado para que a frequência de envio de dados pelos sensores fosse considerada, ou seja, ele possibilitou emular o tempo entre o envio de cada registro pelos sensores IoT.

Ao utilizar o nó *simple queue* uma fila era criada, fazendo com que os registros, os quais eram enviados do nó *csv* e recebidos de uma só vez na entrada do nó, aguardassem e fossem enviados um a um, em um determinado espaço de tempo. Esse intervalo de envio de mensagens foi configurado levando em consideração o intervalo real de envio de mensagens pelos sensores IoT das colheitadeiras. Assim, configuramos o nó para que as mensagens fossem enviadas uma a uma a cada dois segundos, conforme mostra a Figura 28.

Figura 28 - Configuração do nó *Simple Queue*



Fonte: A autora

Ao analisar os dados de entrada do fluxo, percebemos a falta de dados temporais (i.e., *Date* e *Timestamp*) na planilha de dados da colheitadeira e que a primeira coluna da planilha (*Time*) não acrescenta nenhum dado relevante para o Modelo. Por isso, na sequência, incluimos o nó *function* ao fluxo.

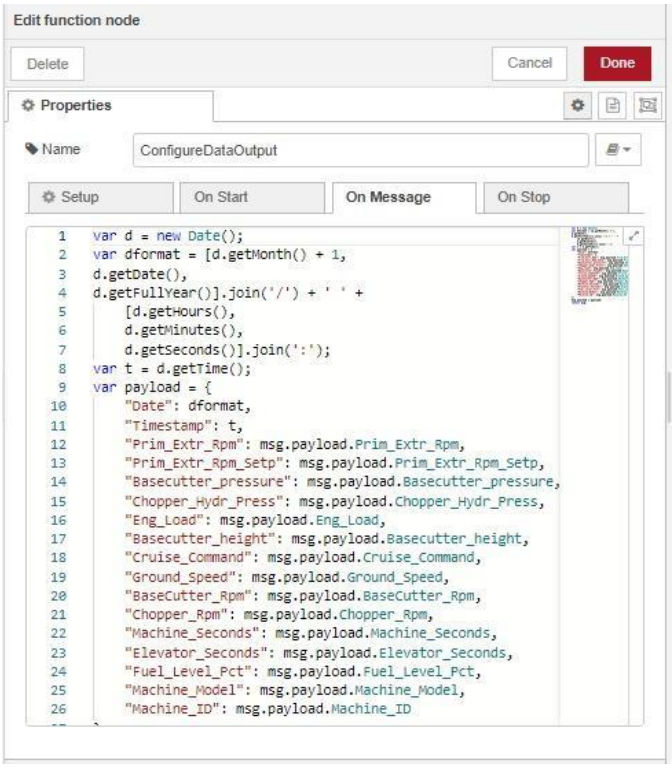
O nó *function* permitiu que um código JavaScript fosse executado quando as mensagens passassem por ele, possibilitando configurar a sintaxe e filtrar os dados das mensagens. Dessa forma, a primeira coluna *Time* foi ocultada e as colunas *Date* e *Timestamp* foram acrescentadas às mensagens. Na Figura 29 é apresentada a configuração do nó *function*.

Na primeira parte do código JavaScript declaramos três variáveis (i.e., *d*, *dformat* e *t*). A variável "*d"* foi responsável por identificar a data em que a mensagem estava sendo criada, já a variável *dformat* utilizou a data informada por *d* para formatar a informação de tal forma que o dia, mês, ano fossem separados por "/" e que o horário com hora, minuto e segundos fossem separados por ":", para que a data apresentada na mensagem fosse de fácil compreensão (e.g., 12/01/2022 12:05:36). Por sua vez, a variável "*t*" foi responsável por acrescentar um carimbo de data e hora (*timestamp*) às mensagens.

Definimos ainda a variável *payload*, contendo todos os tipos de dados que a mensagem deveria conter. Foram declarados os nomes das colunas da planilha com os dados de entrada (exceto Time), além das colunas *Date* e *Timestamp*. Com isso, as mensagens ficaram contendo ao todo dezessete tipos de dados.

Ao longo da criação dos fluxos também utilizamos o nó *debug*, o qual tornou possível visualizar e conferir as mensagens que estavam sendo enviadas. Incluímos um nó *debug* após o nó *function*, tornando possível a visualização das mensagens com os dezessete tipos de dados, os quais haviam sido declarados no nó *function*, bem como o envio das mensagens a cada dois segundos, definido no nó *simple queue*. A Figura 30 ilustra essas mensagens.

Figura 29 - Configuração do nó *Function*



Fonte: A autora

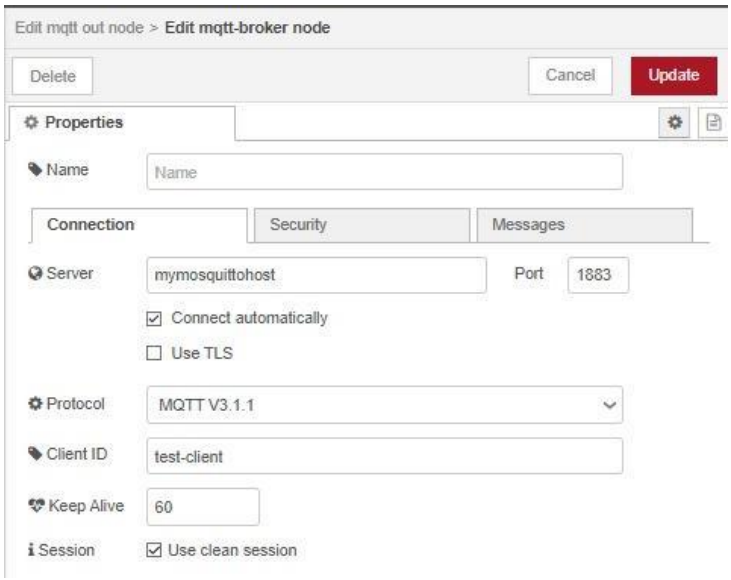
Figura 30 - Visualização de mensagens (nó *debug*)

Fonte: A autora

Por fim, para que as mensagens fossem publicadas, acrescentamos o nó *mqtt* out ao fluxo de injeção de dados. Esse nó teve como objetivo conectar o fluxo ao *broker* Mosquitto e enviar as mensagens contendo os dados para a nuvem, através de um protocolo MQTT. Para

se conectar ao *broker* foi necessário especificar servidor e a porta utilizada, conforme mostra a Figura 31.

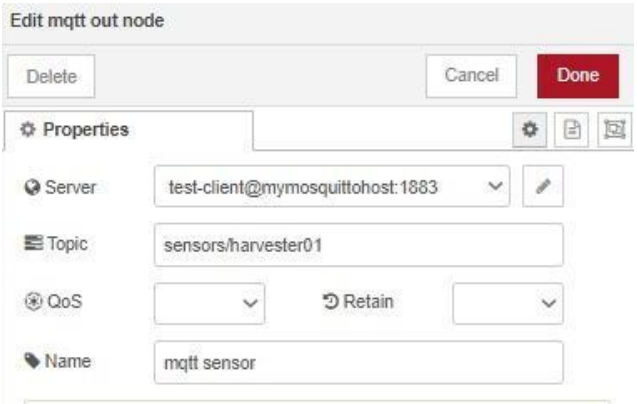
Figura 31 - Nó *mqtt out* – configuração do servidor



Fonte: A autora

Além de configurar o servidor, foi necessário definirmos o tópico no qual as mensagens seriam publicadas. A Figura 32 exemplifica a configuração desse nó. Como se tratava do fluxo presente na guia *Harvester01*, definimos o seguinte tópico: *sensors/harvester01*.

Figura 32 - Nó *mqtt out*



Fonte: A autora

Com a inclusão desse nó concluímos a criação do fluxo responsável pela injeção dos dados da colheitadeira *Harvester01*. Como os arquivos como dados de entrada das outras

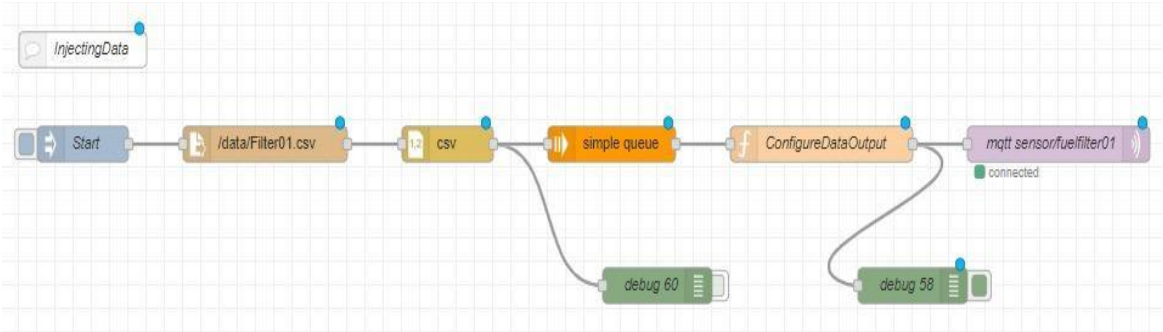
colheitadeiras continham os mesmos tipos de informação, esse fluxo foi replicado para as outras cinco colheitadeiras. Assim criamos outras cinco guias: *Harvester02*, *Harvester03*, *Harvester04*, *Harvester05* e *Harvester06*.

O que difere o fluxo de injeção de dados de cada uma das seis guias que representam a entrada de dados das colheitadeiras são as configurações presentes no nó *read file* e *mqtt out*. O caminho com o nome do arquivo de entrada no nó *read file* é diferente para todas as guias, ou seja, em cada fluxo de injeção de dados é especificado um nome de arquivo diferente, que corresponde a colheitadeira a qual a guia representa (i.e., guia *Harvester02 - /data/Dados_colheitadeira-Harvester02.csv*, guia *Harvester03 - /data/Dados_colheitadeira-Harvester03.csv*).

Outra diferença está no tópico de publicação especificado no nó *mqtt out* de cada guia, ou seja, definiram-se ao todo seis tópicos distintos: *sensors/harvester01, sensors/harvester02, sensors/harvester03, sensors/harvester04, sensors/harvester05, sensors/harvester06*. A definição de tópicos diferentes fez com que, no segundo fluxo, fossem criados arquivos individuais de registros de dados para cada colheitadeira.

Os fluxos de injeção de dados de sensores de pressão de combustível das colheitadeiras foram criados de maneira similar ao fluxo de injeção de dados das colheitadeiras. Para representar o sensor de pressão do filtro de combustível de cada uma das colheitadeiras representadas no modelo, criamos seis guias: *Fuel_Filter01*, *Fuel_Filter02*, *Fuel_Filter03*, *Fuel_Filter04*, *Fuel_Filter05*, *Fuel_Filter06*. A Figura 33 exemplifica o fluxo de injeção de dados da guia *Fuel_Filter01*.

Figura 33 - Fluxo de injeção de dados da guia *Fuel_Filter01*



Fonte: A autora

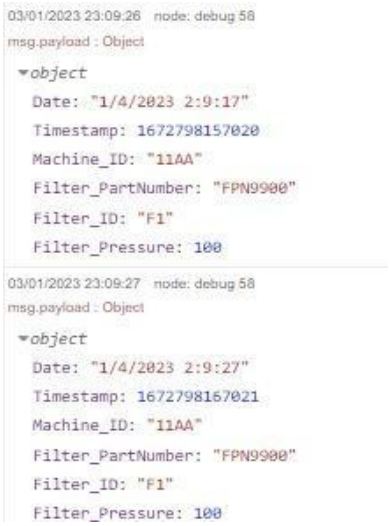
Além de alterar o nome dos arquivos de entrada no nó *read file* de cada uma das seis guias, foi necessário realizar alterações nos "nós" *simple queue, function* e *mqtt out*. No caso

do sensor de pressão de combustível, os registros deveriam ser enviados a cada dez segundos, assim configuramos o nó *simple queue* para que o intervalo entre as mensagens fosse de dez mil milissegundos.

A planilha com dados dos sensores de pressão de combustível contava com quatro colunas de informação: *Machine_ID*, *Filter_PartNumber*, *Filter_ID* e *Filter_Pressure*. A primeira coluna identifica em qual colheitadeira estava instalado o sensor, a segunda informava qual era o tipo de filtro instalado no equipamento, já a terceira identificava de que filtro os dados se tratam. Por fim, a quarta e última coluna do arquivo apresentava a pressão no sistema de combustível da colheitadeira medida pelo sensor.

Ao analisarmos os dados presentes no arquivo advindo dos sensores, percebemos que, assim como nos dados das colheitadeiras, não havia nenhum dado temporal, portanto, através do código JavaScript no nó *function*, acrescentamos informações de data e *timestamp* às mensagens. Através do nó *debug* (*debug*58) é possível visualizar as mensagens de sensores do filtro de combustível das colheitadeiras após passarem pelo nó *function*, como mostra a Figura 34. Ainda, o tópico especificado no nó *mqtt out* foi diferente para cada guia (i.e., *sensor/fuelfilter01*, *sensor/fuelfilter02*, *sensor/fuelfilter03*, *sensor/fuelfilter04*, *sensor/fuelfilter05*, *sensor/fuelfilter06*).

Figura 34 - Visualização de mensagens - Nó *debug*58



Fonte: A autora

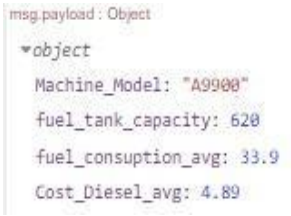
A estrutura dos fluxos utilizados na injeção dos dados das planilhas que continham informações acerca de componentes presentes nos equipamentos, dados específicos do sistema de combustível das colheitadeiras, informações relativas à performance das

colheitadeiras, bem como o preço médio dos filtros de combustível e dados relativos às análises da qualidade do combustível foi a mesma dos outros fluxos apresentados. Para a injeção desses dados criamos outras cinco guias, são elas, respectivamente: *Part_Number*, *Fuel_Information*, *Harvester_Performance*, *Fuel_Filter_Price_AVG*, *Diesel_Quality*. As Figuras Figura 35 até Figura 39 exemplificam os dados das mensagens injetadas em cada uma dessas cinco guias.

Figura 35 - Exemplo de mensagem guia *Part_Number*

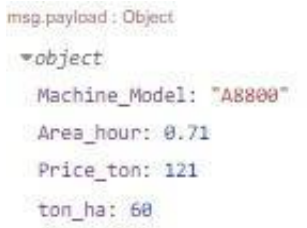
Fonte: A autora

Figura 36 - Exemplo de mensagem guia *Fuel_Information*



Fonte: A autora

Figura 37 - Exemplo de mensagem guia *Harvester_Performance*

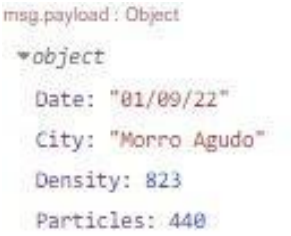


Fonte: A autora

Figura 38 - Exemplo de mensagem guia *Fuel_Filter_Price_AVG*

Fonte: A autora

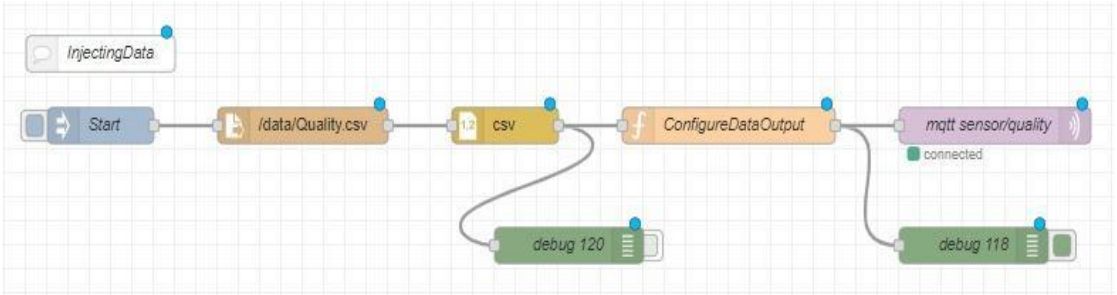
Figura 39 - Exemplo de mensagem guia *Diesel_Quality*



Fonte: A autora

Alteramos apenas os arquivos de entrada de dados (nó *read file*), os dados definidos para serem impressos nas mensagens (nó *function*) e o tópico MQTT (nó *mqtt out*). No entanto, diferente dos dados provenientes de sensores, nessas guias não era necessário que as mensagens fossem enviadas em um intervalo de tempo pré-determinado, por esse motivo, optamos por não utilizar o nó *simple queue* nesses fluxos. A Figura 40 exemplifica esses fluxos de injeção através da guia *Diesel_Quality*.

Figura 40 - Fluxo para injeção de dados - guia *Diesel_Quality*



Fonte: A autora

As guias *Field01_Maintenance*, *Field02_Maintenance*, *Field03_Maintenance*, *Store_Stock*, *Field01_Stock*, *Field02_Stock* e *Field03_Stock* também seguiram a mesma estrutura de "nós" para injeção de dados utilizados para os demais dados. Essas guias foram

responsáveis pela injeção de dados relativos às trocas dos filtros de combustível em cada fazenda, quantidade de filtros de combustível em estoque nas lojas concessionárias, e quantidade de filtros de combustível em estoque nas fazendas consideradas.

Da mesma maneira como nos últimos cinco fluxos criados, modificamos apenas as configurações do nó *read file*, no qual os arquivos de entrada de dados foram alterados, os dados definidos pelo nó *function* para serem impressos nas mensagens e o tópico MQTT no nó *mqtt out*. Nesses fluxos também optamos por não utilizar o nó *simple queue*. As Figura 41, Figura 42 e Figura 43, exemplificam as mensagens dos fluxos de injeção de dados das últimas sete guias criadas.

Figura 41 - Exemplo mensagem guia *Field01_Maintenance*

Fonte: A autora

Figura 42 - Exemplo mensagem guia *Store_Stock*

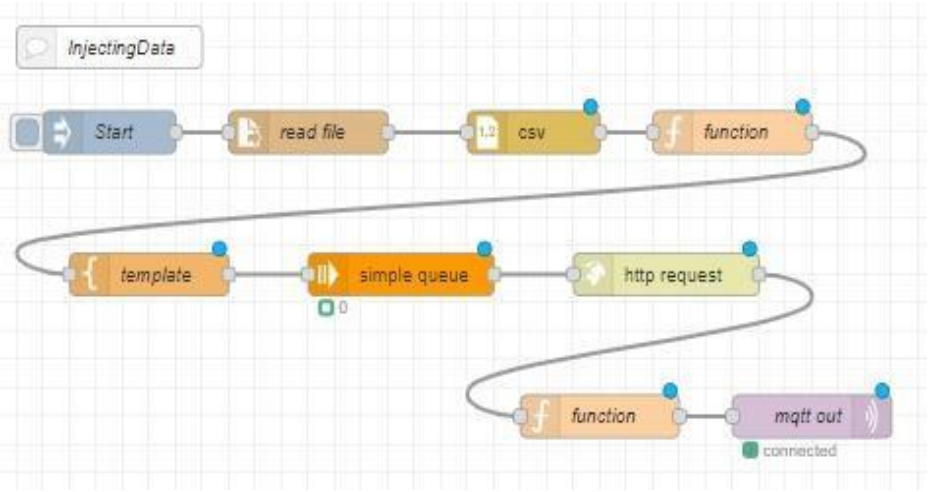
Fonte: A autora

Figura 43 - Exemplo mensagem guia *Field01_Stock*

Fonte: A autora

Além dos fluxos de injeção apresentados anteriormente, desejava-se acrescentar dados como temperatura, umidade, pressão e velocidade do vento ao Modelo de Informação, porém esses dados não estavam presentes em nenhuma das planilhas fornecidas. A solução encontrada para adicionar esses dados ao modelo foi buscar informações via protocolo *web* e injetá-los no Modelo. Para isso, utilizamos um dos API's disponibilizados pela OpenWeather, o qual possibilitou obter através das planilhas que continham dados de posição geográfica das colheitadeiras os dados meteorológicos dos locais onde elas encontravam-se. Portanto, foi necessária a criação de uma estrutura de fluxo diferente da utilizada para a injeção dos outros dados no Node-RED, a qual pode ser observada na Figura 44.

Figura 44 - Fluxo de injeção de dados meteorológicos



Fonte: A autora

Os primeiros três "nós" adicionados - *injection*, *read file* e *csv* - seguiram a mesma lógica e configuração adotada para os demais fluxos. No nó *read file*, a planilha de dados de entrada utilizada continha as seguintes colunas: coordenadas de latitude, coordenadas de longitude e identificação da colheitadeira.

O API *Current Weather Data*, disponibilizado pela organização *OpenWeather*, possibilita fazer chamadas contendo dados de longitude e latitude e receber os dados meteorológicos da localização de maneira gratuita. Para utilizá-lo foi necessário realizarmos um cadastro e solicitarmos uma chave de acesso. Após o recebimento da chave de acesso foi possível começar a realizar as chamadas.

As chamadas são realizadas através de um endereço URL fornecido pela organização que desenvolveu a API. Nas chamadas, para que fosse possível acessar os dados da API foi

necessário adicionarmos a chave de acesso (*API key*) ao endereço URL e substituirmos os dados de latitude e longitude pelas coordenadas do local o qual os dados meteorológicos estavam sendo buscados. A Figura 45 apresenta o endereço URL de chamada do API, onde entre "chaves" estão representados os dados que deveriam ser substituídos.

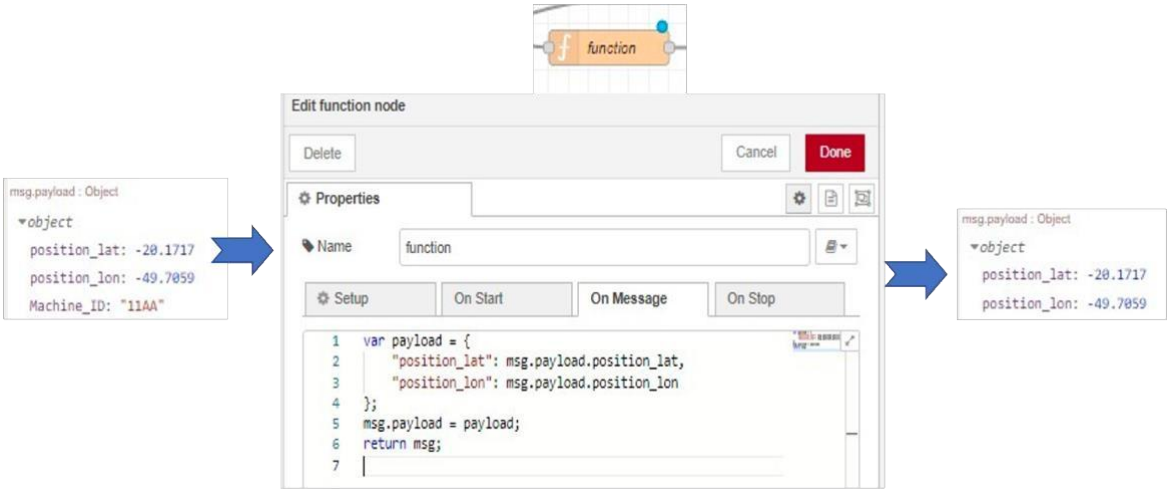
Figura 45 - Endereço URL de chamada do API

Fonte: OpenWeatherMap (2022).

As chamadas API utilizaram como entrada os dados de latitude e longitude presentes na planilha, assim a cada nova chamada, a próxima linha do arquivo que continha esses dados deveria ser lida. Então, o fluxo do Node-RED precisou ser criado de forma a possibilitar que a URL de chamada fosse atualizada automaticamente a cada nova chamada.

Para que fosse possível realizar a busca com o auxílio do API, necessitava-se apenas dos dados da posição da colheitadeira (i.e., latitude e longitude), por isso, nesse primeiro momento de construção do fluxo, através da utilização nó *function*, a coluna de identificação da colheitadeira foi ocultada das mensagens, conforme mostra a Figura 46.

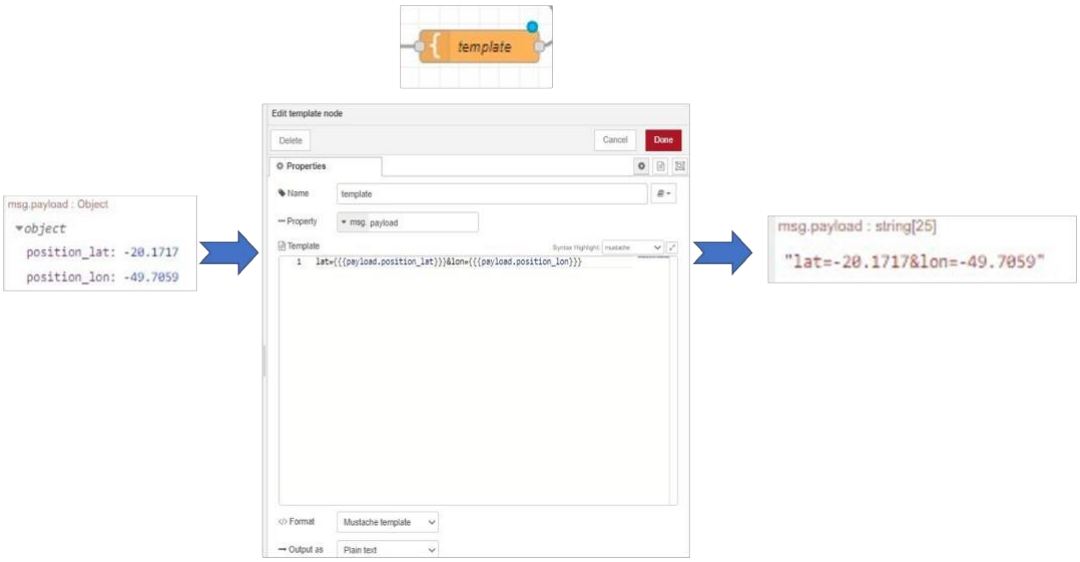
Figura 46 - Exemplo de mensagem antes e após o nó *function*



Fonte: A autora

O quinto nó que adicionamos ao fluxo de injeção dos dados meteorológicos foi o nó *template*. Esse nó define um modelo que pode ser preenchido utilizando as propriedades de uma mensagem e traz como saída um texto no formato de linguagem Mustache. A Figura 47 exemplifica a transformação da mensagem ao passar pelo nó.

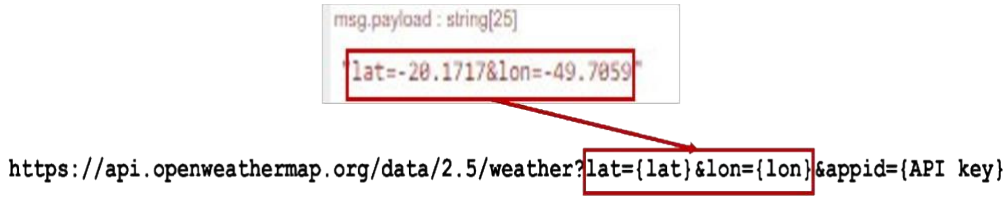
Figura 47 - Exemplo de mensagem antes e após o nó *template*



Fonte: A autora

A transformação da mensagem, realizada no nó *template*, serviu para que a mensagem que saia do nó possuísse exatamente a mesma sintaxe da parte da URL de chamada do API que continha os dados de latitude e longitude, conforme pode ser observado na Figura 48.

Figura 48 - Nó *template* e URL para chamada do API



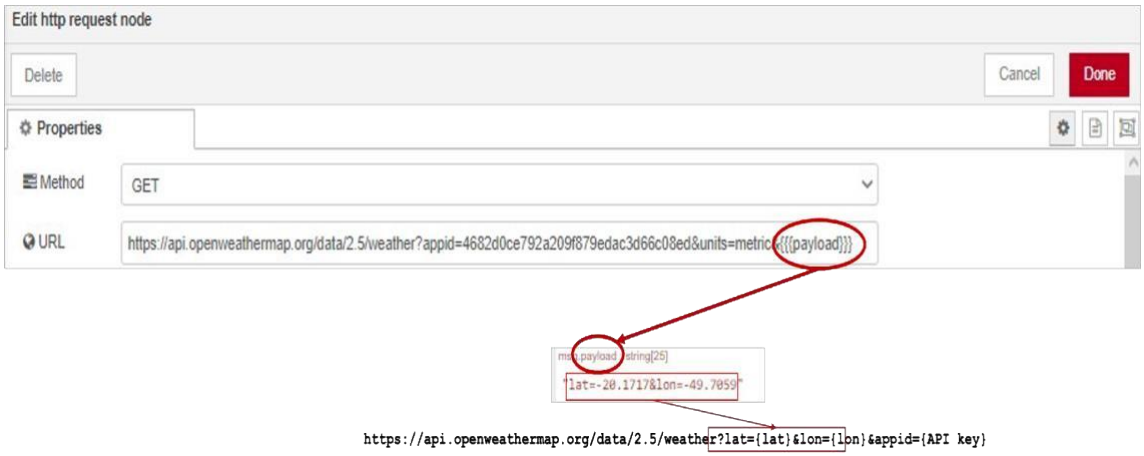
Fonte: A autora

Ao construirmos esse fluxo consideramos que as colheitadeiras de cana se deslocavam em velocidade relativamente baixa, o que fazia com que sua posição geográfica não sofresse alteração significativa em pequenos intervalos de tempo, assim como os aspectos meteorológicos que estavam sendo considerados. Portanto, decidimos acrescentar o nó *simple*

queue ao fluxo e configurar o envio de mensagem de dados de posição geográfica das colheitadeiras a cada trinta minutos. Assim, o intervalo entre cada chamada para o API seguiu essa frequência.

Na sequência adicionamos ao fluxo de injeção de dados meteorológicos o nó *http request*, o qual foi responsável por realizar a solicitação ao API e receber a resposta. Nesse nó foi configurado o endereço URL para as chamadas do API. Para que o endereço fosse dinâmico, substituíram-se as informações de latitude e longitude pela mensagem configurada no nó *template*, conforme é apresentado na Figura 49.

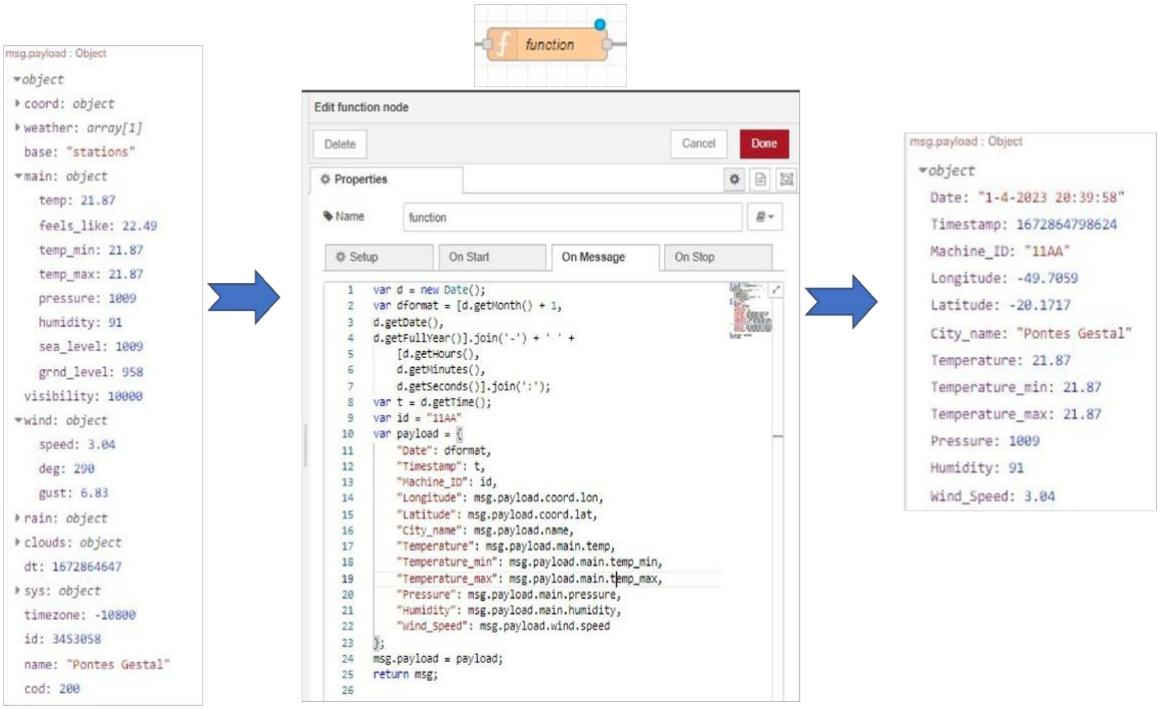
Figura 49 - Configuração do nó *http request*



Fonte: A autora

Após, acrescentamos o nó *debug* ao fluxo para verificar os dados presentes nas mensagens que estavam sendo recebidas através da API. Assim, após visualizar os dados, o nó *function* foi adicionado ao fluxo para que fosse feita uma filtragem dessas informações, e por fim, adicionamos o nó *mqtt out* ao fluxo. A Figura 50 exemplifica a filtragem dos dados das mensagens ao passar pelo nó.

Figura 50 - Exemplo de mensagem antes e após o nó *function*



Fonte: A autora

Devido ao cenário de aplicação contar com seis colheitadeiras, criamos seis guias com esse fluxo de injeção de dados meteorológicos. Foram elas: *Weather_Harvester01*, *Weather_Harvester02*, *Weather_Harvester03*, *Weather_Harvester04*, *Weather_Harvester05*, *Weather_Harvester06*.

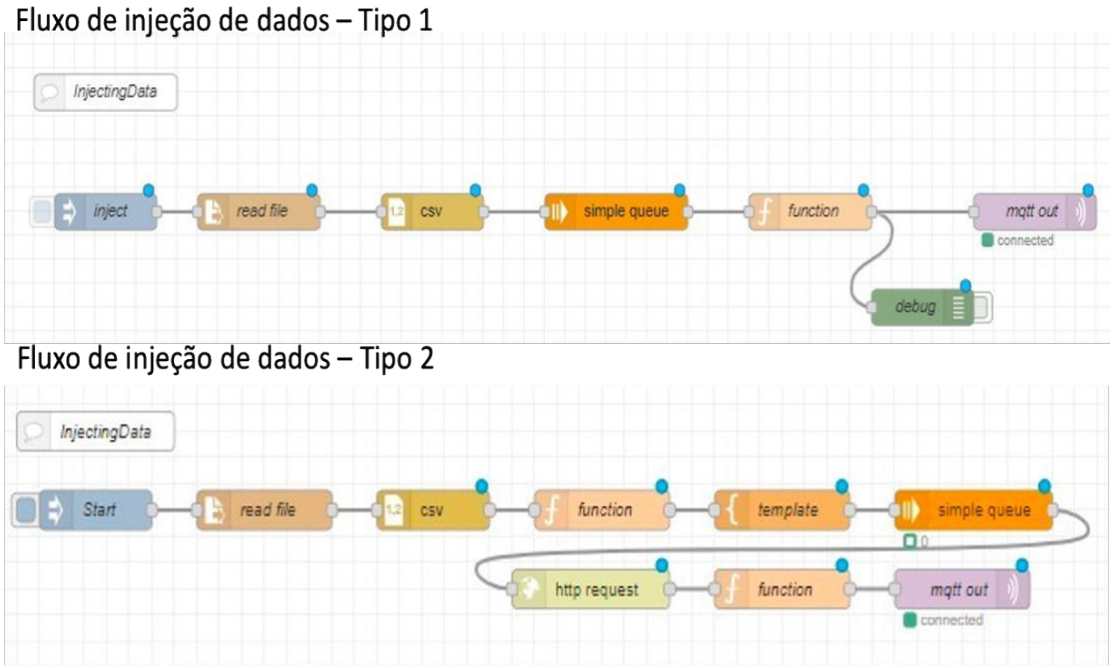
Em cada uma dessas seis guias, foi especificado no nó *read file* o nome do arquivo que continha os dados de entrada (i.e., posição geográfica) relativos à colheitadeira que estava sendo considerada na guia. Da mesma forma, no nó *mqtt out* foram especificados um tópico para cada guia.

Por fim, adicionamos em todos os fluxos criados o nó *injection*, o qual tem a função de acionar o fluxo. Esse nó poderia ser acionado manualmente ao clicar no botão à esquerda do nó dentro do editor, ou poderia ser configurado para ser acionado automaticamente. Como nos fluxos criados os dados das planilhas deveriam ser injetados apenas uma vez, manteve-se a configuração de acionamento manual do nó.

Portanto, definimos ao todo trinta fluxos de injeção de dados, um para cada arquivo com dados de entrada. Pode-se afirmar que se criou dois tipos de fluxos de injeção de dados: um que utilizou diretamente as planilhas de dados como entrada – Tipo 1, e outro que utilizou as planilhas de dados como meio para obter dados de entrada – Tipo 2. Os dois tipos de fluxo

de injeção de dados utilizados no modelo podem ser observados na Figura 51. O Apêndice D apresenta as informações contidas nas planilhas dados.

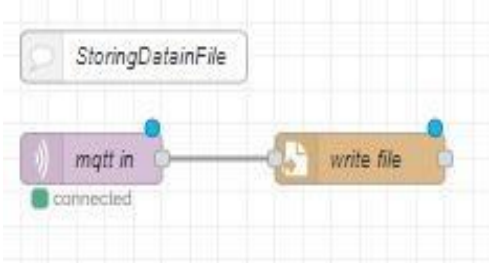
Figura 51 - Tipos de fluxo de injeção de dados utilizados no Modelo



Fonte: A autora

Ao finalizarmos a criação dos fluxos de injeção de dados, iniciamos a construção do segundo fluxo que compõe a primeira etapa de desenvolvimento do Modelo de Informação. Assim, geramos o fluxo o qual recebeu os dados provenientes do primeiro fluxo e os armazenou em um arquivo como registros. Conforme ilustra a Figura 52, na criação desse fluxo utilizamos os seguintes "nós": *mqtt in* e *write file*.

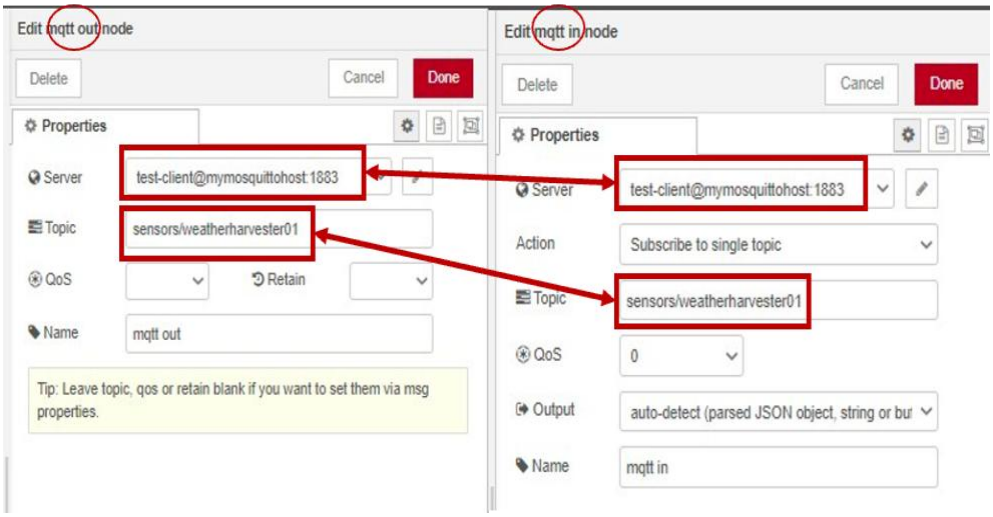
Figura 52 - Fluxo de criação de arquivo de registros



Fonte: A autora

Iniciamos o desenvolvimento desse segundo fluxo com a adição do nó *mqtt in*, o qual possibilitou a entrada das mensagens enviadas pelo nó *mqtt out*. A configuração dos "nós" *mqtt out* e *mqtt in* é quase idêntica, pois a maior parte da configuração desses "nós" está relacionada com a definição do servidor e do tópico MQTT, o qual é realizada aos pares, conforme pode ser visto na Figura 53. É importante salientar que a conexão entre a saída dos dados dos sensores (*mqtt out*) e a entrada das mensagens na nuvem (*mqtt in*), foi possível devido a utilização do *broker* Mosquitto, o qual foi utilizado como servidor.

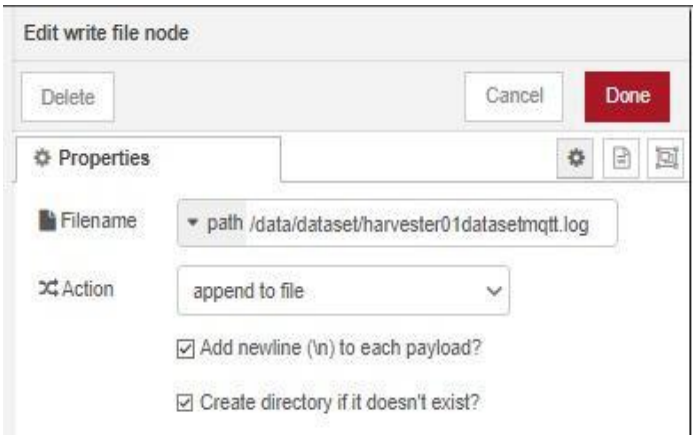
Figura 53 - Semelhanças nas configurações do nó *mqtt out* e *mqtt in*



Fonte: A autora

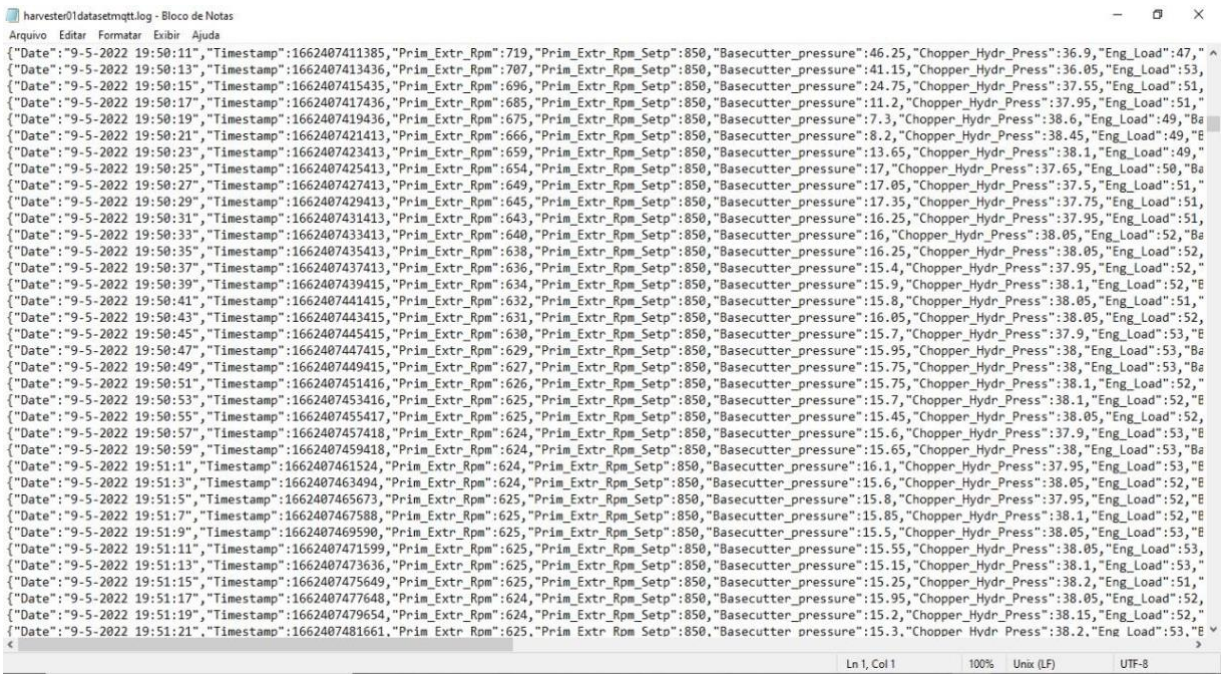
Para criar um arquivo contendo os registros das mensagens acrescentamos o nó *write file*. Nesse nó foi configurado o caminho do arquivo de registros onde as mensagens deveriam ser salvas. Também foi definido que, caso o arquivo ainda não existisse, ele seria criado no diretório especificado neste caminho. Ainda, conforme pode ser observado na Figura 54, configuramos o nó de forma que a cada mensagem recebida uma nova linha contendo os dados da mensagem seria acrescentada no arquivo de registros, mantendo-se os dados já existentes. A Figura 55 exemplifica um arquivo de registros criado através do nó *write file*.

Figura 54 - Configuração do nó *write file*



Fonte: A autora

Figura 55 - Arquivo de registros *harvester01datasetmqtt.log*



Fonte: A autora

O fluxo, o qual teve como objetivo receber os dados provenientes do fluxo de injeção de dados e os armazenar em um arquivo como registros, foi acrescentado em todas as trinta guias criadas no Node-RED. Em cada uma das guias, foram alterados o tópico definido no nó *mqtt in* e o caminho com o nome do arquivo de registros (i.e., */data/dataset/harvester01datasetmqtt.log)* no nó *write file*.

Assim, a primeira etapa da construção do Modelo de Informação utilizou a ferramenta Node-RED para criar o fluxo de entrada de informações. Aliando-se o *broker* Mosquitto foi

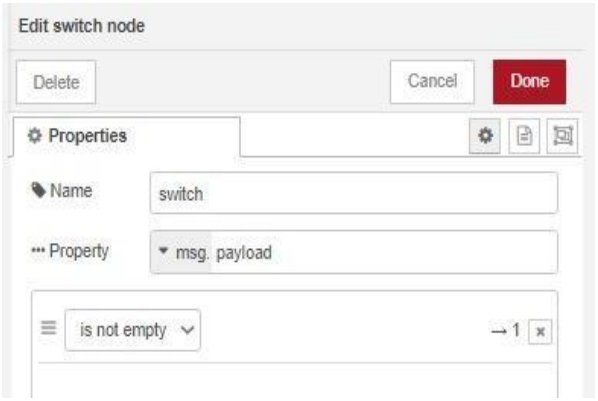
possível criarmos o caminho da informação desde a captura dos dados de sensores até a criação de um arquivo contendo todos os registros desses dados. Dessa maneira, com os dados de todos os arquivos de entrada injetados e salvos nos arquivos de registro, partimos para a segunda etapa de construção do Modelo de Informação.

A segunda etapa de desenvolvimento do Modelo de Informação teve como objetivo armazenar os registros em um banco de dados. Nessa etapa utilizamos a ferramenta Node-RED em conjunto com a base de dados relacional MySQL.

Visando atingir o objetivo dessa segunda etapa, adicionamos um fluxo contendo seis "nós" a cada uma das guias criadas na etapa anterior no Node-RED. A criação desse fluxo foi iniciada adicionando o nó *read file, o qual* foi responsável pela leitura do arquivo de registro. No caso da guia *Harvester02*, o nó foi configurado para ler o arquivo de registros ligado aos dados de sensores de colheitadeira *harvester02* (i.e., *harvester02datasetmqtt.log*). Ainda, definimos que a cada linha do arquivo que fosse lida, seria gerada uma mensagem.

Na sequência, o nó *switch* e *json* foram acrescentados. O nó *switch* permite criar regras, avaliar se as mensagens que passam pelo nó cumprem com as regras estipuladas e ainda, de acordo com as regras, direcionar cada mensagem ao seu nó de destino. No contexto em que o nó estava sendo utilizado, definimos que apenas as mensagens que não estivessem vazias seriam enviadas para o próximo nó, conforme mostra a Figura 56. Assim, utilizando-se o nó switch foi possível filtrar e enviar para o nó *json* apenas aquelas mensagens que continham de fato dados.

Figura 56 - Configuração do nó *switch*

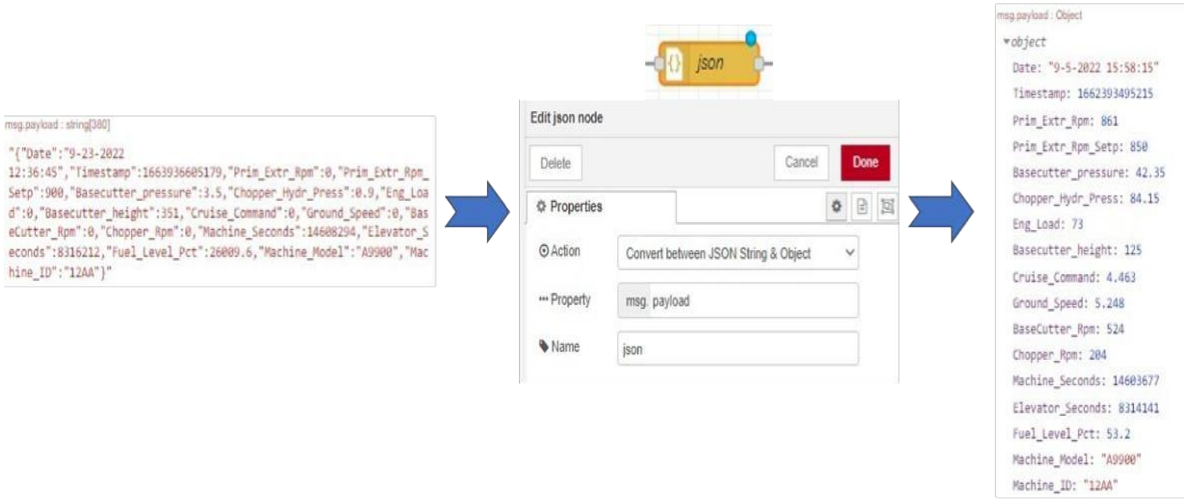


Fonte: A autora

As mensagens geradas pelo nó *read file*, por se tratar de registros de dados, estavam configuradas com *string*, ou seja, eram lidas apenas como um conjunto de caracteres. No

entanto, para que fosse possível acessar as propriedades contidas em cada parte das mensagens e injetar corretamente os dados na base de dados, as mensagens deveriam ser transformadas em objetos JSON. Assim, adicionamos ao fluxo o nó *json*, o qual ao receber as mensagens em formato *string* as transformava em objeto. Essa transformação é exemplificada na Figura 57.

Figura 57 - Exemplo de mensagem antes e após o nó *json*

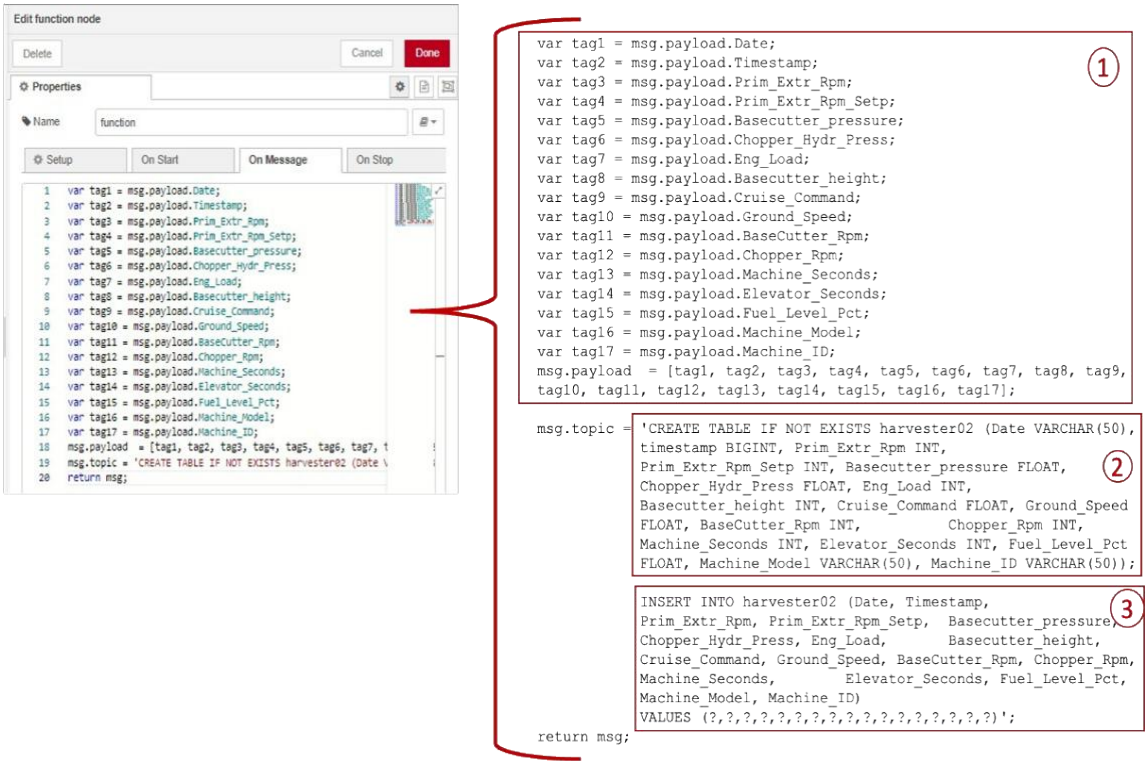


Fonte: A autora

O nó *function* foi o quinto nó adicionado ao fluxo de armazenamento dos dados no banco de dados. Ao configurar o nó, definimos através do código JavaScript a criação de uma tabela na base de dados, os tipos de dados que seriam armazenados em cada campo da tabela foram declarados, e relacionou-se cada parte da mensagem com um campo da tabela. Por meio desse nó, tabelas foram criadas na base de dados e os dados presentes nas mensagens foram registrados.

Na Figura 58 o código que desenvolvemos no nó *function* do fluxo *Harvester02* de armazenamento dos dados no banco de dados é exemplificado. Na primeira parte do código (1), foram declaradas as variáveis que fazem parte da mensagem, totalizando dezessete partes. Na segunda parte do código (2) foi criada a tabela na base de dados, no exemplo criou-se a tabela *harvester02*, e foram especificados os nomes dos campos e os tipos de dados que deveriam constar na tabela. Por fim, os dados foram inseridos na tabela criada e os campos da tabela foram relacionados aos valores das variáveis declaradas (3). Os códigos desse nó para cada um dos fluxos são apresentados no Apêndice E.

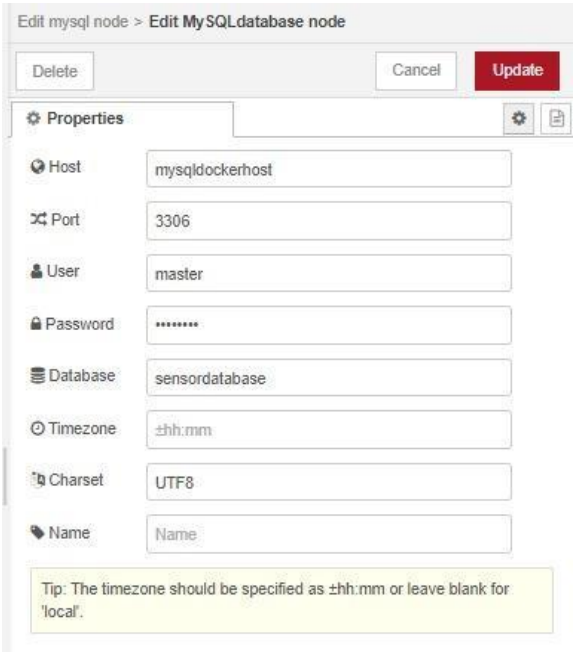
Figura 58 - Nó *function* fluxo de armazenamento de dados (*Harvester02*)



Fonte: A autora

Na sequência, adicionamos ao fluxo o nó *mysql*. Nesse nó foram explicitadas as informações da base de dados onde os dados seriam armazenados como: a localização do servidor da base e porta, o nome da base de dados, nome de usuário e senha de acesso. A Figura 59 apresenta a configuração do nó *mysql* a qual foi utilizada para todos os fluxos de armazenamento de dados do Modelo. Devido ao Modelo de Informação estar sendo desenvolvido utilizando-se o Docker, a base de dados MySQL já estava configurada e pronta para receber os dados, portanto foi necessário apenas utilizar os dados definidos anteriormente no arquivo que que havia sido utilizado para executar e instalar a aplicação. Por fim, como nos outros fluxos criados, adicionamos o nó *injection*, possibilitando que o fluxo de armazenamento dos registros na base de dados fosse acionado.

Figura 59 - Definições do nó *mysql*



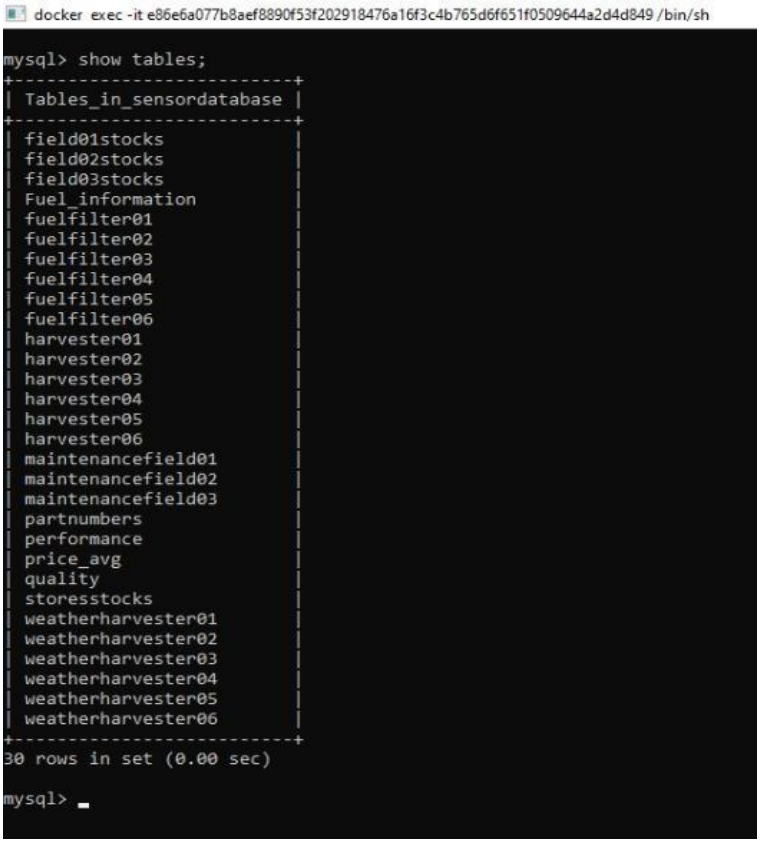
Fonte: A autora

O fluxo criado para armazenar os registros na base de dados foi adicionado a todas as guias do Node-RED. O nome do arquivo de registros que serviu de entrada ao nó *read file*, assim como as variáveis declaradas, os nomes dos campos, tipos de dados e o nome das tabelas definidos no nó *function* foram diferentes para cada uma das guias, respeitando a particularidade de cada um. A Figura 60 lista todas as trinta tabelas criadas na base de dados a partir desse fluxo.

Assim, na segunda etapa de criação do Modelo, a ferramenta Node-RED foi novamente utilizada, mas agora, em conjunto com o MySQL, criando-se uma base contendo as tabelas de dados. Após armazenar todos os registros nas suas respectivas tabelas na base de dados, deu-se início à terceira etapa de desenvolvimento do Modelo de Informação.

Na terceira etapa de desenvolvimento do Modelo de Informação buscamos explorar e analisar os dados armazenados na base de dados com o auxílio da base de dados MySQL e da solução *open source* Metabase. Portanto, nessa etapa o primeiro passo foi conectar a ferramenta Metabase ao *container* MySQL para que os dados contidos na base de dados pudessem ser acessados.

Figura 60 - Lista de tabelas na base de dados



Fonte: A autora

Ao acessar a ferramenta pela primeira vez, via *browser,* foi necessário realizar o cadastro e criar uma conta, logo após, tratamos da conexão com o MySQL. Dados da base de dados como *host* e porta do servidor, nome do banco, usuário e senha foram informados para que a Metabase pudesse se conectar e acessar os dados da base (i.e., *sensordatabase*), a qual continha todas as tabelas de dados. Assim, foi possível realizarmos perguntas para consultar, filtrar e relacionar tabelas entre si de forma simples e visual.

As relações observadas no Modelo Sistêmico entre os agentes que estavam representados e faziam parte do contexto de aplicação foram de suma importância para a análise dos dados. Buscamos analisar os dados presentes em cada uma das tabelas de dados, suas relações e como eles poderiam auxiliar na tomada de decisões relacionadas a manutenção preventiva dos filtros de combustível. O tempo decorrido desde a última troca do filtro de combustível é um exemplo de informação que pode ser obtida através de perguntas realizadas na Metabase utilizando os dados contidos no banco.

Para obter essa informação, utilizamos as tabelas que continham os dados de sensores da colheitadeira e de registros de manutenção. O primeiro passo foi identificar qual era o

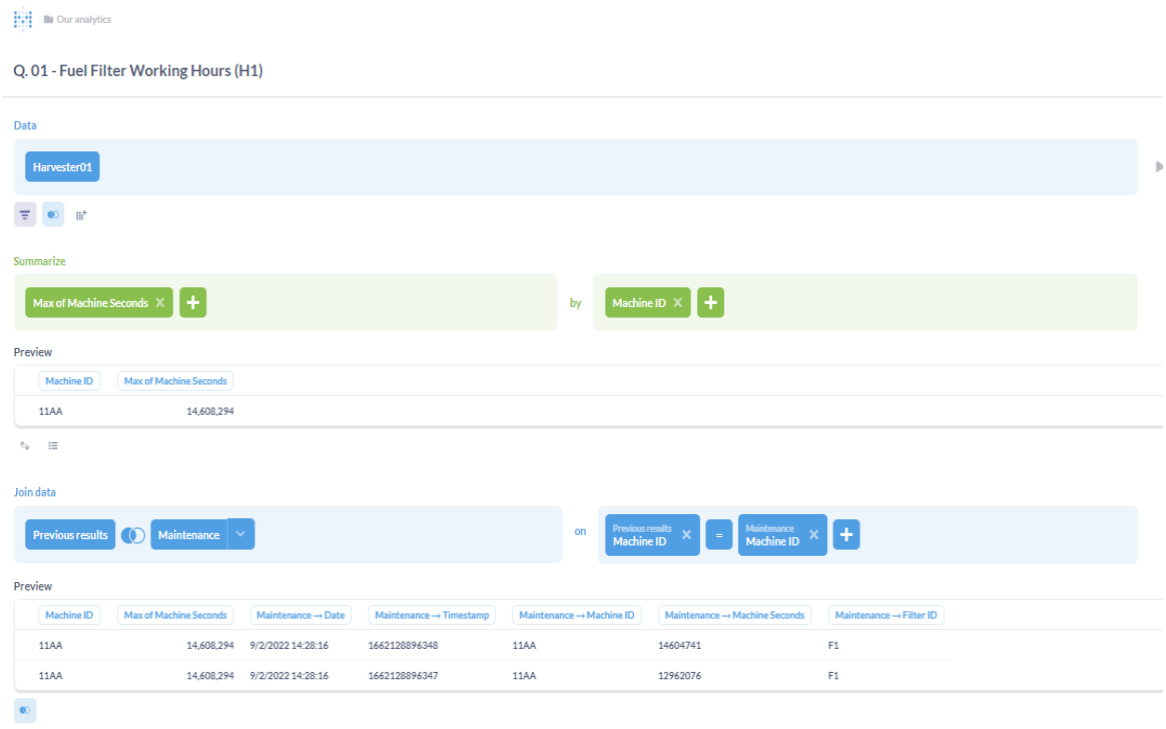
maior número registrado no horímetro da máquina em questão, dado encontrado na coluna *Machine_Seconds*. Na ferramenta Metabase, a tabela contendo os dados de colheitadeira foi selecionada, na sequência utilizou-se a função *summarize*, do inglês resumir. Essa função apresenta diversas métricas relacionadas às colunas da tabela (i.e., realizar um somatório, apresentar a média de valores, apresentar o mínimo ou o máximo valor de uma coluna). No caso dessa pergunta, através dessa função foi obtido o máximo valor da coluna *Machine_Seconds* por máquina (i.e., *Machine_ID*).

Após obter a resposta, utilizamos a função *Join data*, a qual possibilitou fazer a intersecção entre duas tabelas. Essa função pode apresentar a junção de duas tabelas apresentando apenas os dados contidos na intersecção entre elas (i.e., *inner join*), ou todos os dados de uma das tabelas mais os dados contidos na intersecção com outra tabela (i.e., *left outer join* e *right outer join*). Para que seja possível utilizar essa função é necessário que as tabelas contenham uma coluna com o mesmo tipo de dados.

Nesse caso, utilizamos a resposta obtida na pergunta anterior e realizamos um *left outer join* com a tabela *Maintenance*, utilizando a coluna *Machine_ID* como ligação entre as duas tabelas. Assim, obtivemos os dados da tabela com registro de manutenção da colheitadeira em questão juntamente com os dados da máquina obtidos anteriormente através da função sumarize (Figura 61).

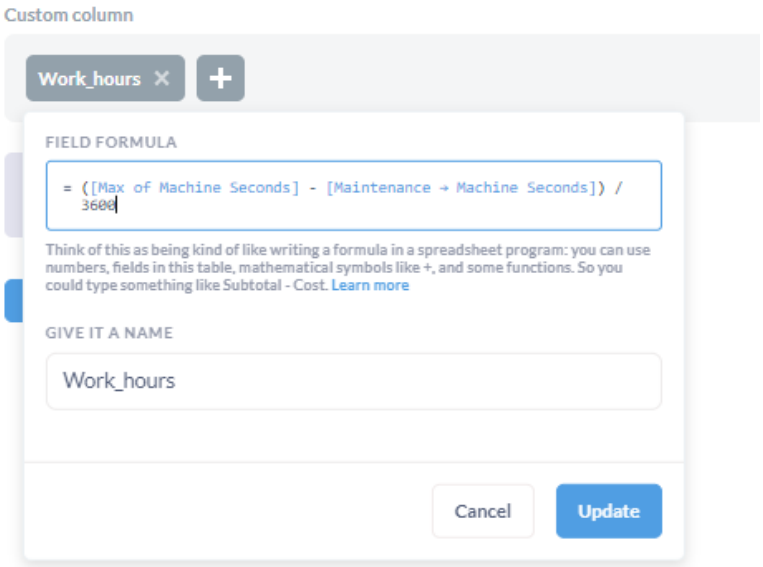
Após obtermos essas informações, acrescentamos à resposta uma coluna contendo as horas decorridas desde as trocas do filtro. Para isso, utilizamos a função *Custom column*, do inglês coluna customizada, a qual permite realizar operações matemáticas entre dados de duas colunas e adiciona uma nova coluna com o resultado da operação. Nesse caso, o máximo valor do horímetro do equipamento, proveniente da tabela *Harvester01*, foi subtraído do valor do horímetro contido na tabela *Maintenance*, retornando os valores na coluna denominada *Work_hours*. Como o valor apresentado pelo horímentro do equipamento era medido em segundos e o tempo de vida dos filtros de combustível em horas, o resultado ainda foi dividido por 3600, conforme mostra a Figura 62.

Figura 61 - Exemplo das funções *Summarize* e *Join Data*



Fonte: A autora

Figura 62 - *Custom column - Work_hours*

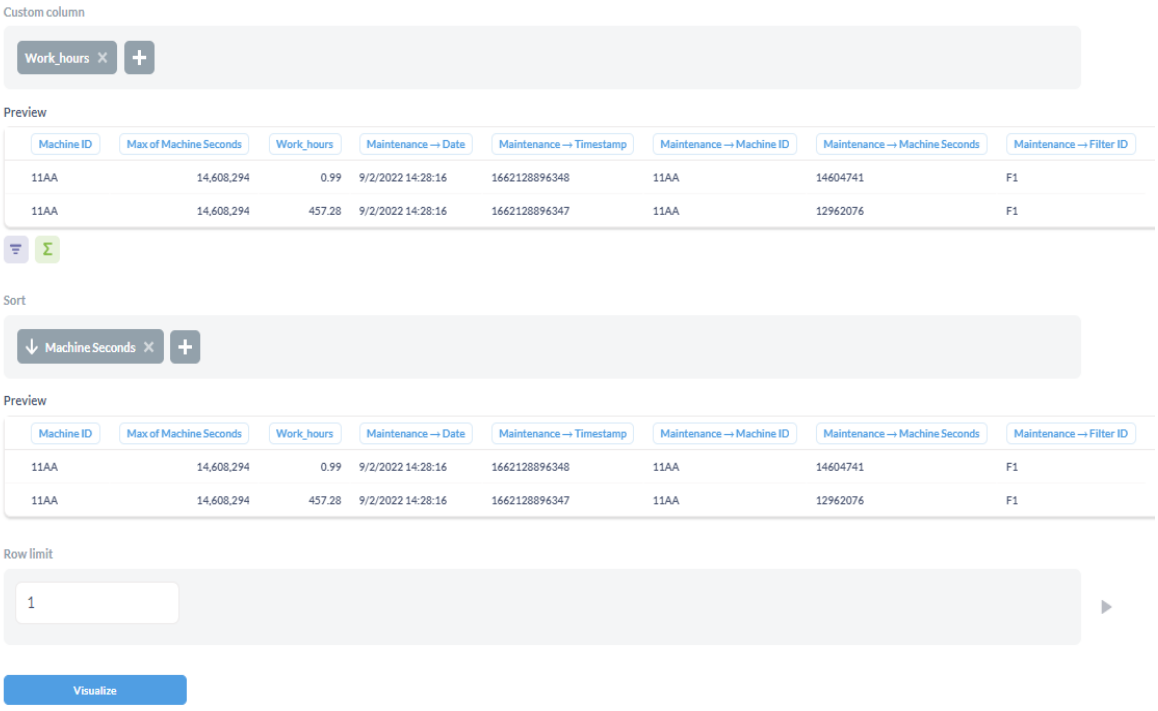


Fonte: A autora

Na mesma pergunta ainda foram utilizadas mais duas funções: *Sort*, do inglês ordenar, e *Row limit*, do inglês limite de linhas. A função Sort foi utilizada para ordernar os resultados a partir da coluna *Work_hours* de maneira crescente. Por sua vez, a função Row limit, limitou

a aparição dos resultados para apenas uma linha, ou seja, apenas o menor valor decorrido desde a troca do filtro de combustível apareceu nos resultados. A Figura 63 apresenta o exemplo dessas funções.

Figura 63 - Exemplo das funções *Custom column*, *Sort* e *Row limit*



Fonte: A autora

Ainda, seria possível controlar a saturação dos filtros através da pressão do combustível medida pelos sensores após a passagem pelo filtro e levar em consideração esse dado para que o filtro fosse utilizado até a sua total saturação, por exemplo. Quanto à visualização dos dados e das respostas das perguntas, há inúmeras possibilidades de visualização dos disponíveis na ferramenta Metabase, conforme será detalhado na seção de demonstração da solução.

Apesar da ferramenta Metabase ser bastante intuitiva e visual, só é possível realizar perguntas se houver um fluxo estruturado para que as informações sejam armazenadas de maneira ordenada na base de dados, por esse motivo as duas etapas anteriores foram de extrema importância para a construção do Modelo. É importante mencionar que, ao longo da terceira etapa de desenvolvimento do Modelo, consultamos diversas vezes as relações entre os agentes de uma fazenda 4.0 representadas pelo Modelo Sistêmico.

Da mesma forma, estudar os dados que foram injetados no modelo é essencial para entender quais informações estão contidas nas tabelas e suas relações, proporcionando aproveitar ao máximo as funções disponíveis na ferramenta, de maneira que os dados possam ser apresentados de maneira a facilitar, de fato, análises e tomadas de decisão.

Ao final dessa terceira etapa, foi possível atingir o objetivo específico de criar um modelo de informação orientado à Agricultura 4.0, capaz de tornar possível a análise de dados e a predição de eventos para antecipar tomadas de decisão, bem como partir para a quarta etapa da abordagem metodológica adotada neste trabalho – Demonstração da Solução. Dessa maneira, a próxima seção apresenta as perguntas realizadas e a construção de um *dashboard* capaz de responder às questões de competência propostas.

4.2 Demonstração da Solução

Seguindo a abordagem metodológica adotada por esta pesquisa, após detalhar o procedimento de construção da solução, iniciou-se sua demonstração.

Diante dos objetivos traçados neste trabalho, primeiramente buscamos entender melhor o contexto envolvendo a troca dos filtros de combustível para que fosse possível elaborar *queries*, capazes de gerar informações que auxiliassem na decisão de trocar ou não o filtro de combustível.

A manutenção desses componentes era realizada de maneira preventiva, seguindo a recomendação do fabricante de trocar o filtro de combustível a cada 500 horas de uso, porém conforme relatado pela empresa parceira, muitas vezes o filtro era trocado sem estar saturado, ou seja, o filtro poderia ter sido utilizado por mais tempo antes de ser substituído. No entanto, caso as trocas não fossem realizadas de acordo com a recomendação do fabricante, haveria a possibilidade de paradas não programadas para a troca do filtro, devido a sua total saturação, incorrendo em manutenções corretivas. Percebemos então que seria útil ter informações relativas à saturação do filtro, juntamente com o tempo decorrido desde a sua última troca.

Consideramos ainda que, mesmo que o filtro ultrapassasse as 500 horas trabalhadas, caso a informação do nível de saturação do filtro estivesse disponível seria viável monitorá-la, possibilitando aguardar para que as trocas do filtro de combustível fossem realizadas, ou não, por exemplo, quando as colheitadeiras parassem para serem abastecidas. Da mesma forma, caso as horas trabalhadas estivessem muito próximas da recomendada para a troca, seria necessário abastecer a colheitadeira, caso a saturação do filtro estivesse alta poderia ser realizada a troca naquele momento. Assim, o tempo de máquina parada relativo ao

deslocamento do local de colheita até o ponto de manutenção e abastecimento poderia ser reduzido.

O tempo de máquina parada representa um custo para as fazendas, assim como o custo da peça, os custos de mão de obra de manutenção. Ainda, a saturação dos filtros de combustível é influenciada pela qualidade do combustível utilizado. Quanto maior o percentual de água e partículas presentes no óleo diesel, mais rapidamente ocorre a saturação do filtro. Outro possível ponto a ser observado quanto ao intervalo entre trocas dos filtros é a influência dos aspectos meteorológicos como a temperatura do local onde a colheitadeira está operando. Deste modo, após compreender o contexto de manutenção dos filtros de combustível, iniciamos a elaboração das *queries*, levando em consideração o cenário de aplicação apresentado na seção 3.3.4.

A partir do cenário apresentado, realizamos diversas perguntas com o propósito de identificar se o modelo desenvolvido seria capaz de auxiliar de maneira satisfatória a decisão de troca do filtro de combustível dentro do contexto desta pesquisa. Essas *queries* foram baseadas no entendimento apresentado no início dessa seção sobre o cenário de aplicação.

Iniciamos a demonstração com a formação de uma *query* para se obter o tempo decorrido desde a última troca do filtro de combustível, uma vez que a manutenção era baseada nas horas de vida útil do componente. Há inúmeras possibilidades de visualização dos dados e respostas disponíveis na ferramenta Metabase. No caso da resposta dessa pergunta, o formato *Gauge*, do inglês medidor, foi escolhido. Conforme mostra a Figura 64, a colheitadeira *Harvester 01* trabalhou menos de uma hora desde a última troca do filtro de combustível.

Conforme pode ser observado na Figura 64, definimos cores e ranges para facilitar a interpretação da resposta. Desse modo, o primeiro range conteve o intervalo de tempo entre a troca e as primeiras 200 horas de uso do filtro, sendo representado pela cor verde. O segundo range, representado pela cor amarela indicou a utilização entre 200 e 400 horas. Por fim, o intervalo das 400 a 500 horas de utilização do filtro foi representado pela cor vermelha, indicando atenção para uma possível necessidade de troca do componente. Essa *query* foi denominada *Q.01 - Fuel Filter Working Hours* e os detalhes da sua construção foram apresentados na seção 4.1, a fim de apresentar algumas das funções da ferramenta Metabase.

Figura 64 - *Q.01 - Fuel Filter Working Hours*

Fonte: A autora

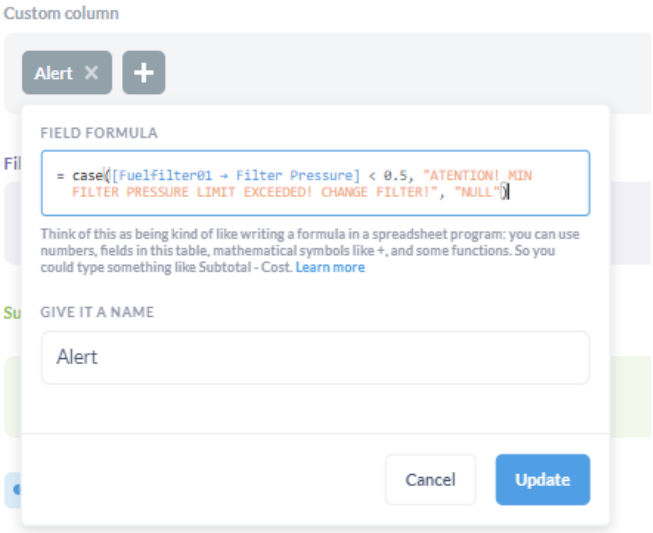
A segunda *query* - *Q.02 - Alert and Change Status* - criou um alerta para caso a pressão do filtro atingisse um valor menor do que o mínimo aceitável. Para a formação dessa *query* foi necessário realizarmos oito etapas e utilizarmos seis funções disponibilizadas pela ferramenta Metabase. Iniciamos a *query* utilizando a função *join*, à qual juntou as informações presentes nas tabelas *Harvester*, *Fuel_Filter* e *Maintenace* da base de dados.

Para que fosse possível gerar o alerta, acrescentamos uma nova coluna (*Alert*) utilizando a função *Custom column* (Figura 65). Nessa função adicionamos uma expressão a qual determinava que se a pressão medida pelo sensor fosse menor que 0.5 bar, o filtro de combustível estaria saturado e o valor do campo da coluna era preenchido com o aviso gerado. Na sequência, as funções *filter* e *sumarize* foram utilizadas com o objetivo de evitar o processamento desnecessário de dados. A função *filter* possibilitou diminuir a quantidade de dados processados, uma vez que ao filtrar os dados apenas as linhas que continham valores de pressão menor que 0.5 bar foram selecionadas. Já a função *summarize* agregou os dados utilizando a coluna *Machine_Seconds* da tabela *Maintenance* e os agrupou considerando quatro dimensões, ou seja, apenas quatro colunas passaram a ser exibidas.

A seguir, utilizamos novamente a função *Custom column*. O status indicando se a troca do filtro já havia sido realizada, também foi adicionado à resposta através da criação da coluna *Change_status*. Comparamos o valor do horímetro da colheitadeira (coluna *Machine_Seconds* da tabela *Harvester*) no momento do alerta com o valor do horímetro da máquina no momento da última troca (coluna *Machine_Seconds* da tabela *Maintenance*). Como apresentado pela Figura 66, a troca era sinalizada caso o valor *Machine_Seconds*

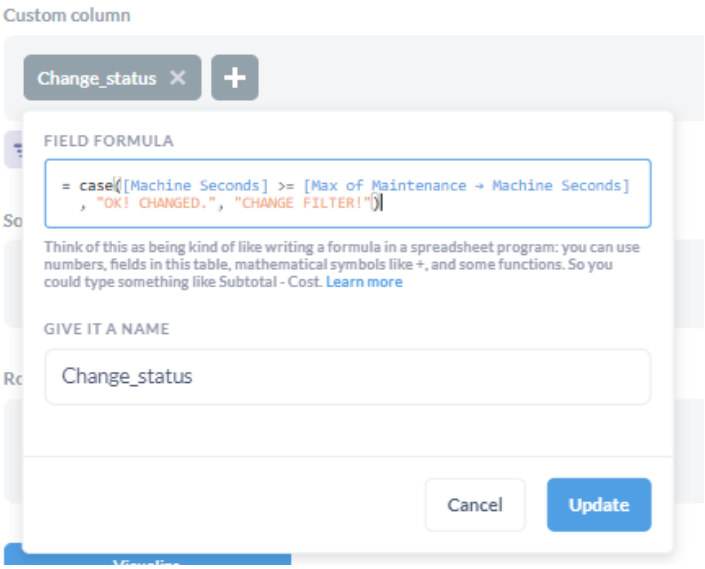
presente na tabela com os registros de manutenção fosse maior ou igual ao valor do *Machine_Seconds* no momento do alerta. Assim, enquanto não ocorresse a troca apareceria a mensagem "*CHANGE FILTER!*", caso contrário, quando ocorresse a troca, apareceria a mensagem "*OK! CHANGED*".

Figura 65 - *Custom Column - Alert*



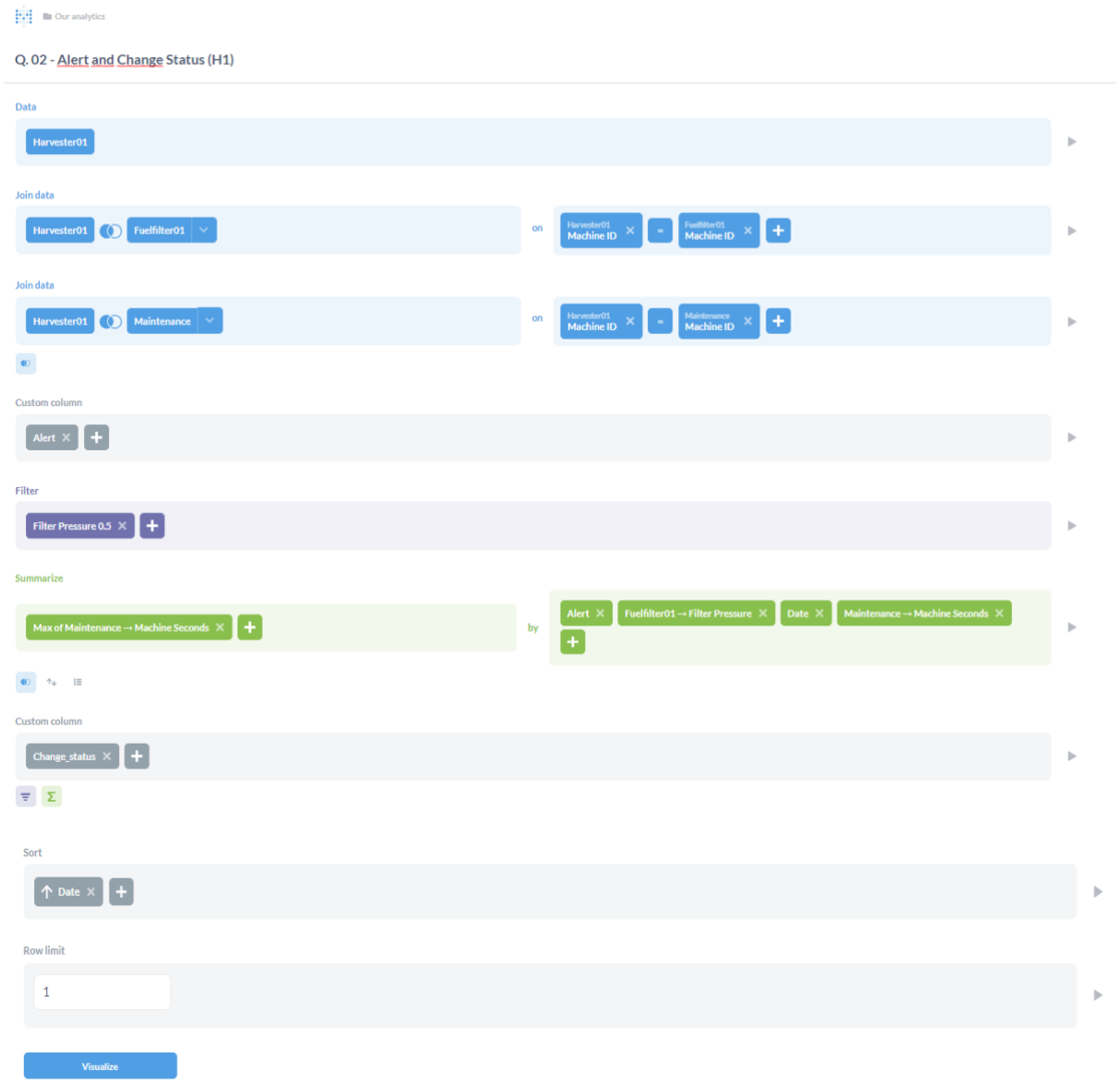
Fonte: A autora

Figura 66 - *Custom Column - Change_status*



Fonte: A autora

Figura 67 - Estrutura da *query Q.02 - Alert and Change Status*



Fonte: A autora

Ainda, ordenamos os resultados de maneira que as datas mais recentes aparecessem primeiro utilizando-se a função *Sort*. Isso possibilitou que o alerta mais recente aparecesse primeiro nos resultados da *query*. Por fim, utilizamos a função *Row limit* a qual limitou a visualização das respostas para apenas uma linha. A Figura 67 apresenta a estrutura da *query*, contendo todas as funções utilizadas para sua criação.

Isto posto, através dessa segunda *query* foi possível receber um alerta caso a pressão de combustível atingisse valores abaixo do limite aceitável, indicando a necessidade de troca devido a saturação do filtro. Também, a *query* foi capaz de indicar se após o surgimento do alerta houve a troca do filtro, conforme ilustra a Figura 68.

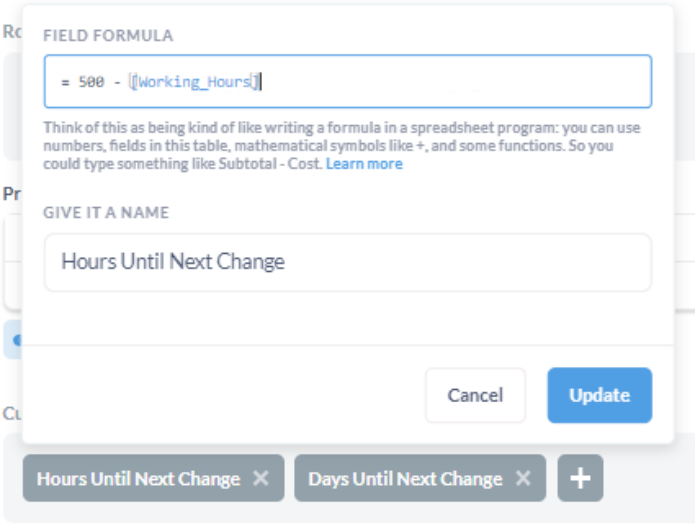
Figura 68 - *Q.02 - Alert and Change Status*

Fonte: A autora

Além dessas informações, consideramos útil indicar o tempo estimado até a próxima troca do filtro, assim elaboramos a *query Q.03 – Time remaining until next filter change*. Para obtermos essa informação, a resposta da primeira *query* (*Fuel Filter Working Hours*) foi utilizada como dado de entrada da pergunta, e foram adicionadas duas colunas através da função *Custom column*. A primeira coluna, denominada *Hours Until Next Change*, continha uma expressão para calcular quantas horas faltavam até a próxima troca do filtro. Essa expressão, a qual pode ser observada na Figura 69, subtraída das 500 horas de vida útil estipuladas pelo fabricante do filtro as horas já trabalhadas (*Working_Hours*).

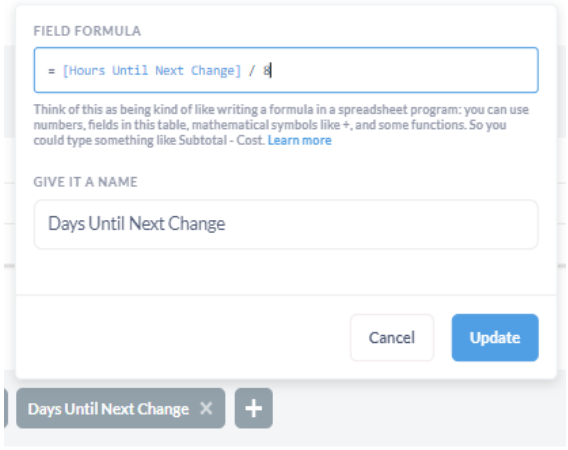
Considerando que a colheitadeira trabalhasse em média oito horas por dia, dividimos a coluna *Hours Until Next Change* por oito para obter a quantidade de dias estimados até a próxima troca, criando a segunda coluna denominada *Days Until Next Change* (Figura 70). Ainda, utilizamos a formatação condicional para realçar as informações através de uma gama de cores. A resposta da terceira *query* pode ser visualizada na Figura 71.

Figura 69 - Custom Column - Hours Until Next Change



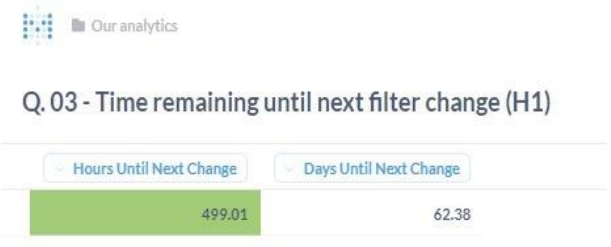
Fonte: A autora

Figura 70 - Custom Column - Days Until Next Change



Fonte: A autora

Figura 71 - Q.03 – Time remaining until next filter change



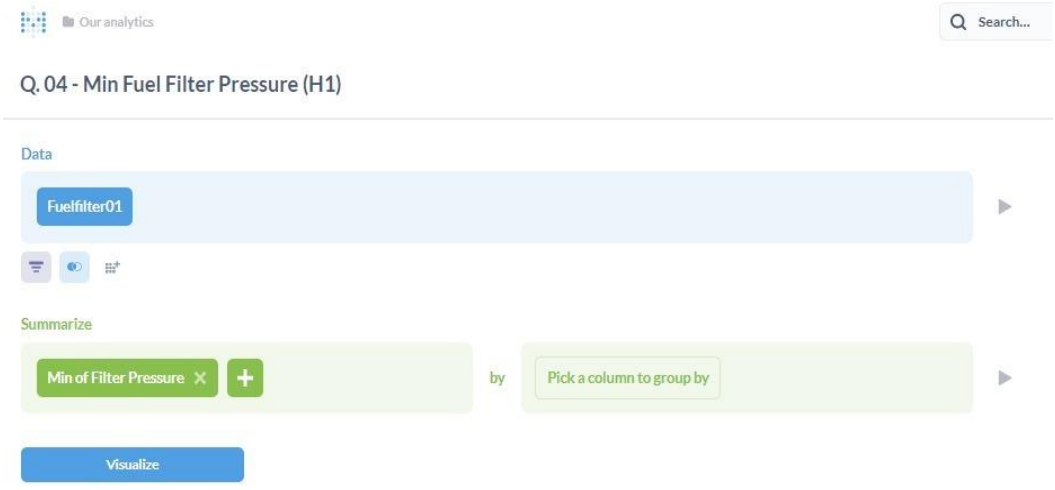
Fonte: A autora

A quarta pergunta realizada - *Q. 04 – Min Fuel Filter Pressure* - apresentou o menor valor medido pelo sensor de pressão de combustível (Figura 72) utilizando os dados da Fuel_Filter como dados de entrada e a função *Summarize*, conforme mostra a Figura 73. Da mesma forma, utilizando o cálculo da média pela função *Summarize*, a quinta *query* - *Q. 05 – Fuel Filter Pressure Average*, informou o valor médio medido pelo mesmo sensor, conforme mostra a Figura 74. Apesar de serem relativamente simples, a elaboração dessas *queries* justifica-se pelo fato de que, em conjunto com as outras *queries*, elas apoiam a tomada de decisões e auxiliam a elucidar a frequência com que as manutenções são realizadas.

Figura 72 - *Q. 04 – Min Fuel Filter Pressure*

Fonte: A autora

Figura 73 - Estrutura da *query Q. 04 – Min Fuel Filter Pressure*



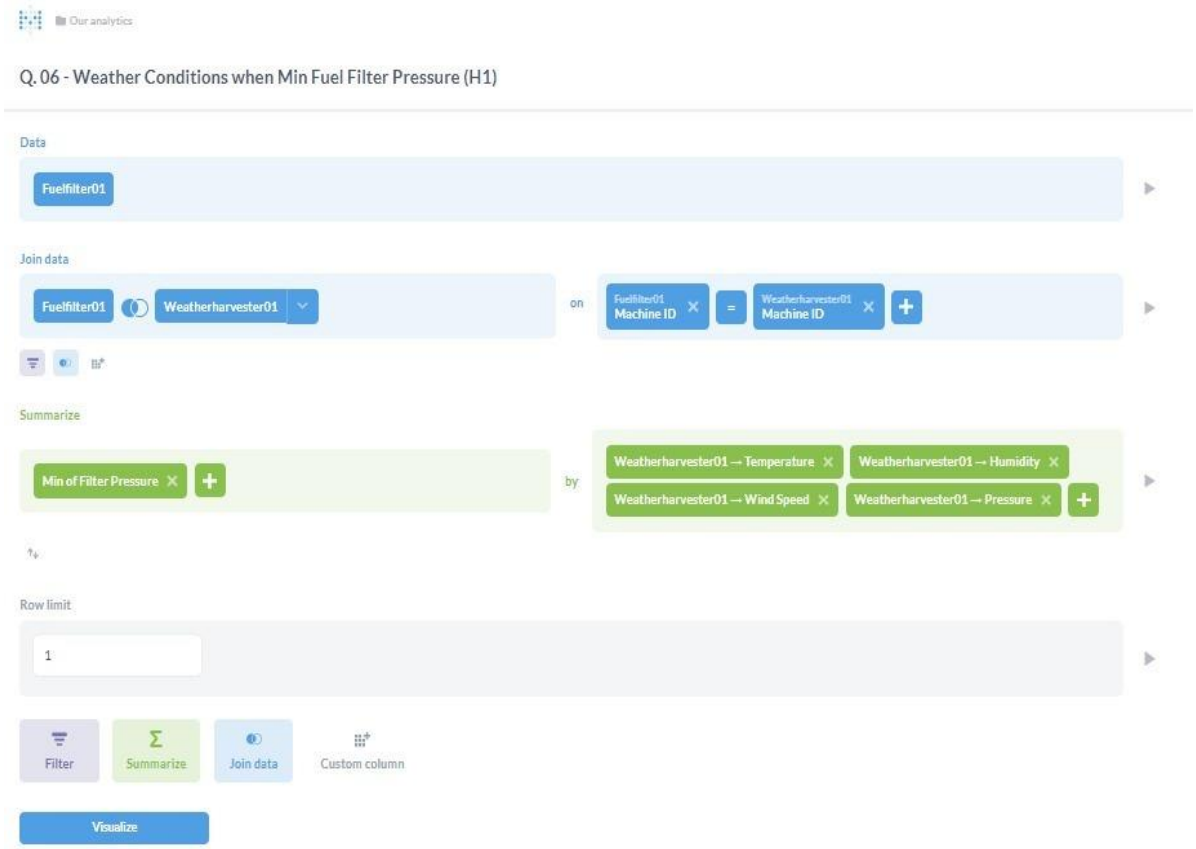
Fonte: A autora

Figura 74 - *Q. 05 – Fuel Filter Pressure Average*

Fonte: A autora

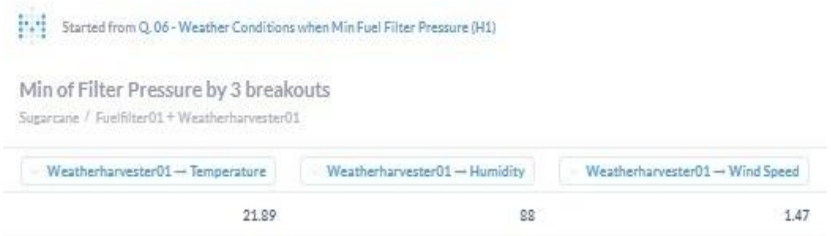
Pensando sobre a influência dos aspectos meteorológicos na vida útil do filtro de combustível e na sua influência sobre o combustível utilizado nas colheitadeiras, foram apontadas as condições do clima quando a menor pressão de combustível foi identificada. Essa *query* foi denominada *Q. 06 – Weather Conditions when Min Fuel Filter* e foi a sexta pergunta realizada para demonstrar a solução proposta. De acordo com a Figura 75, iniciamos a execução dessa *query* considerando os dados da tabela *FuelFilter*, seguido pela sua junção desses dados com os da tabela *WeatherHarvester* e a agregação dos dados utilizando o menor valor da coluna *Filter Pressure* e agrupando-os pelas colunas *Temperature*, *Humidity*, *Wind Speed* e *Pressure*. Por fim limitamos a resposta a uma linha de dados. A resposta gerada pela sexta *query* pode ser visualizada na Figura 76.

Figura 75 - Estrutura da query Q. 06 – Weather Conditions when Min Fuel Filter



Fonte: A autora

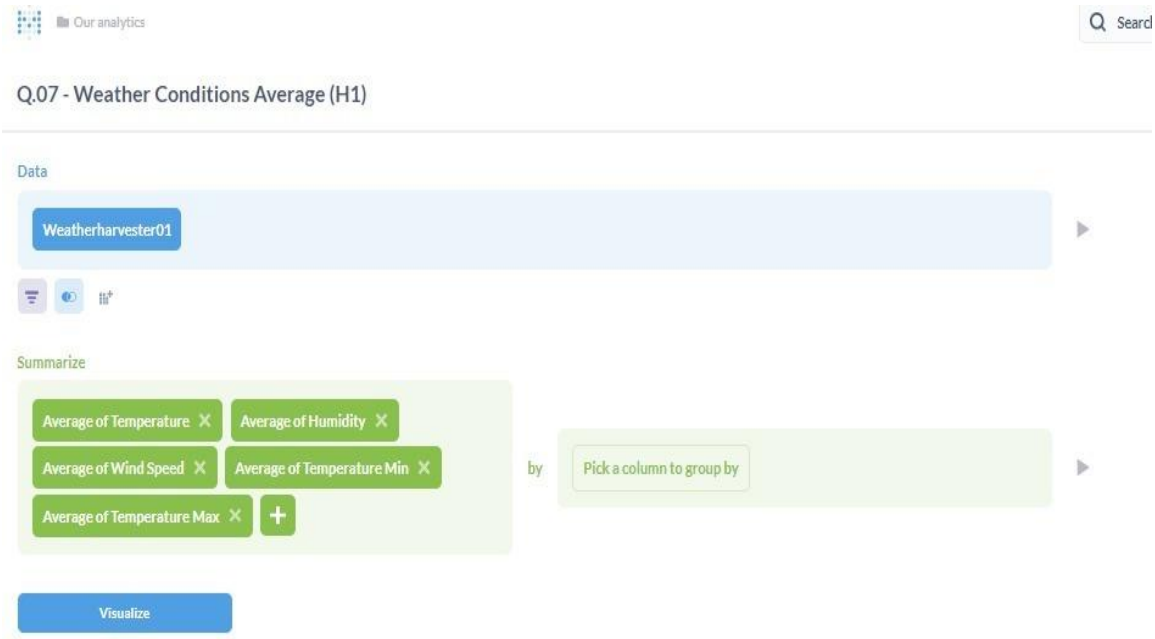
Figura 76 - Q. 06 – Weather Conditions when Min Fuel Filter



Fonte: A autora

Pela mesma razão a qual realizamos a pergunta anterior, geramos a sétima *query* - *Q.07 – Weather Conditions Average*. Realizamos essa *query* utilizando os dados da tabela *WeatherHarvester* e a função *Summarize* a qual agregou os dados realizando a média das colunas *Temperature*, *Humidity*, *Wind Speed*, *Temperature Min* e *Temperature Max*. As Figuras Figura **77** e Figura **78** apresentam a estrutura da *query* no Metabase e a sua resposta, respectivamente.

Figura 77 - Estrutura da *query - Q.07 – Weather Conditions Average*



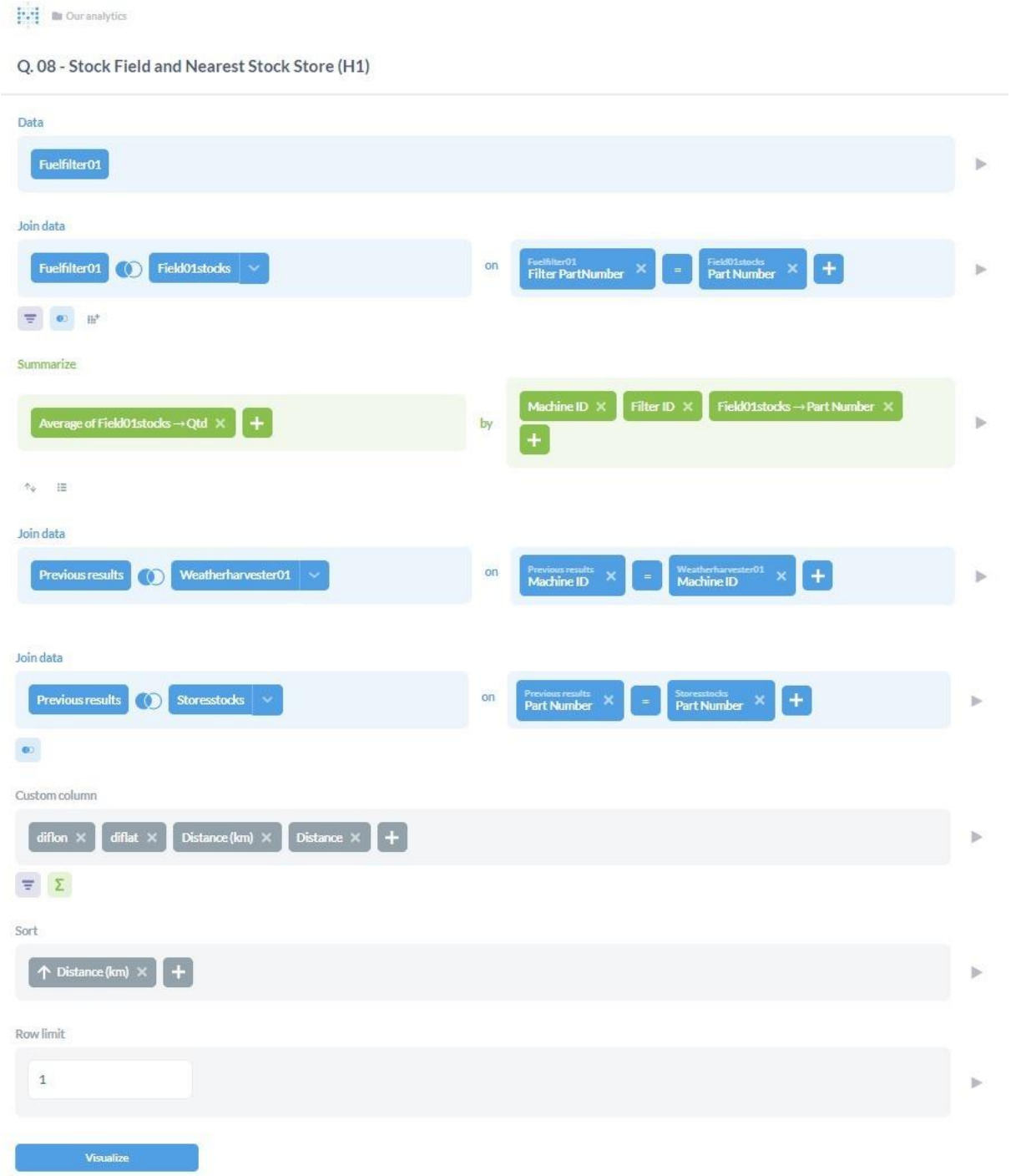
Fonte: A autora

Figura 78 - *Q.07 – Weather Conditions Average*

Fonte: A autora

Pretendendo gerar mais informações que apoiassem as tomadas de decisão acerca do cenário de aplicação, buscamos realizar perguntas que resultassem em informações que ajudassem a garantir a disponibilidade dos filtros em estoque no momento da parada para troca. As tabelas da base de dados que continham os dados de estoques de filtros tanto nas fazendas, quanto nas lojas concessionárias, foram utilizadas para esse fim. Ainda, consideramos que caso não houvesse nenhum filtro em estoque, saber qual a loja mais próxima da fazenda em que a peça poderia ser adquirida, auxiliaria o planejamento de compras e na garantia da disponibilidade da peça no momento da troca. Diante disso, iniciamos a elaboração da oitava *query – Q. 08 – Stock Field and nearest Stock Store*, a qual contou com sete etapas, conforme apresentado na Figura 79.

Figura 79 - Estrutura da *query Q. 08 – Stock Field and nearest Stock Store*



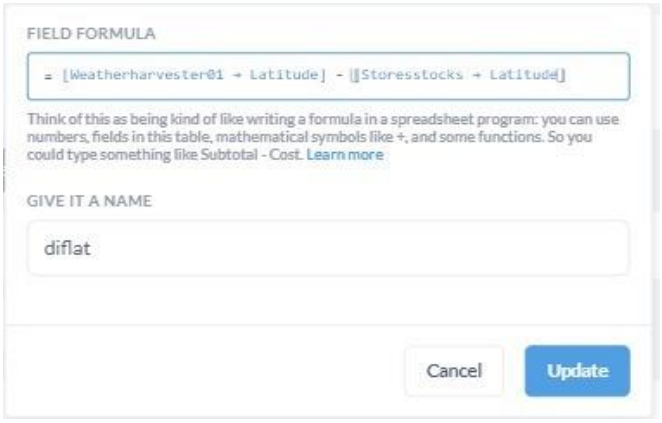
Fonte: A autora

A primeira etapa da elaboração dessa *query* uniu as tabelas *Fuel_Filter* e *Field_Stock*, através da função *Join*, utilizando as colunas *PartNumber* das tabelas como elo. Assim, obteve-se a quantidade em estoque do filtro em questão. Em seguida, utilizamos a função

Summarize para agregar a quantidade em estoque dos filtros, agrupando-a através das colunas *Machine_ID*, *Filter_ID* e *Filter_Part_Number*. Após, realizamos mais duas junções de tabelas, dessa vez unindo os dados obtidos com as tabelas *WeatherHarvester* e *StoreStocks*, obtendo-se assim, as quantidades em estoque de determinado filtro, o nome da cidade onde estava localizada a fazenda, assim como a quantidade em estoque em cada uma das lojas concessionárias.

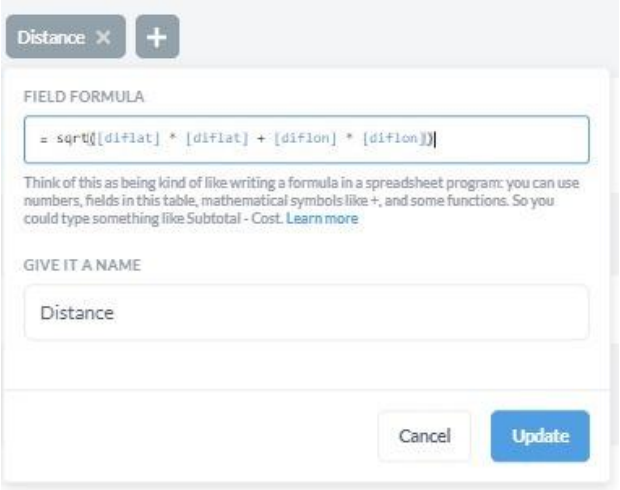
Para identificar a concessionária mais próxima a fazenda que possuía o filtro em estoque, criamos quatro colunas customizadas. Os campos da primeira e da segunda coluna continham o resultado do cálculo da diferença entre a latitude, e longitude da cidade onde a fazenda estava localizada e das lojas, sendo denominadas de *diflat* e *diflon*. A Figura 80 exemplifica a criação de uma dessas colunas. Os campos da terceira coluna (*Distance*) continham o resultado da fórmula do teorema de Pitágoras para o cálculo da distância entre os dois pontos contidos nas colunas *diflat* e *diflon*, conforme mostra a Figura 81. Por fim, a quarta coluna apresenta o resultado da multiplicação dos dados da coluna *Distance* por uma constante, para que a distância entre as cidades fosse apresentada em quilômetros (*Distance(km)*), conforme é apresentado na Figura 82.

Figura 80 - *Custom Column - diflat*



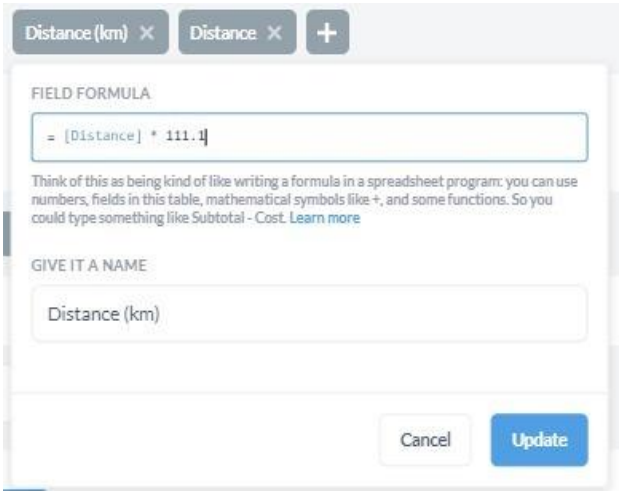
Fonte: A autora

Figura 81 - *Custom Column - Distance*



Fonte: A autora

Figura 82 - *Custom Column – Distance (km)*



Fonte: A autora

Para que fosse mostrada a loja mais próxima da fazenda que continha o filtro em estoque, adicionamos mais duas etapas a *query*. Utilizamos a função *Sort* para ordenar as respostas em ordem crescente, e a função *Row limit*, a qual limitou o resultado a apenas uma linha. Ao resultado obtido no final da *query* (Figura 83) adicionamos ainda a formatação condicional a qual coloria o campo que continha a quantidade de filtros em estoque, a fim de realçar a informação contida no campo.

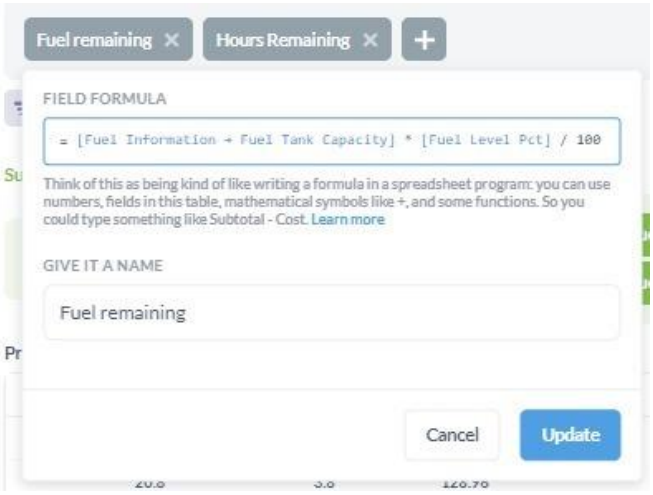
Figura 83 - Q. 08 – Stock Field and nearest Stock Store

Fonte: A autora

Além disso, para que fosse ponderada na tomada de decisão a possibilidade da troca do filtro no momento da parada para abastecimento das colheitadeiras, optamos por realizar outra pergunta a qual informa se a autonomia estimada da colheitadeira considerando a quantidade de combustível ainda disponível no equipamento. Deste modo, elaboramos a nona *query* - *Q.09 – Fuel Level Information – Autonomy*.

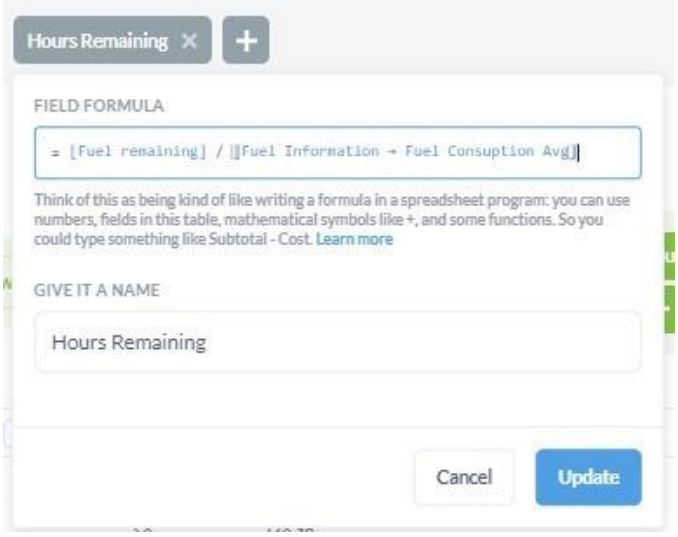
Na nona *query*, utilizamos a função *Summarize,* a partir dos dados da tabela *Harvester*, para que os maiores valores de *Timestamp* fossem exibidos e agrupados pelas colunas que continham o nível de combustível, o modelo e o ID da colheitadeira. Após, juntamos o resultado obtido através dessa função com os dados da tabela *Fuel Information*. Logo após, criamos duas colunas, utilizando a função *Custom column*. A primeira, denominada *Fuel remaining*, realizou o cálculo para obter a quantidade de litros de combustível que restavam na colheitadeira, o qual é possível observar na Figura 84. Já a segunda, denominada *Hours Remaining*, calculou as horas restantes de autonomia da colheitadeira em função do combustível restante e da média de consumo de combustível do equipamento, o cálculo é apresentado na Figura 85.

Figura 84 - *Custom Column - Fuel Remaining*



Fonte: A autora

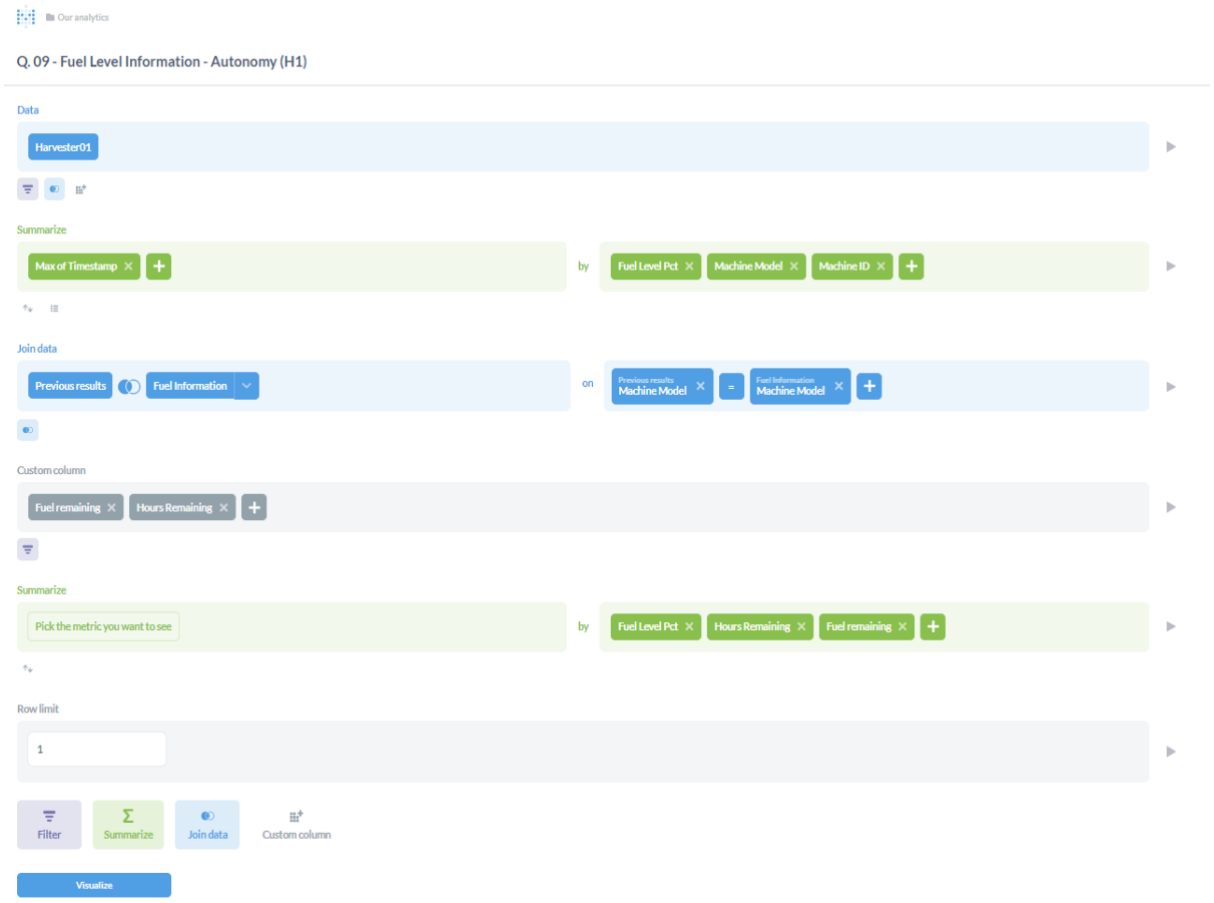
Figura 85 - *Custom Column - Hours Remaining*



Fonte: A autora

Por fim agrupamos os dados, utilizando a função *Summarize*, para que apenas os dados presentes nas colunas *Fuel Level Pct*, *Hours Remaining* e *Fuel Remaining* fossem mostrados, bem como limitamos a aparição dos resultados da *query* a uma linha. Nas Figuras Figura 86 e Figura 87 é possível visualizar todas as funções utilizadas na elaboração dessa *query* e a sua resposta, respectivamente.

Figura 86 - Estrutura da *query Q.09 – Fuel Level Information – Autonomy*



Fonte: A autora

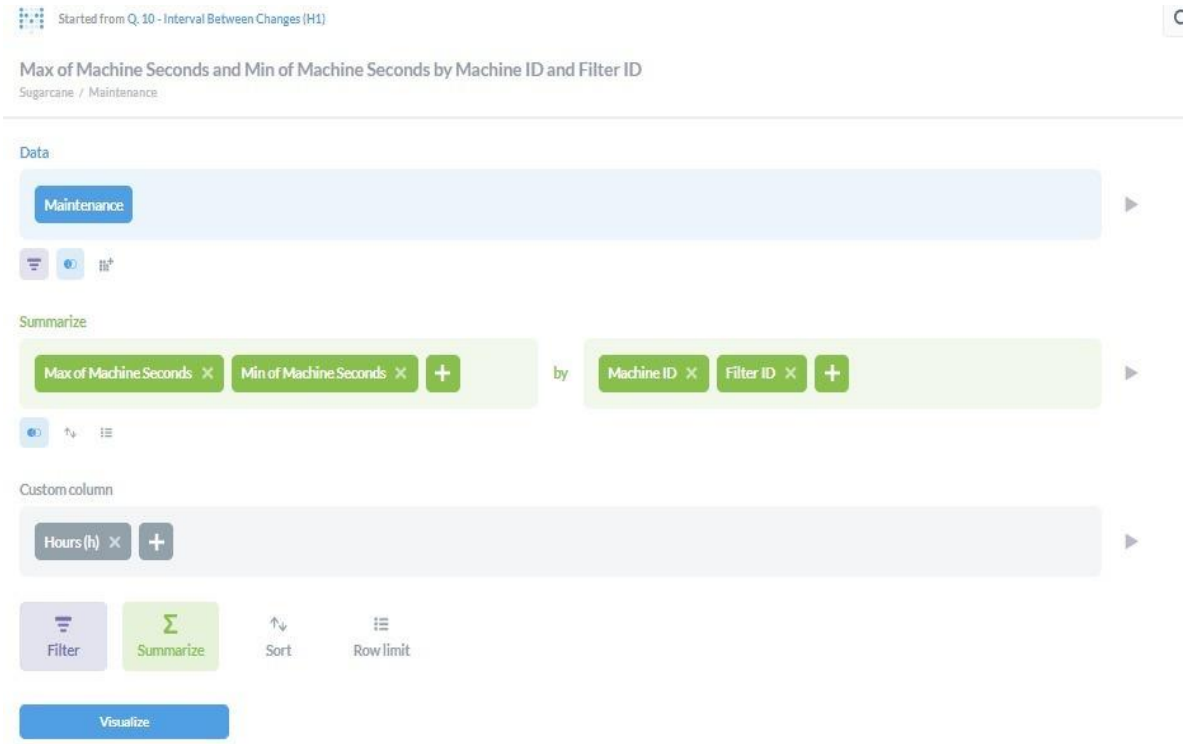
Figura 87 - *Fuel Level Information – Autonomy*

Fonte: A autora

Outra informação que consideramos útil para apoiar a decisão do momento de troca do filtro de combustível foi o intervalo entre as últimas trocas ocorridas. Para obter essa informação criamos a *query Q. 10 – Interval Between Changes* (Figura 88). Essa *query* utilizou como entrada os dados da tabela *Maintenance*, a seguir utilizando a função

Summarize, encontrou-se o maior e o menor valor da coluna *Machine Seconds*, a qual continha os dados do horímetro da colheitadeira.

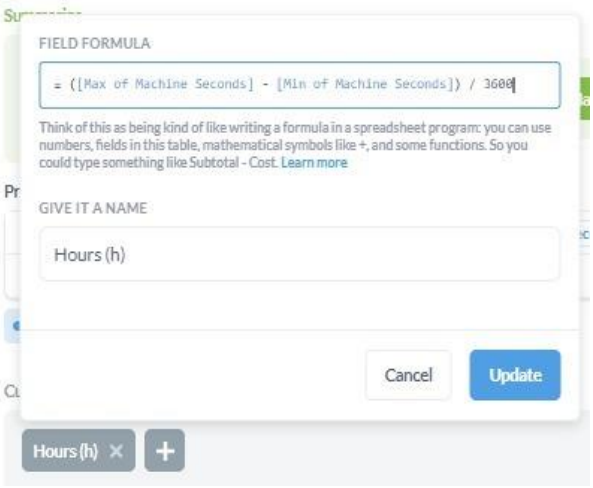
Figura 88 - Estrutura da *query Q. 10 – Interval Between Changes*



Fonte: A autora

A seguir, utilizamos a função *Custom column* para criar a coluna *Hours(h)*. Então os campos dessa coluna foram preenchidos com o resultado da diferença do máximo e mínimo valor de horímetro da máquina. Como os dados de horímetro são medidos em segundos, dividimos o valor por 3600 para apresentar o resultado medido em horas, conforme mostra a Figura 89. Ao observar o resultado desta *query* (Figura 90) é possível perceber a formatação condicional adicionada, na qual, caso o intervalo entre as últimas duas trocas for menor que a vida útil estipulada pelo fabricante do filtro, os dados aparecem em vermelho, caso contrário em verde.

Figura 89 - *Custom Column – Hours (h)*



Fonte: A autora

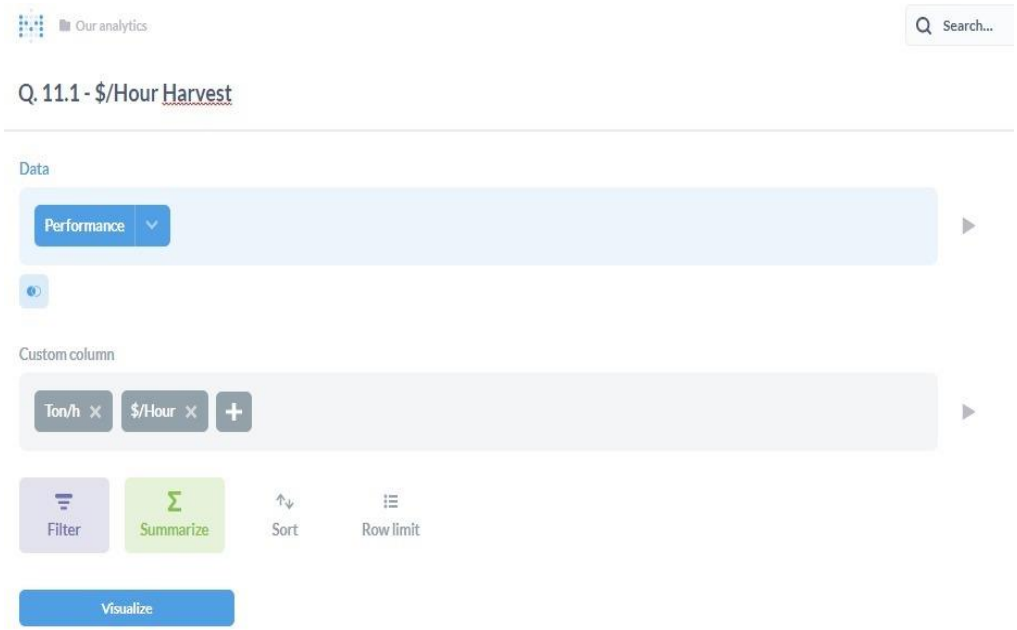
Figura 90 - *Q. 10 – Interval Between Changes*

Fonte: A autora

Ainda, através das tabelas criadas a partir do Modelo de Informação, buscamos calcular e comparar os custos envolvidos na manutenção dos filtros de combustível. No entanto, para criar essa *query* foi necessário elaborarmos primeiro uma outra pergunta (*Q. 11.1 - $/Hour Harvest*), que fosse capaz de informar a quantidade média colhida por hora (*Ton/h*) e o valor médio de uma hora de trabalho das colheitadeiras (*$/Hour*).

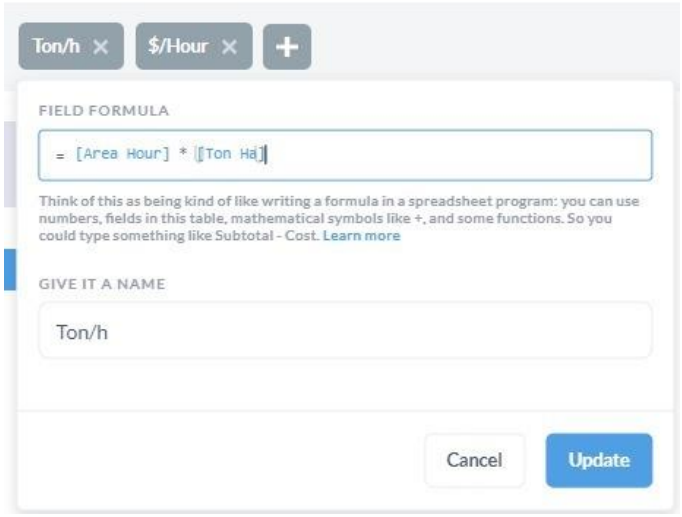
Para a *Q. 11.1 - $/Hour Harvest* utilizamos os dados da tabela *Performance* como dados de entrada e criamos duas colunas utilizando a função *Custom column* (Figura 91). A primeira coluna calculou a quantidade de toneladas colhida (hectares) por hora e a segunda o valor médio ganho em uma hora de trabalho da colheitadeira. Os dados dos campos dessas colunas foram preenchidos utilizando as expressões apresentadas nas Figuras Figura **92** e Figura 93, respectivamente. A resposta dessa *query* é ilustrada pela Figura 94.

Figura 91 - Estrutura da *query Q. 11.1 - $/Hour Harvest*



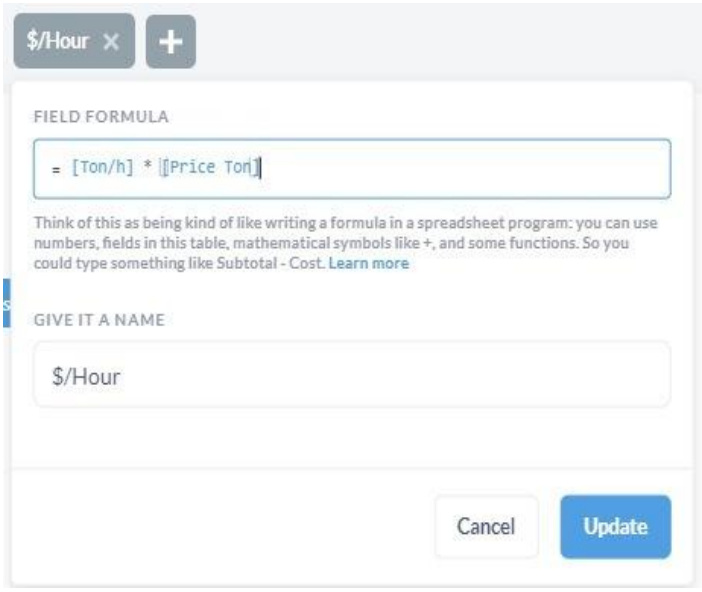
Fonte: A autora

Figura 92 - *Custom Column - Ton/h*



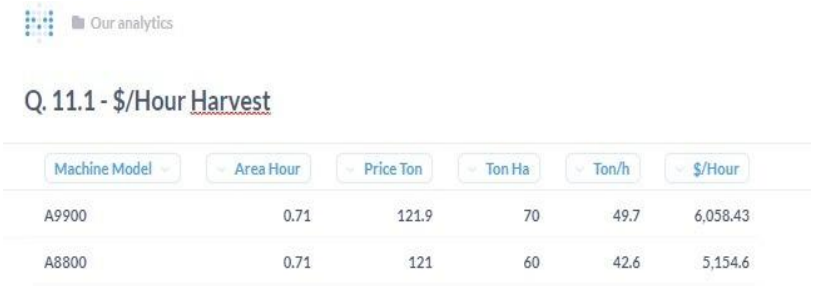
Fonte: A autora

Figura 93 - *Custom Column - $/Hour*



Fonte: A autora

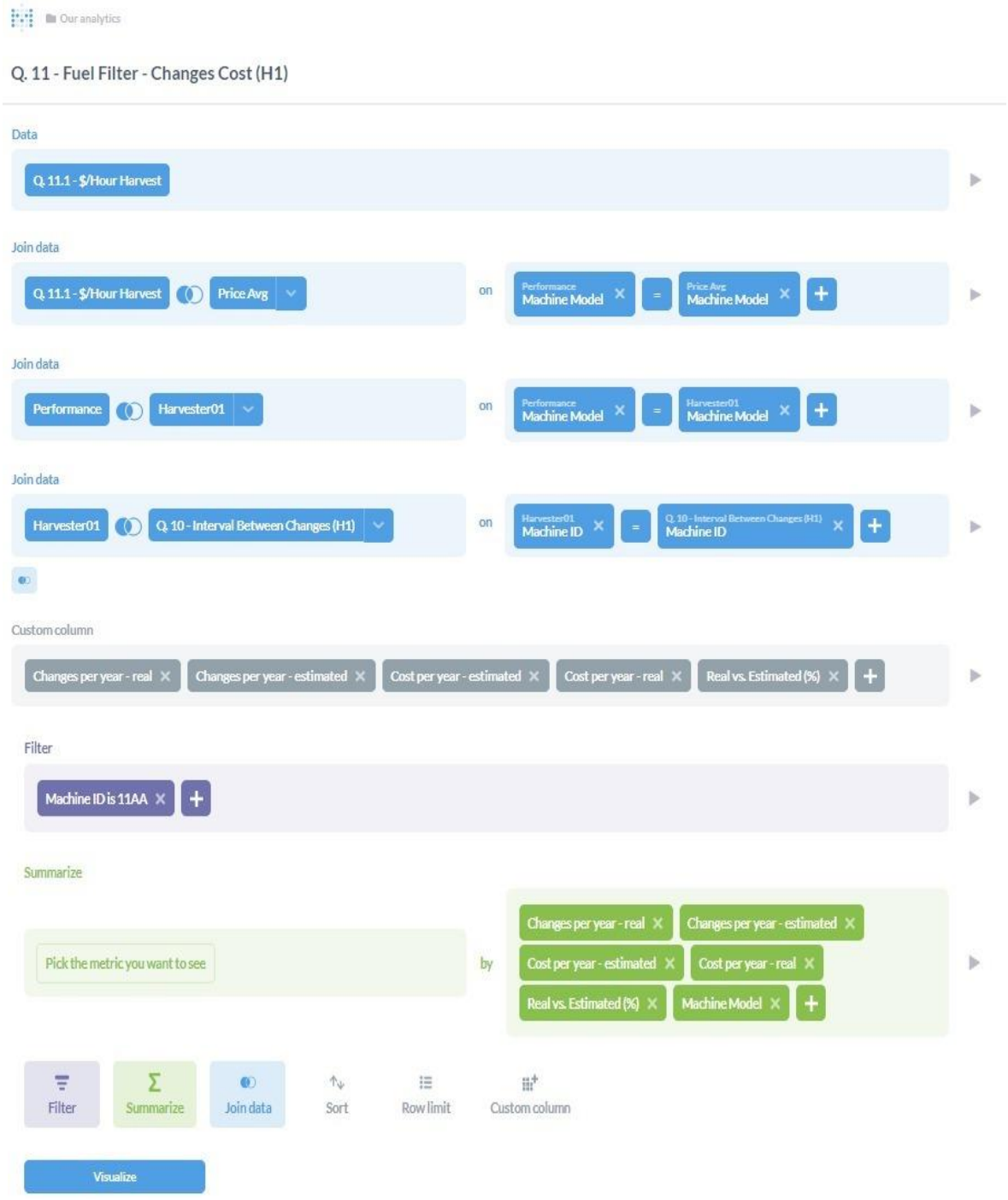
Figura 94 - *Q. 11.1 - $/Hour Harvest*



Fonte: A autora

Dessa maneira, foi possível iniciar a elaboração da *query Q. 11 – Fuel Filter Cost* (Figura 95). Primeiramente realizamos três junções de tabelas (função *Join*), utilizando o resultado da *query* anterior (*Q. 11.1 - $/Hour Harvest*), as tabelas *PriceAvg* e *Harvester*, e a resposta da *query Q. 10 – Interval Between Changes*.

Figura 95 - Estrutura da *query Q. 11 – Fuel Filter Cost*



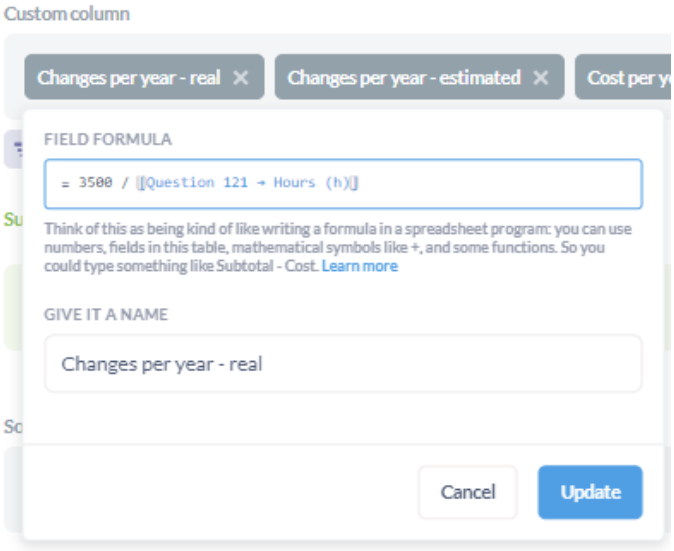
Fonte: A autora

Após, criamos cinco colunas indicando o total de trocas de filtro por ano – realizado, total de trocas de filtro por ano – estimado, custo com trocas de filtro por ano – real, custo com trocas de filtro por ano – estimado, e comparação entre realizado *versus* estimado.

Ao criamos a coluna *Changes per year – real*, consideramos que a colheitadeira trabalha em média 3.500 horas por ano, assim para saber quantas trocas estavam sendo

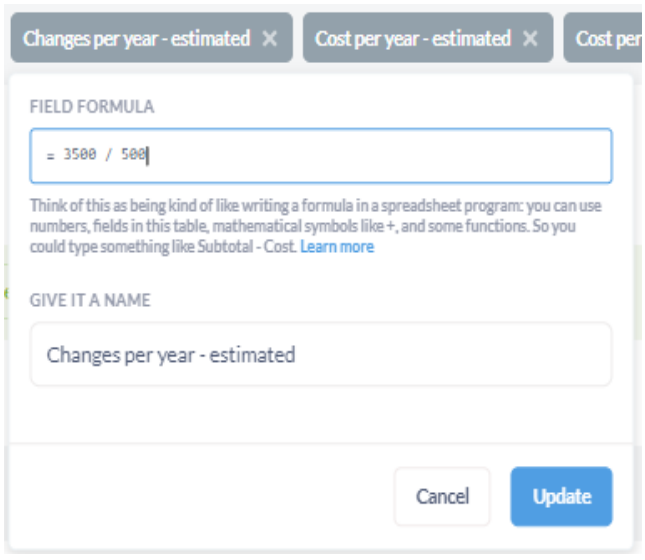
realizadas ao ano dividimos o total de horas trabalhadas pelo valor do intervalo médio de horas entre trocas (Figura 96). Para a coluna *Changes per year – estimated*, calculamos a quantidade de trocas de filtro por ano considerando o total de horas trabalhadas em média por ano pela vida útil dos filtros estipulada pelo fabricante (500 horas), conforme mostra a Figura 97.

Figura 96 - Custom column - Changes per year – real



Fonte: A autora

Figura 97 - Custom column - Changes per year – estimated

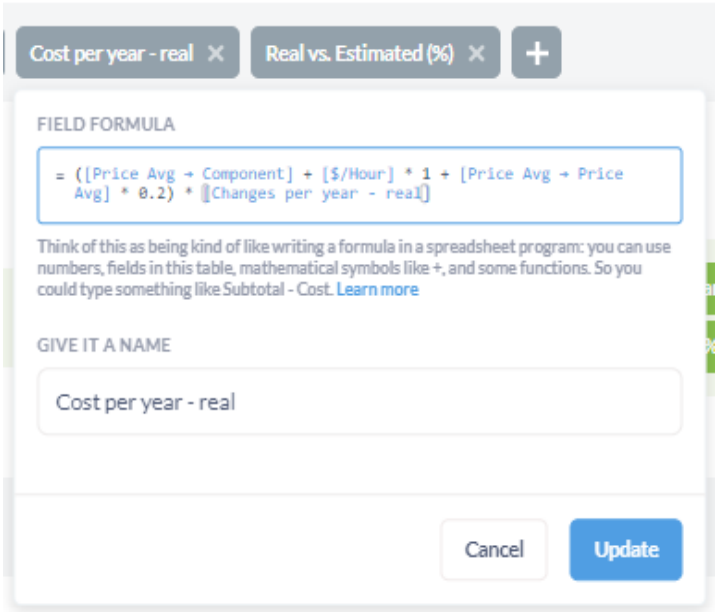


Fonte: A autora

Para calcular o custo com as trocas de filtros de combustível por ano, foram somados os custos do filtro, do tempo que a colheitadeira fica parada para a manutenção do componente e o custo da mão de obra. Após somados os custos foram multiplicados pela quantidade de trocas ao ano. Consideramos que a colheitadeira fica uma hora parada para a realização da troca do filtro, bem como que o custo da mão de obra é de vinte por cento do valor do filtro.

Tanto o custo anual real com as trocas quanto o estimado, foram calculados da mesma forma, sendo alterada apenas a quantidade de trocas por ano de acordo com o custo que estava sendo calculado. A Figura 98 exemplifica essa expressão utilizada na função *Custom column*. Por fim, os resultados foram agrupados pelas colunas criadas e pela coluna *Machine_ID*, através da função *Summarize*, e a formatação condicional foi utilizada para enfatizar os custos caso o valor gasto extrapole o valor estimado. A Figura 99 apresenta o resultado da *query*.

Figura 98 - *Custom column - Cost per year - real*



Fonte: A autora

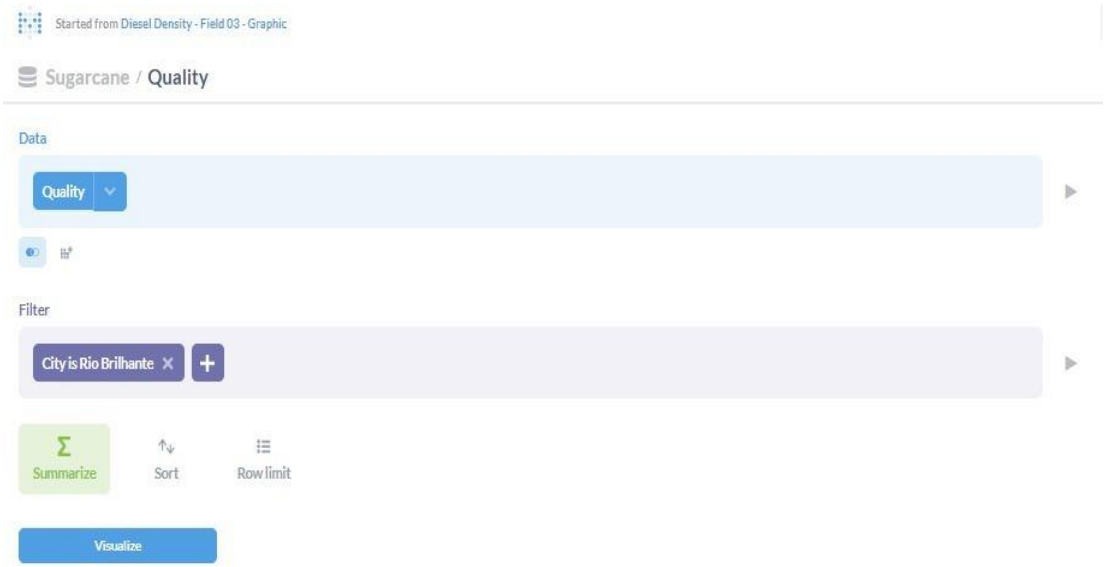
Figura 99 - *Q. 11 – Fuel Filter Cost*

Fonte: A autora

Também criamos algumas *queries* levando em conta a qualidade do combustível em cada uma das três fazendas consideradas, já que esse é um aspecto muito importante quando se trata da saturação dos filtros de combustível.

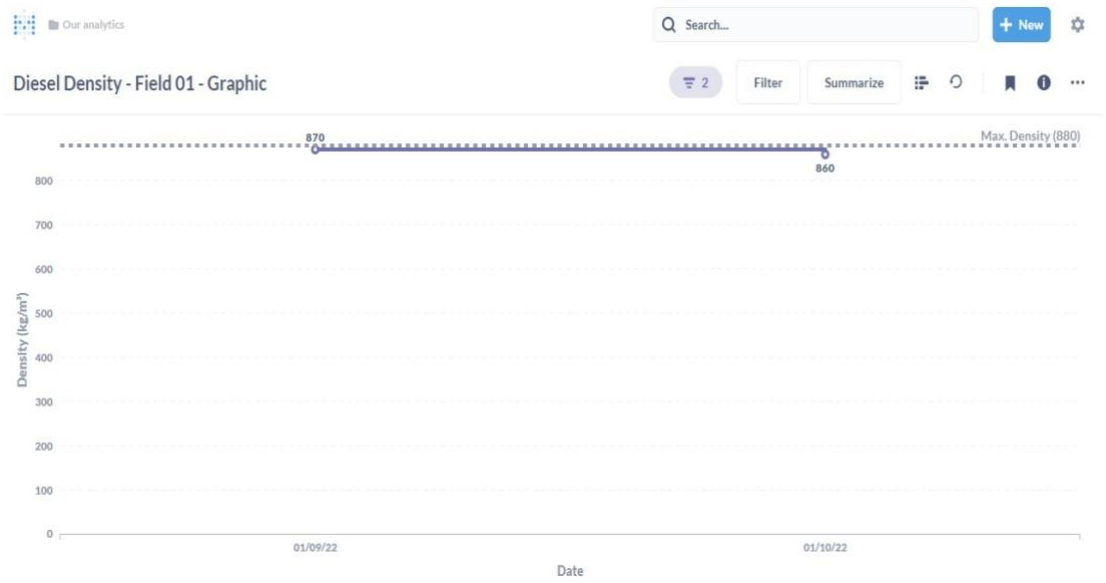
As *queries* utilizaram os dados da tabela *Quality* como dados de entrada e utilizaram a função *filter*, a qual filtrou os dados que seriam apresentados, conforme é exemplificado na Figura 100. As *queries Q.12 – Diesel Density* e *Q.13 – Diesel Particles*, apresentaram de maneira gráfica o histórico dos resultados das análises realizadas, conforme pode ser observado nas Figuras Figura **101** e Figura **102**, respectivamente.

Figura 100 - Estrutura da *query Q.12 – Diesel Density*



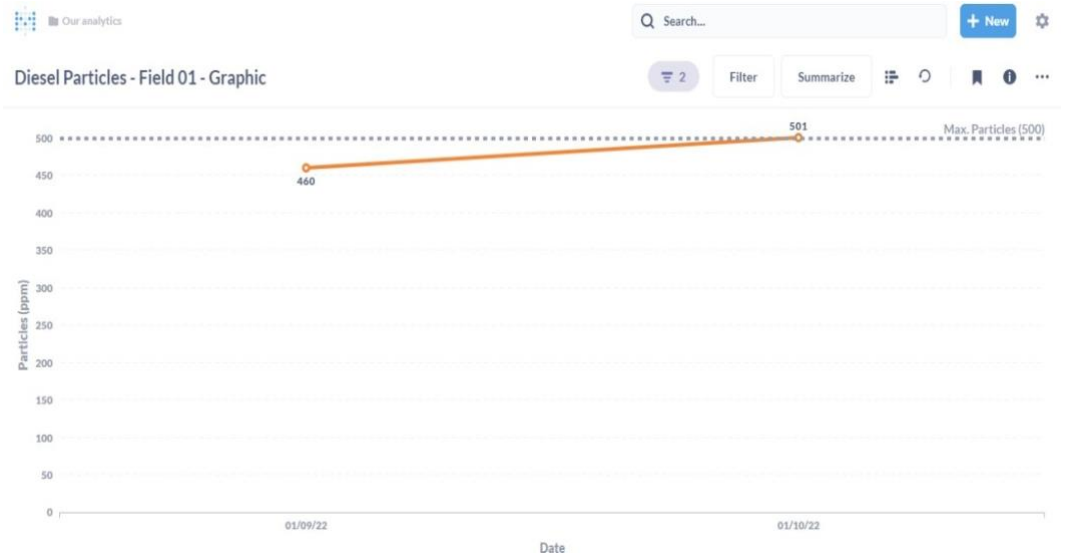
Fonte: A autora

Figura 101 - *Q.12 – Diesel Density*



Fonte: A autora

Figura 102 - *Q.13 – Diesel Particles*



Fonte: A autora

Por sua vez, a *query Q.14 – Diesel Quality* apresenta os dados da tabela *Quality* de maneira visual, através da utilização da formatação condicional para realçar as informações, representando com a cor verde os resultados de análise que estão dentro dos níveis considerados aceitáveis, e vermelha para aqueles resultados que não cumprem os requisitos da qualidade do combustível. A resposta dessa *query* é exemplificada na Figura 103.

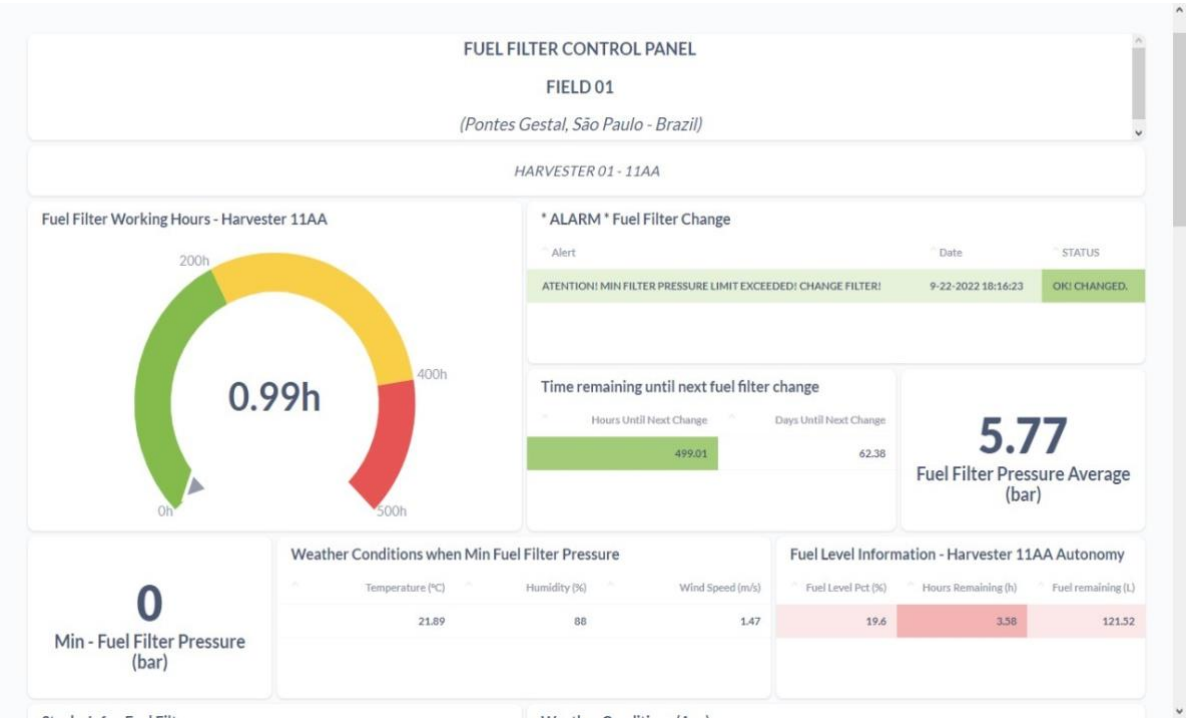
Figura 103 - *Q.14 – Diesel Quality*

Fonte: A autora

Isto posto, considerando o cenário de aplicação, replicamos as primeiras onze *queries* apresentadas para as seis colheitadeiras, o restante replicamos para as três fazendas representadas no cenário. Após, criamos um *dashboard* para cada fazenda, contendo as informações das suas respectivas colheitadeiras. Nesses *dashboards* agrupamos as *queries* considerando as informações contidas e dispostas de maneira a facilitar a visualização dos tomadores de decisão. A Figura 104 exemplifica uma parte do *dashboard* da fazenda localizada em Campos Gestal. O *dashboard* dessa fazenda pode ser observado em detalhe no Apêndice F.

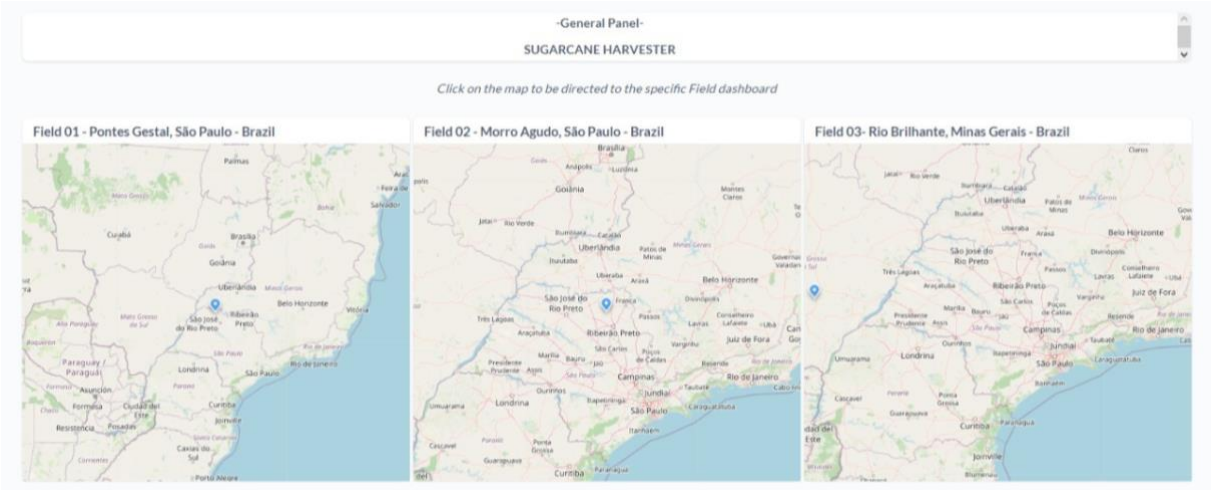
Como estavam sendo retratadas três fazendas distintas, para facilitar as análises, decidimos criar um *dashboard* inicial para que fosse possível selecionar a fazenda a ser analisada, conforme mostra a Figura 105. Ao clicar no mapa, o usuário é direcionado para o *dashboard* contendo as informações das colheitadeiras da cidade escolhida.

Figura 104 - Exemplo de *dashboard*



Fonte: A autora

Figura 105 - *Dashboard* inicial



Fonte: A autora

O objetivo da solução proposta - apoiar as tomadas de decisão relativas à manutenção preditiva das colheitadeiras de cana-de-açúcar- foi atingido e apresentado através da criação das *queries* e *dashboards*, todavia buscamos demonstrar que através da utilização de algoritmos de *Machine Learning,* seria possível utilizar a estrutura desenvolvida para a predição de dados.

Para tal, realizamos a conexão entre a base de dados *MySQL* e a aplicação *JupyterLab*, possibilitando que os dados das tabelas fossem acessados. Após, escolhemos as tabelas que continham os dados das colheitadeiras (tabela *Harvester*) para demonstrar a possibilidade de predição dos dados. Através de um mapa de calor, analisamos as correlações entre as colunas da tabela *Harvester*. Selecionamos então, quatro colunas que apresentavam maior correlação entre si, são elas: *Ground_Speed*, *Cruise_Command*, *Basecutter_height* e *Eng_Load*.

Definimos também que o modelo iria predizer o valor do campo *Eng_Load*, a partir dos valores dos campos das outras três colunas. Em seguida, dividimos os dados de maneira randômica, utilizando trinta e três por cento deles para testar o algoritmo e o restante para treiná-lo. Assim, por meio do algoritmo de *Machine Learning Random Forest*, foi gerado um modelo de classificação de dados. Portanto, após rodar o algoritmo, foi possível predizer o valor da carga do motor. No entanto, ao realizar o cálculo do erro quadrático médio, MSE (da sigla em inglês *Mean Squared Error*), para verificar a acurácia do modelo, percebemos que o dado obtido não é confiável. Esse fato justificou-se visto que a base utilizada não contém dados reais. O código utilizado para realizar a predição pode ser analisado no Apêndice G.

Através dessa etapa foi possível atingir o sexto objetivo específico desta pesquisa, que buscava demonstrar a aplicação da solução por meio de provas de conceito no contexto da empresa parceira. Então, seguindo a abordagem metodológica adotada por este trabalho, partiu-se para a etapa de avaliação da solução, a qual está descrita da seção 4.3.

4.3 Avaliação da solução

Nesta seção, a avaliação da solução proposta é apresentada. Após a demonstração é necessário avaliar a solução para identificar se os objetivos propostos na segunda etapa do DSR foram alcançados. Conforme definido na seção 3.3.5, a solução foi avaliada de maneira descritiva, quanto a sua generalidade, eficiência e aplicabilidade.

O modelo de informação proposto nesta pesquisa foi desenvolvido utilizando o contexto de uma fazenda de cana-de-açúcar, no entanto, ele pode vir a ser utilizado por qualquer outro tipo de fazenda, uma vez que a estrutura e os caminhos de informação criados são genéricos. Essa generalidade pode ser observada nas diferentes guias criadas no Node-RED ao longo da construção do modelo, em que diversos tipos de informação, vindas de fontes diferentes foram tratadas da mesma maneira, sendo customizadas nos fluxos apenas as particularidades de cada fonte de informação. Ademais, a demonstração do modelo utilizou o filtro de combustível da colheitadeira de cana como caso de uso, porém a construção das

queries no Metabase e a análise das relações entre cada tabela da base de dados seguiu as relações descritas no Modelo Sistêmico de Fazenda 4.0.

A solução também foi avaliada quanto a sua eficiência, ou seja, buscou-se avaliar se foram atingidos os objetivos definidos na segunda etapa do DSR. De acordo com Lacerda et al. (2013), em pesquisas desenvolvidas utilizando essa abordagem é necessário evidenciar o funcionamento do artefato criado com a resolução de problemas reais. Ainda, segundo os autores, em cada etapa da abordagem deve ser realizada uma avaliação parcial dos resultados. Dessa maneira, a avaliação da solução quanto a sua eficiência ocorreu ao longo de todo o processo de desenvolvimento da solução, para garantir que estava sendo construída uma solução que cumpriria de fato com os objetivos traçados.

Por fim, a aplicabilidade foi avaliada através da etapa de demonstração da solução. Por meio de reuniões com as pessoas envolvidas na área de inovação e especialistas em manutenção de colheitadeiras de cana-de-açúcar da empresa parceira, onde os *dashboards* foram apresentados, e com os feedbacks recebidos, foi possível validar a aplicabilidade da solução e atingir o sétimo objetivo específico desta pesquisa.

5 CONCLUSÕES

Com o desenvolvimento tecnológico dos últimos anos ocorreram diversas mudanças no setor agrícola, fazendo com que se busque novas maneiras de obter maior produtividade com custos menores. Para isso, se faz necessário tomar decisões melhores e realizar um controle gerencial eficiente por meio de informações e conhecimentos gerados na fazenda em seu contexto específico (WOLFERT et al., 2017). Através da aplicação de novas tecnologias é possível coletar e processar informações, que auxiliem as tomadas de decisão estratégicas e gerenciais envolvidas nos processos (PIVOTO et al., 2018).

Considerando todas as mudanças tecnológicas e um mercado cada vez mais competitivo, há a necessidade de se tomar decisões mais rápidas e acertadas. Devido a essa demanda, buscamos entender como a criação de um modelo sistêmico de fazenda 4.0 poderia facilitar a utilização dos conceitos da quarta revolução industrial para auxiliar tomadas de decisão nas fazendas.

Entendemos que as tomadas de decisão presentes em todos os níveis organizacionais deveriam ser consideradas, portanto a solução proposta por esta pesquisa foi composta por três artefatos: um modelo que caracterizasse a Agricultura 4.0; um modelo que retratasse todos os agentes de uma fazenda orientada à Agricultura 4.0 - incluindo suas tecnologias e relações; e um modelo de informação que representasse um cenário agrícola genérico que possibilitasse a análise de dados e a predição de eventos. Cada artefato buscou auxiliar as decisões de um dos níveis organizacionais das empresas.

O primeiro artefato desenvolvido – Modelo de Referência para Agricultura 4.0 – auxilia as tomadas de decisões estratégicas, pois apresenta uma visão global das tecnologias 4.0 envolvidas no cenário agrícola. Esse modelo, baseado no RAMI 4.0, retrata uma visão integrada de alto nível que guia a implementação da Agricultura 4.0, servindo de base para qualquer tipo de atividade agrícola.

Após representar as tecnologias necessárias para a implantação da Agricultura 4.0, criamos o segundo artefato: Modelo Sistêmico de Fazenda 4.0. Esse modelo considera uma fazenda inserida no contexto de Agricultura 4.0, e representa todos os agentes e todas as ligações entre eles e o meio onde estão inseridos. Com essa representação é possível apoiar as decisões táticas, uma vez que o Modelo foca nos processos que fazem parte da Fazenda 4.0.

O último artefato criado foi um Modelo de Informação, no qual consideramos os processos de extração, transformação e carregamento de dados, para gerar informações no contexto de uma fazenda de cana-de-açúcar. Para desenvolvermos esse modelo consideramos

a manutenção do filtro de combustível de uma colheitadeira de cana, portanto os dados de entrada do modelo foram extraídos de sensores do equipamento e de sensores de clima.

Depois dos dados serem extraídos, eles foram transformados e inseridos em uma base de dados, o que viabilizou relacioná-los, considerando as ligações representadas pelo Modelo Sistêmico de Fazenda 4.0. Por meio dessas relações, realizamos buscas e apresentamos as respostas em um *dashboard* com diversas informações, que contribuem para tomadas de decisão operacionais da organização (i.e., troca do filtro de combustível).

Assim, por meio da criação desses três modelos, atingimos o objetivo dessa pesquisa de desenvolver um modelo sistêmico capaz de selecionar e interpretar dados originados de sensores presentes em implementos agrícolas para controlar e prever a necessidade de manutenção dos equipamentos, auxiliando tomadas de decisão de maneira preditiva. Além disso, construímos esse Modelo de Informação utilizando o conceito de conteinerização, por meio da plataforma Docker, o que possibilita a portabilidade do modelo e a sua pronta utilização pela empresa parceira.

Nas etapas de demonstração e avaliação da solução, constatou-se que a solução criada havia alcançado o objetivo proposto no início desse trabalho, uma vez que foram desenvolvidos três artefatos, cada um apoiando as tomadas de decisão de um dos níveis organizacionais. Os modelos foram testados e aplicados em um cenário de manutenção de filtros de combustível de colheitadeiras de cana. Por meio da solução criada, foram obtidas diversas informações que podem ajudar os tomadores de decisão, tanto no momento de implementar tecnologias da Agricultura 4.0, como para entender de maneira sistêmica as relações entre os agentes de uma Fazenda 4.0, ou até mesmo para tomar decisões operacionais.

A utilização da abordagem DSR garantiu a delimitação das etapas a serem seguidas para o desenvolvimento da solução, além de garantir que o modelo proposto neste trabalho fosse aplicado e avaliado em um contexto agrícola real. Contar com a participação de uma empresa ao longo do desenvolvimento desta pesquisa foi muito relevante. A importância da empresa parceira na realização do trabalho justifica-se por uma parte significativa dos dados utilizados serem dados reais por ela disponibilizados. Além disso, a troca de informações durante as reuniões e os *feedbacks* recebidos ao longo da demonstração e avaliação da solução possibilitaram criar um modelo que pudesse, de fato, ser utilizado posteriormente pela organização.

Este trabalho teve como foco colheitadeiras de cana-de-açúcar, mais especificamente, a realização de manutenção dos filtros de combustível, contudo os modelos são genéricos, ou

seja, é possível utilizá-los para qualquer outro tipo de equipamento e cultura, incluindo novas fontes de dados e novas relações no Modelo de Informação. A injeção dos dados na base de dados do Modelo pode ser realizada de maneira automática, definindo-se o intervalo entre cada injeção de novos dados. Ainda é possível definir se os dados serão sobrescritos ou um novo *dataset* será criado a cada injeção de dados.

Dentre as atividades realizadas neste trabalho, considerando o que aconteceria num cenário real caso houvesse conectividade no campo, a atividade mais desafiadora foi a de criar uma estrutura que emulasse o intervalo de tempo entre a entrada de cada dado. Outro ponto desafiador, foi o tempo dispendido na criação dos *datasets*, uma vez que era necessária uma quantidade significativa de registros para realizar a demonstração da solução, e para isso, os dados deveriam ser injetados um a um. A limitação do trabalho também se encontra relacionada aos dados de entrada do Modelo, uma vez que nem todos os dados utilizados são dados reais devido as questões de falta de conectividade e informações sigilosas.

Diante disso, como recomendações futuras sugerimos utilizar *datasets* compostos inteiramente por dados de entrada reais. Outra sugestão é utilizar *datasets* maiores para que as informações geradas sejam mais confiáveis. Ainda, é interessante testar a solução em uma plataforma de computação na nuvem, e prever trocas de manutenção por meio de algoritmos de *machine learning*.

REFERÊNCIAS

ABBASI, A. Z.; ISLAM, N.; SHAIKH, Z. A. A review of wireless sensors and networks' applications in agriculture. **Computer Standards & Interfaces**, v. 36, n. 2, p. 263–270, 2014.

ABHIJITH, H. V; JAIN, D. A.; ADITHYA ATHREYA RAO, U. **Intelligent agriculture mechanism using internet of things**. 2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017. **Anais**...2017. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85042650973&doi=10.1109%2FICACCI.2017.8126169&partnerID=40&md5=0db28815340960a3f96bab5d3f485cbd>

ABUABDO, A.; AL-SHARIF, Z. A. **Virtualization vs. Containerization: Towards a Multithreaded Performance Evaluation Approach**. 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA). **Anais**...2019.

ADOLPHS, P., BEDENBENDER, H., DIRZUS, D., EHLICH, M., EPPLE, U., HANKEL, M., WOLLSCHLAEGER, M. **Reference Architecture Model Industrie 4.0 (RAMI 4.0). VDI/VDE and ZVEI.** , 2015. Disponível em: <https://www.zvei.org/en/subjects/industry-4-0/the-reference-architectural-model-rami-40-and-the-industrie-40- component/>

ALAM, K. M.; EL SADDIK, A. C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems. **IEEE access**, v. 5, p. 2050–2062, 2017.

ALANSARI, Z. et al. **Challenges of internet of things and big data integration**. International Conference for Emerging Technologies in Computing. **Anais**...Springer, 2018.

ANDRITOIU, D. et al. **Agriculture autonomous monitoring and decisional mechatronic system**. 2018 19th International Carpathian Control Conference (ICCC). **Anais**...2018.

BANSAL, S. K. **Towards a semantic extract-transform-load (ETL) framework for big data integration**. 2014 IEEE International Congress on Big Data. **Anais**...IEEE, 2014.

BELSA, A. et al. **Flow-based programming interoperability solution for IoT platform applications**. 2018 IEEE International Conference on Cloud Engineering (IC2E). **Anais**...IEEE, 2018.

BEN-DAYA, M. et al. **Handbook of maintenance management and engineering**. [s.l.] Springer, 2009. v. 7

BENDRE, M. R.; THOOL, R. C.; THOOL, V. R. **Big data in precision agriculture: Weather forecasting for future farming**. Proceedings on 2015 1st International Conference on Next Generation Computing Technologies, NGCT 2015. **Anais**...2016.

BENTALEB, O. et al. Containerization technologies: taxonomies, applications and challenges. **The Journal of Supercomputing**, v. 78, n. 1, p. 1144–1181, 2022.

BERGAMASCHI, S. et al. A semantic approach to ETL technologies. **Data & Knowledge Engineering**, v. 70, n. 8, p. 717–731, 2011.

BERMEJO, B.; JUIZ, C.; GUERRERO, C. Virtualization and consolidation: a systematic review of the past 10 years of research on energy and performance. **The Journal of Supercomputing**, v. 75, n. 2, p. 808–836, 2019.

BLACKSTOCK, M.; LEA, R. **Toward a distributed data flow platform for the web of things (distributed node-red)**. Proceedings of the 5th International Workshop on Web of Things. **Anais**...2014.

BOETTIGER, C. An introduction to Docker for reproducible research. **ACM SIGOPS Operating Systems Review**, v. 49, n. 1, p. 71–79, 2015.

BONAT, D. **Metodologia da pesquisa**. [s.l.] IESDE BRASIL SA, 2009.

BONNEAU, V. et al. Industry 4.0 in Agriculture: Focus on IoT Aspects. **European Comission**, 2017.

BOOCH, G. UML in action. **Communications of the ACM**, v. 42, n. 10, p. 26–28, 1999.

CHANTHAKIT, S.; KEERATIWINTAKORN, P.; RATTANAPOKA, C. **An iot system design with real-time stream processing and data flow integration**. 2019 Research, Invention, and Innovation Congress (RI2C). **Anais**...IEEE, 2019.

CHIAVENATO, I. **Introdução à teoria geral da administração**. [s.l.] Elsevier Brasil, 2003.

CHIAVENATO, I. **Administração Teoria, processo e pratica. Rio de Janeiro, RJ**. Elsevier, , 2007.

COELHO, P. M. N. **Rumo à indústria 4.0**. , 2016.

DA SILVA, A. C. F. et al. **OpenTOSCA for IoT: automating the deployment of IoT applications based on the mosquitto message broker**. Proceedings of the 6th International Conference on the Internet of Things. **Anais**...2016.

DAN, L. et al. **Intelligent Agriculture Greenhouse Environment Monitoring System Based on the Android Platform**. 2017 International Conference on Smart Grid and Electrical Automation (ICSGEA). **Anais**...IEEE, 2017.

DELLIGATTI, L. **SysML distilled: A brief guide to the systems modeling language**. [s.l.] Addison-Wesley, 2013.

DEWI, C.; CHEN, R.-C. **Decision making based on IoT data collection for precision agriculture**. **Studies in Computational Intelligence**, 2020.

DIOUF, P. S.; BOLY, A.; NDIAYE, S. **Variety of data in the ETL processes in the cloud: State of the art**. 2018 IEEE International Conference on Innovative Research and Development (ICIRD). **Anais**...IEEE, 2018.

DONZIA, S. K. Y.; KIM, H.-K.; HWANG, H. J. **A software model for precision agriculture framework based on smart farming system and application of IoT gateway**. **Studies in Computational Intelligence**, 2019.

DUA, R.; RAJA, A. R.; KAKADIA, D. **Virtualization vs Containerization to Support PaaS**. 2014 IEEE International Conference on Cloud Engineering. **Anais**...2014.

ELENA, C. Business intelligence. **Journal of Knowledge Management, Economics and Information Technology**, v. 1, n. 2, p. 1–12, 2011.

ELIJAH, O. et al. An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges. **IEEE Internet of Things Journal**, v. 5, n. 5, p. 3758–3773, 2018.

ENNIS, C. et al. **A conceptual framework for servitization in Industry 4.0: Distilling directions for future research**. The Advance Services Group Spring Servitization Conference 2018. **Anais**...Aston University and Higher Education Academy, 2018.

EYADA, M. M. et al. Performance Evaluation of IoT Data Management Using MongoDB Versus MySQL Databases in Different Cloud Environments. **IEEE Access**, v. 8, p. 110656–110668, 2020.

FATHONI, F. et al. Fundraising decision support system on Indonesias oil palm public service agency using kimball-ross four-step dimensional process and metabase dashboard. **Journal of Theoretical and Applied Information Technology**, v. 100, n. 14, 2022.

FERENCZ, K.; DOMOKOS, J. Using Node-RED platform in an industrial environment. **XXXV. Jubileumi Kandó Konferencia, Budapest**, p. 52–63, 2019.

FLANDERS, J.; JANNIDIS, F. Data modeling. **A new companion to digital humanities**, p. 229–237, 2015.

FOWLER, M. **UML Essencial: um breve guia para linguagem padrão**. [s.l.] Bookman editora, 2014.

FRESCO, R.; FERRARI, G. Enhancing precision agriculture by Internet of Things and cyber physical systems. **Atti della Societa Toscana di Scienze Naturali, Memorie Serie B**, v. 125, p. 53–60, 2018.

FRIEDENTHAL, S.; MOORE, A.; STEINER, R. **A practical guide to SysML: the systems modeling language**. [s.l.] Morgan Kaufmann, 2014.

GHATREHSAMANI, D. et al. **The art of cpu-pinning: Evaluating and improving the performance of virtualization and containerization platforms**. 49th international conference on parallel processing-ICPP. **Anais**...2020.

GRIGORYEV, I. AnyLogic 7 in three days. **A quick course in simulation modeling**, v. 2, 2015.

GUIMARÃES, E. M. P.; ÉVORA, Y. D. M. Sistema de informação: instrumento para tomada de decisão no exercício da gerência. **Ciência da informação**, v. 33, n. 1, p. 72–80, 2004.

HAGINO, T. **Practical Node-RED Programming: Learn powerful visual programming techniques and best practices for the web and IoT**. [s.l.] Packt Publishing Ltd, 2021.

HAO, F.; ZHAO, X.; WANG, M. Intelligent agricultural machinery monitoring system based on the cloud. **Advances in Intelligent Systems and Computing**, v. 613, p. 92–99, 2018.

HART, L. E. **Introduction to model-based system engineering (MBSE) and SysML**. Delaware Valley INCOSE Chapter Meeting. **Anais**...Ramblewood Country Club Mount Laurel, New Jersey, 2015.

HAUSE, M. **The SysML modelling language**. Fifteenth European Systems Engineering Conference. **Anais**...2006.

HERMANN, M.; PENTEK, T.; OTTO, B. **Design principles for industrie 4.0 scenarios**. 2016 49th Hawaii international conference on system sciences (HICSS). **Anais**...IEEE, 2016.

HEVNER, A.; CHATTERJEE, S. **Design research in information systems: theory and practice**. [s.l.] Springer Science & Business Media, 2010. v. 22

HIMESH, S. et al. Digital revolution and Big Data: A new revolution in agriculture. **CAB Reviews: Perspectives in Agriculture, Veterinary Science, Nutrition and Natural Resources**, v. 13, 2018.

HOSSAIN, N. U. I. et al. Modeling and Analysis of Unmanned Aerial Vehicle System leveraging Systems Modeling Language (SysML). **Systems**, v. 10, n. 6, p. 264, 2022.

HUANG, E.; RAMAMURTHY, R.; MCGINNIS, L. F. **System and simulation modeling using SysML**. 2007 Winter Simulation Conference. **Anais**...IEEE, 2007.

HUH, J.-H.; KIM, K.-Y. Time-based trend of carbon emissions in the composting process of swine manure in the context of agriculture 4.0. **Processes**, v. 6, n. 9, p. 168, 2018.

HUSTI, I.; DAROCZI, M.; KOVACS, I. Message from "Industry 4.0" to agriculture. **Towards Sustainable Agriculture and Biosystems Engineering, Universiteas-Gyor Nonprofit Ltd., Mosonmagyaróvár, Hungary**, p. 63–78, 2017.

IAKSCH, J.; FERNANDES, E.; BORSATO, M. **Digitalization and Big Data in Smart Farming–Bibliometric and Systemic Analysis**. Transdisciplinary engineering for complex socio-technical systems—real-life applications: proceedings of the 27th ISTE international conference on transdisciplinary engineering. **Anais**...2020.

IAKSCH, J.; FERNANDES, E.; BORSATO, M. Digitalization and Big data in smart farming–a review. **Journal of Management Analytics**, v. 8, n. 2, p. 333–349, 2021.

JOVANOSKA, D.; NECHKOSKA, R. P.; MANCHESKI, G. **Metabase cockpits as a base for BI in Strategic management**. International Scientific Conference Strategic Management and Decision Support Systems in Strategic Management. **Anais**...2021.

KAMBLE, S. S.; GUNASEKARAN, A.; GAWANKAR, S. A. Sustainable Industry 4.0 framework: A systematic literature review identifying the current trends and future perspectives. **Process Safety and Environmental Protection**, v. 117, p. 408–425, 2018.

KAMILARIS, A. et al. **Agri-IoT: A semantic framework for Internet of Things-enabled smart farming applications**. 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT). **Anais**...2016.

KHODABAKHSHIAN, R. Maintenance management of tractors and agricultural machinery: Preventive maintenance systems. **Agricultural Engineering International: CIGR Journal**, v. 15, n. 4, p. 147–159, 2013.

KODALI, R. K.; ANJUM, A. **IoT based home automation using node-red**. 2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT). **Anais**...IEEE, 2018.

KÖKSAL, Ö.; TEKINERDOGAN, B. Architecture design approach for IoT-based farm management information systems. **Precision Agriculture**, v. 20, n. 5, p. 926–958, 2019.

KOVÁCS, I.; HUSTI, I. The role of digitalization in the agricultural 4.0–how to connect the industry 4.0 to agriculture? **Hungarian Agricultural Engineering**, n. 33, p. 38–42, 2018.

LACERDA, D. P. et al. Design Science Research: método de pesquisa para a engenharia de produção. **Gestão & produção**, v. 20, n. 4, p. 741–761, 2013.

LAMPKIN, V. et al. **Building smarter planet solutions with mqtt and ibm websphere mq telemetry**. [s.l.] IBM Redbooks, 2012.

LAW, A. M.; KELTON, W. D.; KELTON, W. D. **Simulation modeling and analysis**. [s.l.] McGraw-Hill New York, 2000. v. 3

LEE, M.; KIM, H.; YOE, H. **Intelligent environment management system for controlled horticulture**. 2017 4th NAFOSTED Conference on Information and Computer Science, NICS 2017 - Proceedings. **Anais**...2017.

LIM, E.-P.; CHEN, H.; CHEN, G. Business intelligence and analytics: Research directions. **ACM Transactions on Management Information Systems (TMIS)**, v. 3, n. 4, p. 1–10, 2013.

LIMA, R.; CRUZ, E. F. **Extraction and Multidimensional Analysis of Data from Unstructured Data Sources: A Case Study.** ICEIS (1). **Anais**...2019.

LIPS, M.; BUROSE, F. Repair and maintenance costs for agricultural machines. **International Journal of Agricultural Management**, v. 1, n. 1029-2016–82247, p. 40–46, 2012.

LÜTTENBERG, H.; BARTELHEIMER, C.; BEVERUNGEN, D. Designing predictive maintenance for agricultural machines. 2018.

MADUSHANKI, A. A. R. et al. Adoption of the Internet of Things (IoT) in agriculture and smart farming towards urban greening: A review. **International Journal of Advanced Computer Science and Applications**, v. 10, n. 4, p. 11–28, 2019.

MANN, C. J. H. A Practical Guide to SysML: The Systems Modeling Language. **Kybernetes**, v. 38, n. 1/2, 1 jan. 2009.

MARIA, A. **Introduction to modeling and simulation**. Proceedings of the 29th conference on Winter simulation. **Anais**...1997.

MARTIN, A. et al. Docker ecosystem–vulnerability analysis. **Computer Communications**, v. 122, p. 30–43, 2018.

MEDINA-PÉREZ, A.; SÁNCHEZ-RODRÍGUEZ, D.; ALONSO-GONZÁLEZ, I. An Internet of Thing Architecture Based on Message Queuing Telemetry Transport Protocol and Node-RED: A Case Study for Monitoring Radon Gas. **Smart Cities**, v. 4, n. 2, p. 803–818, 2021.

MERKEL, D. Docker: lightweight linux containers for consistent development and deployment. **Linux j**, v. 239, n. 2, p. 2, 2014.

MIELL, I.; SAYERS, A. **Docker in practice**. [s.l.] Simon and Schuster, 2019.

MILLER, I.; KOSSIK, R.; VOSS, C. **General requirements for simulation models in waste management**. [s.l.] Golder Associates Inc, 200-18300 Union Hill Road NE, Redmond, WA 98052 (US), 2003.

MOBLEY, R. K. **An introduction to predictive maintenance**. [s.l.] Elsevier, 2002.

MOHANRAJ, I.; ASHOKUMAR, K.; NAREN, J. Field monitoring and automation using IOT in agriculture domain. **Procedia Computer Science**, v. 93, p. 931–939, 2016.

MONDESIRE, S. C. et al. Combining virtualization and containerization to support interactive games and simulations on the cloud. **Simulation Modelling Practice and Theory**, v. 93, p. 233–244, 2019.

MUANGPRATHUB, J. et al. IoT and agriculture data analysis for smart farm. **Computers and Electronics in Agriculture**, v. 156, p. 467–474, 2019.

MUÑOZ, L.; MAZÓN, J.-N.; TRUJILLO, J. **Automatic generation of ETL processes from conceptual models**. Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP. **Anais**...2009.

NADJ, M. et al. **A Situation Awareness Driven Design for predictive Maintenance Systems: the Case of Oil and gas Pipeline Operations.** ECIS. **Anais**...2016.

NARIMOTO, L. R.; BELUSSI, S. E. A. C. **The Role of Design in Use in Agriculture: The Case of Brazilian Crops**. Congress of the International Ergonomics Association. **Anais**...Springer, 2018.

NIKOLIC, B. et al. PREDICTIVE MANUFACTURING SYSTEMS IN INDUSTRY 4.0: TRENDS, BENEFITS AND CHALLENGES. **Annals of DAAAM & Proceedings**, v. 28, 2017.

NIU, L.; LU, J.; ZHANG, G. Cognition-driven decision support for business intelligence. **Models, Techniques, Systems and Applications. Studies in Computational Intelligence, Springer, Berlin**, p. 4–5, 2009.

NUKALA, R. et al. **Internet of Things: A review from "Farm to Fork"**. 2016 27th Irish Signals and Systems Conference, ISSC 2016. **Anais**...2016.

O'GRADY, M. J.; O'HARE, G. M. P. Modelling the smart farm. **Information processing in agriculture**, v. 4, n. 3, p. 179–187, 2017.

ORTIZ, G. et al. Evaluating a Flow-Based Programming Approach as an Alternative for Developing CEP Applications in IoT. **IEEE Internet of Things Journal**, v. 9, n. 13, p. 11489–11499, 2022.

OZDOGAN, B.; GACAR, A.; AKTAS, H. Digital agriculture practices in the context of agriculture 4.0. **Journal of Economics, Finance and Accounting (JEFA)**, v. 4, p. 184–191, 2017.

OZTEMEL, E.; GURSEV, S. Literature review of Industry 4.0 and related technologies. **Journal of Intelligent Manufacturing**, v. 31, n. 1, p. 127–182, 2020.

PARAISO, F. et al. **Model-driven management of docker containers**. 2016 IEEE 9th International Conference on cloud Computing (CLOUD). **Anais**...IEEE, 2016.

PEFFERS, K. et al. A design science research methodology for information systems research. **Journal of management information systems**, v. 24, n. 3, p. 45–77, 2007.

PISCHING, M. A. et al. An architecture based on RAMI 4.0 to discover equipment to process operations required by products. **Computers & Industrial Engineering**, v. 125, p. 574–591, 2018.

PIVOTO, D. et al. Scientific development of smart farming technologies and their application in Brazil. **Information Processing in Agriculture**, v. 5, n. 1, p. 21–32, 2018.

POPEANGĂ, J.; LUNGU, I. Real-time business intelligence for the utilities industry. **Database systems journal**, v. 3, n. 4, p. 15–24, 2012.

POSADA, J. et al. Visual computing as a key enabling technology for industrie 4.0 and industrial internet. **IEEE computer graphics and applications**, v. 35, n. 2, p. 26–40, 2015.

PREETH, E. N. et al. **Evaluation of Docker containers based on hardware utilization**. 2015 international conference on control communication & computing India (ICCC). **Anais**...IEEE, 2015.

RAD, B. B.; BHATTI, H. J.; AHMADI, M. An introduction to docker and analysis of its performance. **International Journal of Computer Science and Network Security (IJCSNS)**, v. 17, n. 3, p. 228, 2017.

RAINS, G. C.; THOMAS, D. L. Precision farming: An introduction. 2009.

RAJESWARI, S.; SUTHENDRAN, K.; RAJAKUMAR, K. **A smart agricultural model by integrating IoT, mobile and cloud-based big data analytics**. Proceedings of 2017 International Conference on Intelligent Computing and Control, I2C2 2017. **Anais**...2018.

RAUTENBERG, S.; DO CARMO, P. R. V. Big Data e Ciência de Dados: complementariedade conceitual no processo de tomada de decisão. **Brazilian Journal of Information Science**, v. 13, n. 1, p. 56–67, 2019.

RAY, P. P. Internet of things for smart agriculture: Technologies, practices and future direction. **Journal of Ambient Intelligence and Smart Environments**, v. 9, n. 4, p. 395–420, 2017.

RESMAN, M. et al. A new architecture model for smart manufacturing: A performance analysis and comparison with the RAMI 4.0 reference model. **Adv. Prod. Eng. Manag**, v. 14, n. 2, p. 153–165, 2019.

ROJKO, A. Industry 4.0 concept: background and overview. **International Journal of Interactive Mobile Technologies (iJIM)**, v. 11, n. 5, p. 77–90, 2017.

ROTZ, S. et al. The Politics of Digital Agricultural Technologies: A Preliminary Review. **Sociologia Ruralis**, v. 59, n. 2, p. 203–229, 2019.

RÜSSMANN, M. et al. Industry 4.0: The future of productivity and growth in manufacturing industries. **Boston Consulting Group**, v. 9, n. 1, p. 54–89, 2015.

SANDERS, A.; ELANGESWARAN, C.; WULFSBERG, J. P. Industry 4.0 implies lean manufacturing: Research activities in industry 4.0 function as enablers for lean manufacturing. **Journal of Industrial Engineering and Management (JIEM)**, v. 9, n. 3, p. 811–833, 2016.

SCHEEPERS, M. J. **Virtualization and containerization of application infrastructure: A comparison**. 21st twente student conference on IT. **Anais**...2014.

SHAMSHIRI, R. R. et al. Research and development in agricultural robotics: A perspective of digital farming. **International Journal of Agricultural and Biological Engineering**, v. 11, n. 4, p. 1–14, jul. 2018.

SHENOY, J.; PINGLE, Y. **IOT in agriculture**. Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development, INDIACom 2016. **Anais**...2016.

SILVA, R. P. DA et al. Controle estatístico aplicado ao processo de colheita mecanizada de cana-de-açúcar. **Engenharia Agrícola**, p. 292–304, 2008.

SIMON, H. A. **The sciences of the artificial**. [s.l.] MIT press, 2019.

STROZZI, F. et al. Literature review on the 'Smart Factory'concept using bibliometric tools. **International Journal of Production Research**, v. 55, n. 22, p. 6572–6591, 2017.

SUAKANTO, S. et al. **Sensor networks data acquisition and task management for decision support of smart farming**. 2016 International Conference on Information Technology Systems and Innovation, ICITSI 2016 - Proceedings. **Anais**...2017.

framework for data extraction, transformation and loading in data warehouse. **International Journal of Advanced Computer Science and Applications**, v. 7, n. 11, 2016.

TASCA, J. E. et al. An approach for selecting a theoretical framework for the evaluation of training programs. **Journal of European industrial training**, 2010.

TRANFIELD, D.; DENYER, D.; SMART, P. Towards a methodology for developing evidence-informed management knowledge by means of systematic review. **British journal of management**, v. 14, n. 3, p. 207–222, 2003.

TRAORÉ, M. K. Chapter 3 - Unified Approaches to Modeling. Em: ZHANG, L.; ZEIGLER, B. P.; LAILI, Y. (Eds.). **Model Engineering for Simulation**. [s.l.] Academic Press, 2019. p. 43–56.

TURNBULL, J. **The docker book**. , 2016.

VASSILIADIS, P.; SIMITSIS, A.; BAIKOUSI, E. **A taxonomy of ETL activities**. Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP. **Anais**...2009.

VERCELLIS, C. **Business intelligence: data mining and optimization for decision making**. [s.l.] Wiley Online Library, 2009.

VON ALAN, R. H. et al. Design science in information systems research. **MIS quarterly**, v. 28, n. 1, p. 75–105, 2004.

WAAS, F. et al. On-demand ELT architecture for right-time BI: extending the vision. **International Journal of Data Warehousing and Mining (IJDWM)**, v. 9, n. 2, p. 21–38, 2013.

WAZLAWICK, R. **Análise e design orientados a objetos para sistemas de informação: Modelagem com UML, OCL e IFML**. [s.l.] Elsevier Brasil, 2016.

WELTZIEN, C. Digital Agriculture or Why Agriculture 4.0 Still Offers Only Modest Returns. **Landtechnik**, v. 71, n. 2, p. 66–68, 2016.

WOLFERT, S. et al. Big Data in Smart Farming – A review. **Agricultural Systems**, v. 153, p. 69–80, 2017.

WOLNY, S. et al. Thirteen years of SysML: a systematic mapping study. **Software and Systems Modeling**, v. 19, n. 1, p. 111–169, 2020.

XIN, J.; ZAZUETA, F. Technology trends in ICT - Towards data-driven, farmer-centered. **Agricultural Engineering International: CIGR Journal**, v. 18, n. 4, p. 275–279, 2016.

YANE, D. **Research and Analysis about System of Digital Agriculture Based on a Network Platform**. International Conference on Computer and Computing Technologies in Agriculture. **Anais**...Springer, 2010.

YARLAGADDA, R. T. Data models in information technology. **INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY [IJIERT]**, 2016.

ZAMBON, I. et al. Revolution 4.0: Industry vs. agriculture in a future development for SMEs. **Processes**, v. 7, n. 1, p. 36, 2019.

ZAMORA-IZQUIERDO, M. A. et al. Smart farming IoT platform based on edge and cloud computing. **Biosystems Engineering**, v. 177, p. 4–17, 2019.

ZHOU, J. et al. Integration and Analysis of Agricultural Market Information Based on Web Mining. **IFAC-PapersOnLine**, v. 51, n. 17, p. 778–783, 2018.

APÊNDICE A – Código do arquivo yaml – *docker compose*

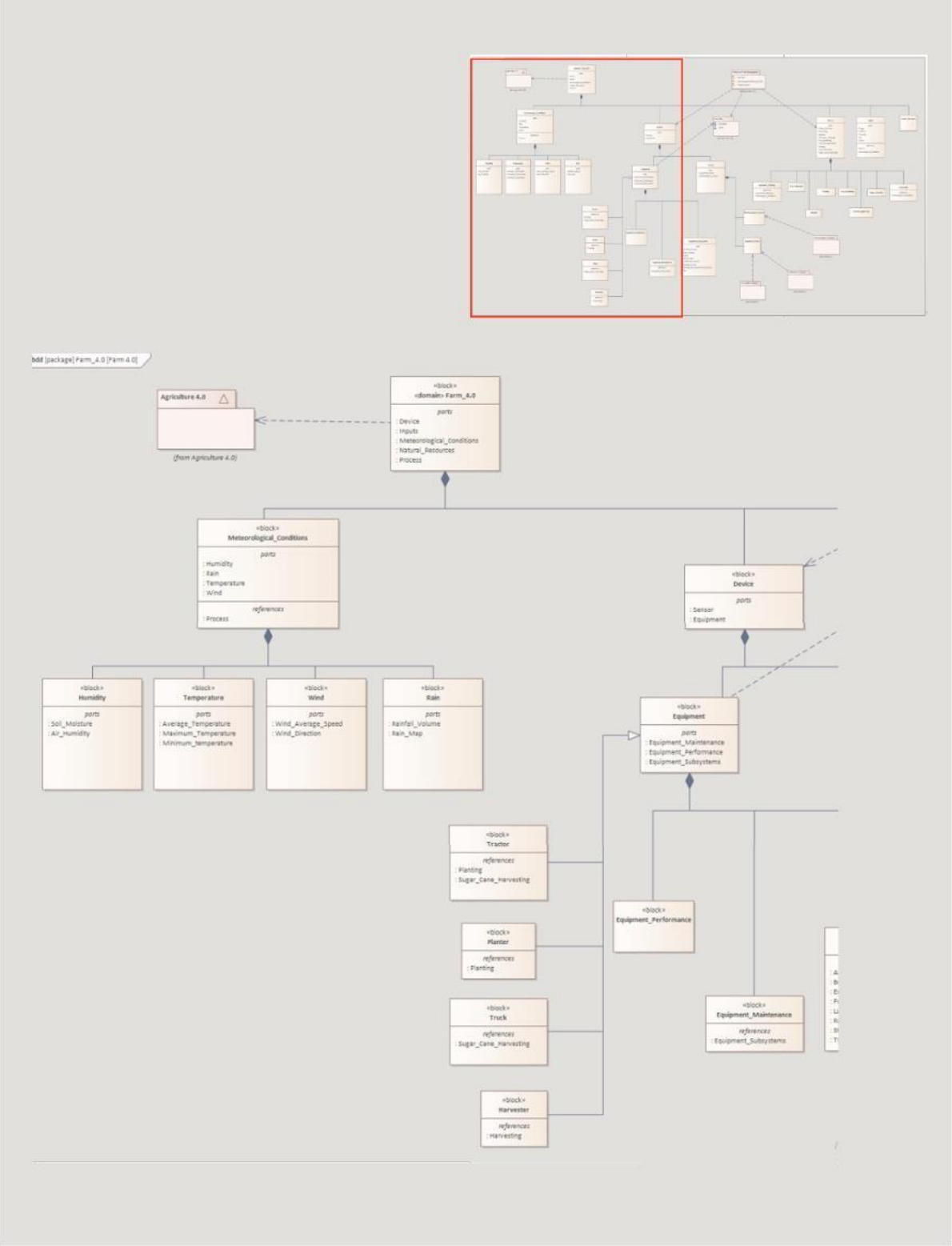
```
services:
  mysql:
    image: mysql/mysql-server:latest
    container_name: MySQL
    hostname: mysqldockerhost
    environment:
      - MYSQL_ROOT_PASSWORD=supersecret
      - MYSQL_DATABASE=sensordatabase
      - MYSQL_USER=master
      - MYSQL_PASSWORD=supersecret
    command: ['--default-authentication-plugin=mysql_native_password']
    ports:
      - 3311:3306
    volumes:
      - ../docker_volumes/mydb:/var/lib/mysql
    networks:
      - isolated

  metabase:
    image: metabase/metabase:latest
    container_name: Metabase
    depends_on:
      - mysql
    links:
      - mysql
    volumes:
      - ../docker_volumes/metabase:/metabase
      - ../docker_volumes/metabase-data:/metabase-data
    environment:
      - MB_DB_FILE=/metabase-data/metabase.db
    restart: always
    ports:
      - 3001:3000

    networks:
      - isolated

  mosquitto:
    image: eclipse-mosquitto:latest
    container_name: Mosquitto
    hostname: mymosquittohost
    volumes:
     - ../docker_volumes/mosquitto/config:/mosquitto/config
     - ../docker_volumes/mosquitto/data:/mosquitto/data
     - ../docker_volumes/mosquitto/log:/mosquitto/log
    restart: always
    ports:
      - 1884:1883
    networks:
      - isolated

  nodered:
    image: nodered/node-red:latest
    container_name: Nodered
    volumes:
      - ../docker_volumes/nodered/data:/data
    restart: always
    ports:
```

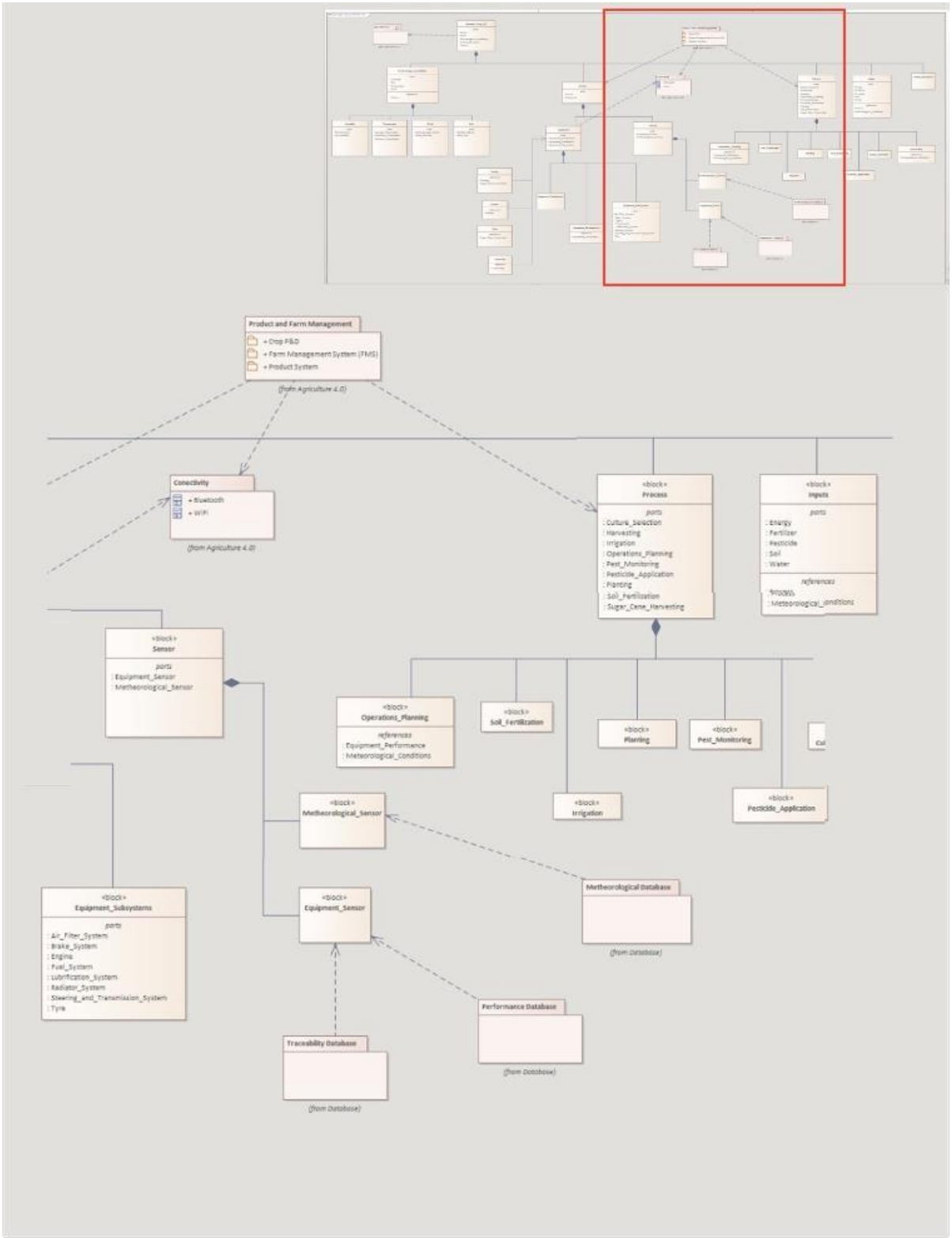
```
      - 1881:1880
    networks:
      - isolated

  jupyter:
    image: jupyter/scipy-notebook:latest
    container_name: Jupyter
    hostname: myjupyterhost
    container_name: jupyter_lab
    build: .
    command:
      - /bin/bash
      - -c
      - start-notebook.sh --NotebookApp.token='' --NotebookApp.password=''
      - pip3 install pymysql
    volumes:
    - ../docker_volumes/jupyter/notebooks:/home/jovyan/notebooks
    environment:
      - JUPYTER_ENABLE_LAB=yes
      - GRANT_SUDO=yes
    ports:
      - 8889:8888
    networks:
      - isolated

networks:
  isolated:
    driver: bridge
```

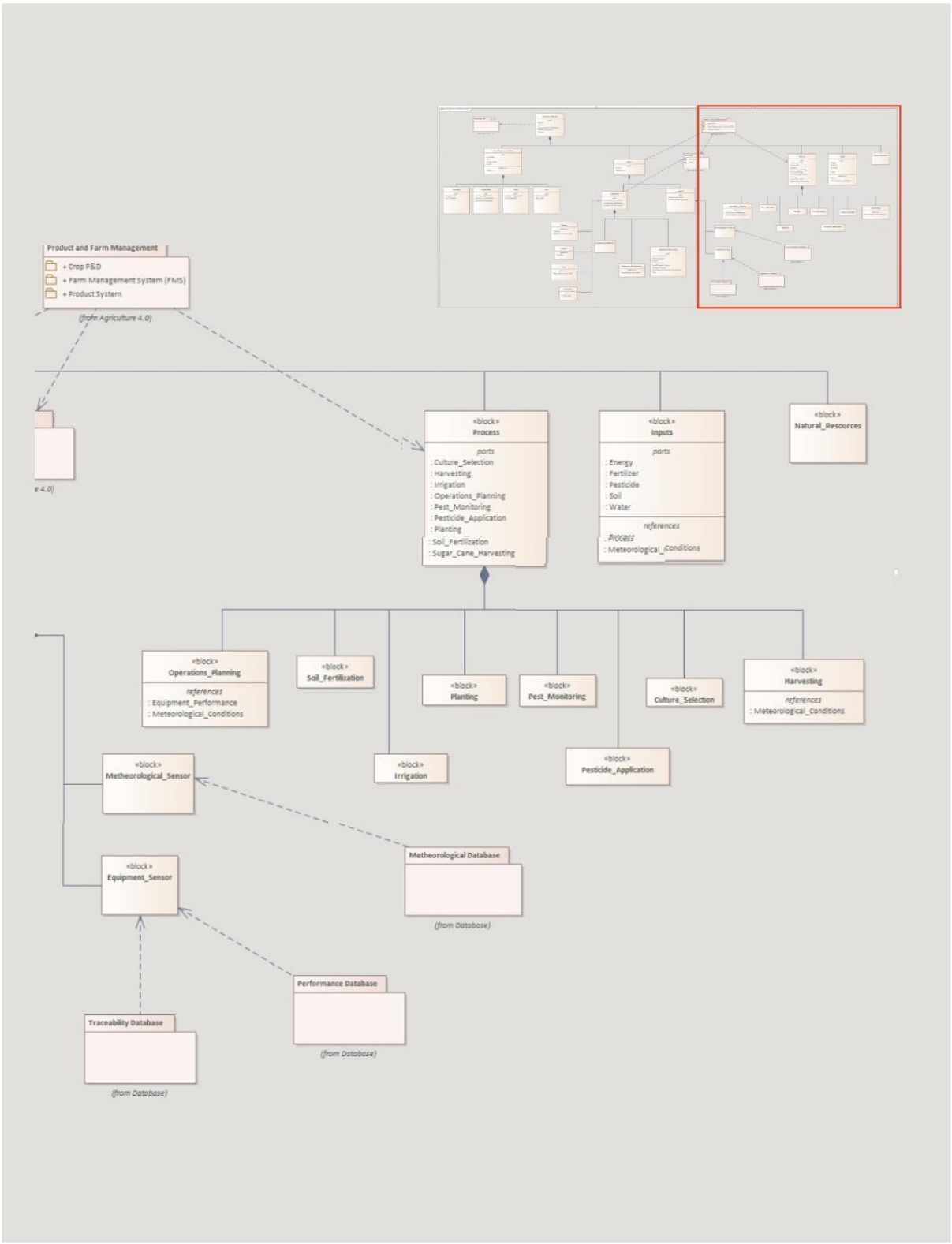
APÊNDICE B – “nós” utilizados e suas respectivas funções

Nó	Função
injection	acionar o início do fluxo
read file	ler arquivos csv
csv	gerar strings no formato csv
simple queue	limita as mensagens que são processadas em um determinado período de tempo
function	permitir que um código JavaScript seja executado nas mensagens que são passadas pelo nó
mqtt out	publicar mensagem em um tópico MQTT
debug	mostrar as mensagens no painel lateral
mqtt in	assinar uma mensagem em um tópico MQTT
write file	escrever arquivo
template	gerar texto utilizando propriedades de uma mensagem para preencher um modelo
http request	enviar um pedido e receber uma resposta de um endereço URL
switch	rotear as mensagens para diferentes ramificações de um fluxo de acordo com regras estabelecidas
json	converter um objeto em uma string JSON
mysql	ler e escrever mensagens em uma base de dados MySQL

APÊNDICE C – Modelo sistêmico de Fazenda 4.0




Product and Farm Management
+ Crop R&D
+ Farm Management System (FMS)
+ Product System
(from Agriculture 4.0)
Conectivity
+ Bluetooth
+ WiFi
(from Agriculture 4.0)
«block»
Process
parts
: Culture_Selection
: Harvesting
: Irrigation
: Operations_Planning
: Pest_Monitoring
: Pesticide_Application
: Planting
: Soil_Fertilization
: Sugar_Cane_Harvesting
«block»
Inputs
parts
: Energy
: Fertilizer
: Pesticide
: Soil
: Water
references
: Process
: Meteorological_conditions
«block»
Sensor
parts
: Equipment_Sensor
: Metheorological_Sensor
«block»
Operations_Planning
references
: Equipment_Performance
: Meteorological_Conditions
«block»
Soil_Fertilization
«block»
Planning
«block»
Pest_Monitoring
«block»
Irrigation
«block»
Pesticide_Application
«block»
Metheorological_Sensor
«block»
Equipment_Subsystems
parts
: Air_Filter_System
: Brake_System
: Engine
: Fuel_System
: Lubrification_System
: Radiator_System
: Steering_and_Trasmission_System
: Tyre
«block»
Equipment_Sensor
Metheorological Database
(from Database)
Performance Database
(from Database)
Traceability Database
(from Database)


Product and Farm Management
+ Crop P&D
+ Farm Management System (FMS)
+ Product System
(from Agriculture 4.0)
«block»
Process
ports
: Culture_Selection
: Harvesting
: Irrigation
: Operations_Planning
: Pest_Monitoring
: Pesticide_Application
: Planting
: Soil_Fertilization
: Sugar_Cane_Harvesting
«block»
Inputs
ports
: Energy
: Fertilizer
: Pesticide
: Soil
: Water
references
: Process
: Meteorological_Conditions
«block»
Natural_Resources
«block»
Operations_Planning
references
: Equipment_Performance
: Meteorological_Conditions
«block»
Soil_Fertilization
«block»
Planting
«block»
Pest_Monitoring
«block»
Culture_Selection
«block»
Harvesting
references
: Meteorological_Conditions
«block»
Irrigation
«block»
Pesticide_Application
«block»
Metheorological_Sensor
«block»
Equipment_Sensor
Metheorological Database
(from Database)
Performance Database
(from Database)
Traceability Database
(from Database)

APÊNDICE D – Dados das planilhas de entrada do Modelo

Nome da Guia (Node-RED)	**Dados da planilha de entrada (csv)**
Harvester01, Harvester02, Harvester03, Harvester04, Harvester05, Harvester06	Time
	Prim_Extr_Rpm
	Prim_Extr_Rpm_Setp
	Basecutter_pressure
	Chopper_Hydr_Press
	Eng_Load
	Basecutter_height
	Cruise_Command
	Ground_Speed
	BaseCutter_Rpm
	Chopper_Rpm
	Machine_Seconds
	Elevator_Seconds
	Fuel_Level_Pct
	Machine_Model
	Machine_ID
Price_AVG	Machine_Model
	Component
	Part_Number
	Price_avg
Quality	Date
	City
	Density
	Particles
Store_stock	Part_Number
	City
	Qtd
	Store
	Contact
Part_Number	Machine_Model
	Component
	Part_Number
Performance	Machine_Model
	Area_hour
	Price_ton
	ton_ha

Nome da Guia (Node-RED)	**Dados da planilha de entrada (csv)**
Weather_Harvester01, Weather_Harvester02, Weather_Harvester03, Weather_Harvester04, Weather_Harvester05, Weather_Harvester06	position_lat
	position_lon
	Machine_ID
Maintenance_Field01, Maintenance_Field02, Maintenance_Field03	Machine_ID
	Machine_Seconds
	Filter_ID
Field01_stock, Field02_stock, Field03_stock	Part_Number
	Qtd
FuelFilter01, FuelFilter02, FuelFilter03, FuelFilter04, FuelFilter05, FuelFilter06	Machine_ID
	Part_Number
	Filter_ID
	Filter_Pressure
Fuel_Information	Machine_Model
	fuel_tank_capacity
	fuel_consuption_avg
	cost_diesel_avg

APÊNDICE E– Nó *function* fluxo de armazenamento de dados

- Harvester01

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Prim_Extr_Rpm;
var tag4 = msg.payload.Prim_Extr_Rpm_Setp;
var tag5 = msg.payload.Basecutter_pressure;
var tag6 = msg.payload.Chopper_Hydr_Press;
var tag7 = msg.payload.Eng_Load;
var tag8 = msg.payload.Basecutter_height;
var tag9 = msg.payload.Cruise_Command;
var tag10 = msg.payload.Ground_Speed;
var tag11 = msg.payload.BaseCutter_Rpm;
var tag12 = msg.payload.Chopper_Rpm;
var tag13 = msg.payload.Machine_Seconds;
var tag14 = msg.payload.Elevator_Seconds;
var tag15 = msg.payload.Fuel_Level_Pct;
var tag16 = msg.payload.Machine_Model;
var tag17 = msg.payload.Machine_ID;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12, tag13, tag14, tag15, tag16, tag17];
msg.topic = 'CREATE TABLE IF NOT EXISTS harvester01 (Date
VARCHAR(50), timestamp BIGINT, Prim_Extr_Rpm INT, Prim_Extr_Rpm_Setp
INT, Basecutter_pressure FLOAT, Chopper_Hydr_Press FLOAT, Eng_Load
INT, Basecutter_height INT, Cruise_Command FLOAT, Ground_Speed FLOAT,
BaseCutter_Rpm INT, Chopper_Rpm INT, Machine_Seconds INT,
Elevator_Seconds INT, Fuel_Level_Pct FLOAT, Machine_Model
VARCHAR(50), Machine_ID VARCHAR(50)); INSERT INTO harvester01 (Date,
Timestamp, Prim_Extr_Rpm, Prim_Extr_Rpm_Setp, Basecutter_pressure,
Chopper_Hydr_Press, Eng_Load, Basecutter_height, Cruise_Command,
Ground_Speed, BaseCutter_Rpm, Chopper_Rpm, Machine_Seconds,
Elevator_Seconds, Fuel_Level_Pct, Machine_Model, Machine_ID) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Harvester02

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Prim_Extr_Rpm;
var tag4 = msg.payload.Prim_Extr_Rpm_Setp;
var tag5 = msg.payload.Basecutter_pressure;
var tag6 = msg.payload.Chopper_Hydr_Press;
var tag7 = msg.payload.Eng_Load;
var tag8 = msg.payload.Basecutter_height;
var tag9 = msg.payload.Cruise_Command;
var tag10 = msg.payload.Ground_Speed;
var tag11 = msg.payload.BaseCutter_Rpm;
var tag12 = msg.payload.Chopper_Rpm;
var tag13 = msg.payload.Machine_Seconds;
var tag14 = msg.payload.Elevator_Seconds;
var tag15 = msg.payload.Fuel_Level_Pct;
var tag16 = msg.payload.Machine_Model;
```

```
var tag17 = msg.payload.Machine_ID;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12, tag13, tag14, tag15, tag16, tag17];
msg.topic = 'CREATE TABLE IF NOT EXISTS harvester02 (Date
VARCHAR(50), timestamp BIGINT, Prim_Extr_Rpm INT, Prim_Extr_Rpm_Setp
INT, Basecutter_pressure FLOAT, Chopper_Hydr_Press FLOAT, Eng_Load
INT, Basecutter_height INT, Cruise_Command FLOAT, Ground_Speed FLOAT,
BaseCutter_Rpm INT, Chopper_Rpm INT, Machine_Seconds INT,
Elevator_Seconds INT, Fuel_Level_Pct FLOAT, Machine_Model
VARCHAR(50), Machine_ID VARCHAR(50)); INSERT INTO harvester02 (Date,
Timestamp, Prim_Extr_Rpm, Prim_Extr_Rpm_Setp, Basecutter_pressure,
Chopper_Hydr_Press, Eng_Load, Basecutter_height, Cruise_Command,
Ground_Speed, BaseCutter_Rpm, Chopper_Rpm, Machine_Seconds,
Elevator_Seconds, Fuel_Level_Pct, Machine_Model, Machine_ID) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Harvester03

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Prim_Extr_Rpm;
var tag4 = msg.payload.Prim_Extr_Rpm_Setp;
var tag5 = msg.payload.Basecutter_pressure;
var tag6 = msg.payload.Chopper_Hydr_Press;
var tag7 = msg.payload.Eng_Load;
var tag8 = msg.payload.Basecutter_height;
var tag9 = msg.payload.Cruise_Command;
var tag10 = msg.payload.Ground_Speed;
var tag11 = msg.payload.BaseCutter_Rpm;
var tag12 = msg.payload.Chopper_Rpm;
var tag13 = msg.payload.Machine_Seconds;
var tag14 = msg.payload.Elevator_Seconds;
var tag15 = msg.payload.Fuel_Level_Pct;
var tag16 = msg.payload.Machine_Model;
var tag17 = msg.payload.Machine_ID;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12, tag13, tag14, tag15, tag16, tag17];
msg.topic = 'CREATE TABLE IF NOT EXISTS harvester03 (Date
VARCHAR(50), timestamp BIGINT, Prim_Extr_Rpm INT, Prim_Extr_Rpm_Setp
INT, Basecutter_pressure FLOAT, Chopper_Hydr_Press FLOAT, Eng_Load
INT, Basecutter_height INT, Cruise_Command FLOAT, Ground_Speed FLOAT,
BaseCutter_Rpm INT, Chopper_Rpm INT, Machine_Seconds INT,
Elevator_Seconds INT, Fuel_Level_Pct FLOAT, Machine_Model
VARCHAR(50), Machine_ID VARCHAR(50)); INSERT INTO harvester03 (Date,
Timestamp, Prim_Extr_Rpm, Prim_Extr_Rpm_Setp, Basecutter_pressure,
Chopper_Hydr_Press, Eng_Load, Basecutter_height, Cruise_Command,
Ground_Speed, BaseCutter_Rpm, Chopper_Rpm, Machine_Seconds,
Elevator_Seconds, Fuel_Level_Pct, Machine_Model, Machine_ID) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Harvester04

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Prim_Extr_Rpm;
var tag4 = msg.payload.Prim_Extr_Rpm_Setp;
var tag5 = msg.payload.Basecutter_pressure;
var tag6 = msg.payload.Chopper_Hydr_Press;
var tag7 = msg.payload.Eng_Load;
var tag8 = msg.payload.Basecutter_height;
var tag9 = msg.payload.Cruise_Command;
var tag10 = msg.payload.Ground_Speed;
var tag11 = msg.payload.BaseCutter_Rpm;
var tag12 = msg.payload.Chopper_Rpm;
var tag13 = msg.payload.Machine_Seconds;
var tag14 = msg.payload.Elevator_Seconds;
var tag15 = msg.payload.Fuel_Level_Pct;
var tag16 = msg.payload.Machine_Model;
var tag17 = msg.payload.Machine_ID;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12, tag13, tag14, tag15, tag16, tag17];
msg.topic = 'CREATE TABLE IF NOT EXISTS harvester04 (Date
VARCHAR(50), timestamp BIGINT, Prim_Extr_Rpm INT, Prim_Extr_Rpm_Setp
INT, Basecutter_pressure FLOAT, Chopper_Hydr_Press FLOAT, Eng_Load
INT, Basecutter_height INT, Cruise_Command FLOAT, Ground_Speed FLOAT,
BaseCutter_Rpm INT, Chopper_Rpm INT, Machine_Seconds INT,
Elevator_Seconds INT, Fuel_Level_Pct FLOAT, Machine_Model
VARCHAR(50), Machine_ID VARCHAR(50)); INSERT INTO harvester04 (Date,
Timestamp, Prim_Extr_Rpm, Prim_Extr_Rpm_Setp, Basecutter_pressure,
Chopper_Hydr_Press, Eng_Load, Basecutter_height, Cruise_Command,
Ground_Speed, BaseCutter_Rpm, Chopper_Rpm, Machine_Seconds,
Elevator_Seconds, Fuel_Level_Pct, Machine_Model, Machine_ID) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Harvester05

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Prim_Extr_Rpm;
var tag4 = msg.payload.Prim_Extr_Rpm_Setp;
var tag5 = msg.payload.Basecutter_pressure;
var tag6 = msg.payload.Chopper_Hydr_Press;
var tag7 = msg.payload.Eng_Load;
var tag8 = msg.payload.Basecutter_height;
var tag9 = msg.payload.Cruise_Command;
var tag10 = msg.payload.Ground_Speed;
var tag11 = msg.payload.BaseCutter_Rpm;
var tag12 = msg.payload.Chopper_Rpm;
var tag13 = msg.payload.Machine_Seconds;
var tag14 = msg.payload.Elevator_Seconds;
var tag15 = msg.payload.Fuel_Level_Pct;
var tag16 = msg.payload.Machine_Model;
var tag17 = msg.payload.Machine_ID;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12, tag13, tag14, tag15, tag16, tag17];
msg.topic = 'CREATE TABLE IF NOT EXISTS harvester05 (Date
VARCHAR(50), timestamp BIGINT, Prim_Extr_Rpm INT, Prim_Extr_Rpm_Setp
INT, Basecutter_pressure FLOAT, Chopper_Hydr_Press FLOAT, Eng_Load
```

```
INT, Basecutter_height INT, Cruise_Command FLOAT, Ground_Speed FLOAT,
BaseCutter_Rpm INT, Chopper_Rpm INT, Machine_Seconds INT,
Elevator_Seconds INT, Fuel_Level_Pct FLOAT, Machine_Model
VARCHAR(50), Machine_ID VARCHAR(50)); INSERT INTO harvester05 (Date,
Timestamp, Prim_Extr_Rpm, Prim_Extr_Rpm_Setp, Basecutter_pressure,
Chopper_Hydr_Press, Eng_Load, Basecutter_height, Cruise_Command,
Ground_Speed, BaseCutter_Rpm, Chopper_Rpm, Machine_Seconds,
Elevator_Seconds, Fuel_Level_Pct, Machine_Model, Machine_ID) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Harvester06

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Prim_Extr_Rpm;
var tag4 = msg.payload.Prim_Extr_Rpm_Setp;
var tag5 = msg.payload.Basecutter_pressure;
var tag6 = msg.payload.Chopper_Hydr_Press;
var tag7 = msg.payload.Eng_Load;
var tag8 = msg.payload.Basecutter_height;
var tag9 = msg.payload.Cruise_Command;
var tag10 = msg.payload.Ground_Speed;
var tag11 = msg.payload.BaseCutter_Rpm;
var tag12 = msg.payload.Chopper_Rpm;
var tag13 = msg.payload.Machine_Seconds;
var tag14 = msg.payload.Elevator_Seconds;
var tag15 = msg.payload.Fuel_Level_Pct;
var tag16 = msg.payload.Machine_Model;
var tag17 = msg.payload.Machine_ID;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12, tag13, tag14, tag15, tag16, tag17];
msg.topic = 'CREATE TABLE IF NOT EXISTS harvester06 (Date
VARCHAR(50), timestamp BIGINT, Prim_Extr_Rpm INT, Prim_Extr_Rpm_Setp
INT, Basecutter_pressure FLOAT, Chopper_Hydr_Press FLOAT, Eng_Load
INT, Basecutter_height INT, Cruise_Command FLOAT, Ground_Speed FLOAT,
BaseCutter_Rpm INT, Chopper_Rpm INT, Machine_Seconds INT,
Elevator_Seconds INT, Fuel_Level_Pct FLOAT, Machine_Model
VARCHAR(50), Machine_ID VARCHAR(50)); INSERT INTO harvester06 (Date,
Timestamp, Prim_Extr_Rpm, Prim_Extr_Rpm_Setp, Basecutter_pressure,
Chopper_Hydr_Press, Eng_Load, Basecutter_height, Cruise_Command,
Ground_Speed, BaseCutter_Rpm, Chopper_Rpm, Machine_Seconds,
Elevator_Seconds, Fuel_Level_Pct, Machine_Model, Machine_ID) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Weatherharvester01

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Longitude;
var tag5 = msg.payload.Latitude;
var tag6 = msg.payload.City_name;
```

```
var tag7 = msg.payload.Temperature;
var tag8 = msg.payload.Temperature_min;
var tag9 = msg.payload.Temperature_max;
var tag10 = msg.payload.Pressure;
var tag11 = msg.payload.Humidity;
var tag12 = msg.payload.Wind_Speed;
msg.payload = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12];
msg.topic = 'CREATE TABLE IF NOT EXISTS weatherharvester01 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50), Longitude
FLOAT, Latitude FLOAT, City_name VARCHAR(50), Temperature FLOAT,
Temperature_min FLOAT, Temperature_max FLOAT, Pressure INT, Humidity
INT, Wind_Speed FLOAT); INSERT INTO weatherharvester01 (Date,
Timestamp, Machine_ID, Longitude, Latitude, City_Name, Temperature,
Temperature_min, Temperature_max, Pressure, Humidity, Wind_Speed)
VALUES (?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Weatherharvester02

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Longitude;
var tag5 = msg.payload.Latitude;
var tag6 = msg.payload.City_name;
var tag7 = msg.payload.Temperature;
var tag8 = msg.payload.Temperature_min;
var tag9 = msg.payload.Temperature_max;
var tag10 = msg.payload.Pressure;
var tag11 = msg.payload.Humidity;
var tag12 = msg.payload.Wind_Speed;
msg.payload = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12];
msg.topic = 'CREATE TABLE IF NOT EXISTS weatherharvester02 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50), Longitude
FLOAT, Latitude FLOAT, City_name VARCHAR(50), Temperature FLOAT,
Temperature_min FLOAT, Temperature_max FLOAT, Pressure INT, Humidity
INT, Wind_Speed FLOAT); INSERT INTO weatherharvester02 (Date,
Timestamp, Machine_ID, Longitude, Latitude, City_Name, Temperature,
Temperature_min, Temperature_max, Pressure, Humidity, Wind_Speed)
VALUES (?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Weatherharvester03

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Longitude;
var tag5 = msg.payload.Latitude;
var tag6 = msg.payload.City_name;
var tag7 = msg.payload.Temperature;
var tag8 = msg.payload.Temperature_min;
var tag9 = msg.payload.Temperature_max;
var tag10 = msg.payload.Pressure;
```

```
var tag11 = msg.payload.Humidity;
var tag12 = msg.payload.Wind_Speed;
msg.payload = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12];
msg.topic = 'CREATE TABLE IF NOT EXISTS weatherharvester03 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50), Longitude
FLOAT, Latitude FLOAT, City_name VARCHAR(50), Temperature FLOAT,
Temperature_min FLOAT, Temperature_max FLOAT, Pressure INT, Humidity
INT, Wind_Speed FLOAT); INSERT INTO weatherharvester03 (Date,
Timestamp, Machine_ID, Longitude, Latitude, City_Name, Temperature,
Temperature_min, Temperature_max, Pressure, Humidity, Wind_Speed)
VALUES (?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Weatherharvester04

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Longitude;
var tag5 = msg.payload.Latitude;
var tag6 = msg.payload.City_name;
var tag7 = msg.payload.Temperature;
var tag8 = msg.payload.Temperature_min;
var tag9 = msg.payload.Temperature_max;
var tag10 = msg.payload.Pressure;
var tag11 = msg.payload.Humidity;
var tag12 = msg.payload.Wind_Speed;
msg.payload = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12];
msg.topic = 'CREATE TABLE IF NOT EXISTS weatherharvester04 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50), Longitude
FLOAT, Latitude FLOAT, City_name VARCHAR(50), Temperature FLOAT,
Temperature_min FLOAT, Temperature_max FLOAT, Pressure INT, Humidity
INT, Wind_Speed FLOAT); INSERT INTO weatherharvester04 (Date,
Timestamp, Machine_ID, Longitude, Latitude, City_Name, Temperature,
Temperature_min, Temperature_max, Pressure, Humidity, Wind_Speed)
VALUES (?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Weatherharvester05

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Longitude;
var tag5 = msg.payload.Latitude;
var tag6 = msg.payload.City_name;
var tag7 = msg.payload.Temperature;
var tag8 = msg.payload.Temperature_min;
var tag9 = msg.payload.Temperature_max;
var tag10 = msg.payload.Pressure;
var tag11 = msg.payload.Humidity;
var tag12 = msg.payload.Wind_Speed;
```

```
msg.payload = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12];
msg.topic = 'CREATE TABLE IF NOT EXISTS weatherharvester05 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50), Longitude
FLOAT, Latitude FLOAT, City_name VARCHAR(50), Temperature FLOAT,
Temperature_min FLOAT, Temperature_max FLOAT, Pressure INT, Humidity
INT, Wind_Speed FLOAT); INSERT INTO weatherharvester05 (Date,
Timestamp, Machine_ID, Longitude, Latitude, City_Name, Temperature,
Temperature_min, Temperature_max, Pressure, Humidity, Wind_Speed)
VALUES (?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Weatherharvester06

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Longitude;
var tag5 = msg.payload.Latitude;
var tag6 = msg.payload.City_name;
var tag7 = msg.payload.Temperature;
var tag8 = msg.payload.Temperature_min;
var tag9 = msg.payload.Temperature_max;
var tag10 = msg.payload.Pressure;
var tag11 = msg.payload.Humidity;
var tag12 = msg.payload.Wind_Speed;
msg.payload = [tag1, tag2, tag3, tag4, tag5, tag6, tag7, tag8, tag9,
tag10, tag11, tag12];
msg.topic = 'CREATE TABLE IF NOT EXISTS weatherharvester06 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50), Longitude
FLOAT, Latitude FLOAT, City_name VARCHAR(50), Temperature FLOAT,
Temperature_min FLOAT, Temperature_max FLOAT, Pressure INT, Humidity
INT, Wind_Speed FLOAT); INSERT INTO weatherharvester06 (Date,
Timestamp, Machine_ID, Longitude, Latitude, City_Name, Temperature,
Temperature_min, Temperature_max, Pressure, Humidity, Wind_Speed)
VALUES (?,?,?,?,?,?,?,?,?,?,?,?)';
return msg;
```

- Maintenance

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Machine_Seconds;
var tag5 = msg.payload.Filter_ID;
msg.payload  = [tag1, tag2, tag3, tag4, tag5];
msg.topic = 'CREATE TABLE IF NOT EXISTS maintenance (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50),
Machine_Seconds INT, Filter_ID VARCHAR(50)); INSERT INTO maintenance
(Date, Timestamp, Machine_ID, Machine_Seconds, Filter_ID) VALUES
(?,?,?,?,?)';
return msg;
```

- Price_AVG

```
var tag1 = msg.payload.Machine_Model;
```

```
var tag2 = msg.payload.Component;
var tag3 = msg.payload.Part_Number;
var tag4 = msg.payload.Price_avg;
msg.payload  = [tag1, tag2, tag3, tag4];
msg.topic = 'CREATE TABLE IF NOT EXISTS price_avg (Machine_Model
VARCHAR(50), Component VARCHAR(50), Part_Number VARCHAR(50),
Price_avg INT); INSERT INTO price_avg (Machine_Model, Component,
Part_Number, Price_avg) VALUES (?,?,?,?)';
return msg;
```

- Performance

```
var tag1 = msg.payload.Machine_Model;
var tag2 = msg.payload.Area_hour;
var tag3 = msg.payload.Price_ton;
var tag4 = msg.payload.ton_ha;
msg.payload  = [tag1, tag2, tag3, tag4];
msg.topic = 'CREATE TABLE IF NOT EXISTS performance (Machine_Model
VARCHAR(50), Area_hour FLOAT, Price_ton FLOAT, ton_ha FLOAT); INSERT
INTO performance (Machine_Model, Area_hour, Price_ton, ton_ha) VALUES
(?,?,?,?)';
return msg;
```

- Fuel_information

```
var tag1 = msg.payload.Machine_Model;
var tag2 = msg.payload.fuel_tank_capacity;
var tag3 = msg.payload.fuel_consuption_avg;
var tag4 = msg.payload.Cost_Diesel_avg;
msg.payload  = [tag1, tag2, tag3, tag4];
msg.topic = 'CREATE TABLE IF NOT EXISTS Fuel_information
(Machine_Model VARCHAR(50), fuel_tank_capacity INT,
fuel_consuption_avg FLOAT, Cost_Diesel_avg FLOAT); INSERT INTO
Fuel_information (Machine_Model, fuel_tank_capacity,
fuel_consuption_avg, Cost_Diesel_avg) VALUES (?,?,?,?)';
return msg;
```

- FuelFilter01

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Filter_PartNumber;
var tag5 = msg.payload.Filter_ID;
var tag6 = msg.payload.Filter_Pressure;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6];
msg.topic = 'CREATE TABLE IF NOT EXISTS fuelfilter01 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50),
Filter_PartNumber VARCHAR(50), Filter_ID VARCHAR(50), Filter_Pressure
INT); INSERT INTO fuelfilter01 (Date, Timestamp, Machine_ID,
Filter_PartNumber, Filter_ID, Filter_Pressure) VALUES (?,?,?,?,?,?)';
return msg;
```

- FuelFilter02

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Filter_PartNumber;
var tag5 = msg.payload.Filter_ID;
var tag6 = msg.payload.Filter_Pressure;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6];
msg.topic = 'CREATE TABLE IF NOT EXISTS fuelfilter02 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50),
Filter_PartNumber VARCHAR(50), Filter_ID VARCHAR(50), Filter_Pressure
INT); INSERT INTO fuelfilter02 (Date, Timestamp, Machine_ID,
Filter_PartNumber, Filter_ID, Filter_Pressure) VALUES (?,?,?,?,?,?)';
return msg;
```

- FuelFilter03

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Filter_PartNumber;
var tag5 = msg.payload.Filter_ID;
var tag6 = msg.payload.Filter_Pressure;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6];
msg.topic = 'CREATE TABLE IF NOT EXISTS fuelfilter03 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50),
Filter_PartNumber VARCHAR(50), Filter_ID VARCHAR(50), Filter_Pressure
INT); INSERT INTO fuelfilter03 (Date, Timestamp, Machine_ID,
Filter_PartNumber, Filter_ID, Filter_Pressure) VALUES (?,?,?,?,?,?)';
return msg;
```

- FuelFilter04

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Filter_PartNumber;
var tag5 = msg.payload.Filter_ID;
var tag6 = msg.payload.Filter_Pressure;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6];
msg.topic = 'CREATE TABLE IF NOT EXISTS fuelfilter04 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50),
Filter_PartNumber VARCHAR(50), Filter_ID VARCHAR(50), Filter_Pressure
INT); INSERT INTO fuelfilter04 (Date, Timestamp, Machine_ID,
Filter_PartNumber, Filter_ID, Filter_Pressure) VALUES (?,?,?,?,?,?)';
return msg;
```

- FuelFilter05

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Filter_PartNumber;
var tag5 = msg.payload.Filter_ID;
var tag6 = msg.payload.Filter_Pressure;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6];
```

```
msg.topic = 'CREATE TABLE IF NOT EXISTS fuelfilter05 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50),
Filter_PartNumber VARCHAR(50), Filter_ID VARCHAR(50), Filter_Pressure
INT); INSERT INTO fuelfilter05 (Date, Timestamp, Machine_ID,
Filter_PartNumber, Filter_ID, Filter_Pressure) VALUES (?,?,?,?,?,?)';
return msg;
```

- FuelFilter06

```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.Timestamp;
var tag3 = msg.payload.Machine_ID;
var tag4 = msg.payload.Filter_PartNumber;
var tag5 = msg.payload.Filter_ID;
var tag6 = msg.payload.Filter_Pressure;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6];
msg.topic = 'CREATE TABLE IF NOT EXISTS fuelfilter06 (Date
VARCHAR(50), timestamp BIGINT, Machine_ID VARCHAR(50),
Filter_PartNumber VARCHAR(50), Filter_ID VARCHAR(50), Filter_Pressure
INT); INSERT INTO fuelfilter06 (Date, Timestamp, Machine_ID,
Filter_PartNumber, Filter_ID, Filter_Pressure) VALUES (?,?,?,?,?,?)';
return msg;
```

- PartNumber

```
var tag1 = msg.payload.Machine_Model;
var tag2 = msg.payload.Component;
var tag3 = msg.payload.Part_Number;
msg.payload  = [tag1, tag2, tag3];
msg.topic = 'CREATE TABLE IF NOT EXISTS partnumbers (Machine_Model
VARCHAR(50), Component VARCHAR(50), Part_Number VARCHAR(50)); INSERT
INTO partnumbers (Machine_Model, Component, Part_Number) VALUES
(?,?,?)';
return msg;
```

- StoreStocks

```
var tag1 = msg.payload.Part_Number;
var tag2 = msg.payload.City;
var tag3 = msg.payload.Longitude;
var tag4 = msg.payload.Latitude;
var tag5 = msg.payload.Qtd;
var tag6 = msg.payload.Store;
var tag7 = msg.payload.Contact;
msg.payload  = [tag1, tag2, tag3, tag4, tag5, tag6, tag7];
msg.topic = 'CREATE TABLE IF NOT EXISTS storesstocks (Part_Number
VARCHAR(50), City VARCHAR(50), Longitude FLOAT, Latitude FLOAT, Qtd
INT, Store VARCHAR(50), Contact VARCHAR(50)); INSERT INTO
storesstocks (Part_Number, City, Longitude, Latitude, Qtd, Store,
Contact) VALUES (?,?,?,?,?,?,?)';
return msg;
```

- FieldStock01

```
var tag1 = msg.payload.Part_Number;
var tag2 = msg.payload.Qtd;
msg.payload  = [tag1, tag2];
msg.topic = 'CREATE TABLE IF NOT EXISTS field01stocks (Part_Number
VARCHAR(50), Qtd INT); INSERT INTO field01stocks (Part_Number, Qtd)
VALUES (?,?)';
return msg;
```

- FieldStock02

```
var tag1 = msg.payload.Part_Number;
var tag2 = msg.payload.Qtd;
msg.payload  = [tag1, tag2];
msg.topic = 'CREATE TABLE IF NOT EXISTS field02stocks (Part_Number
VARCHAR(50), Qtd INT); INSERT INTO field02stocks (Part_Number, Qtd)
VALUES (?,?)';
return msg;
```

- FieldStock03

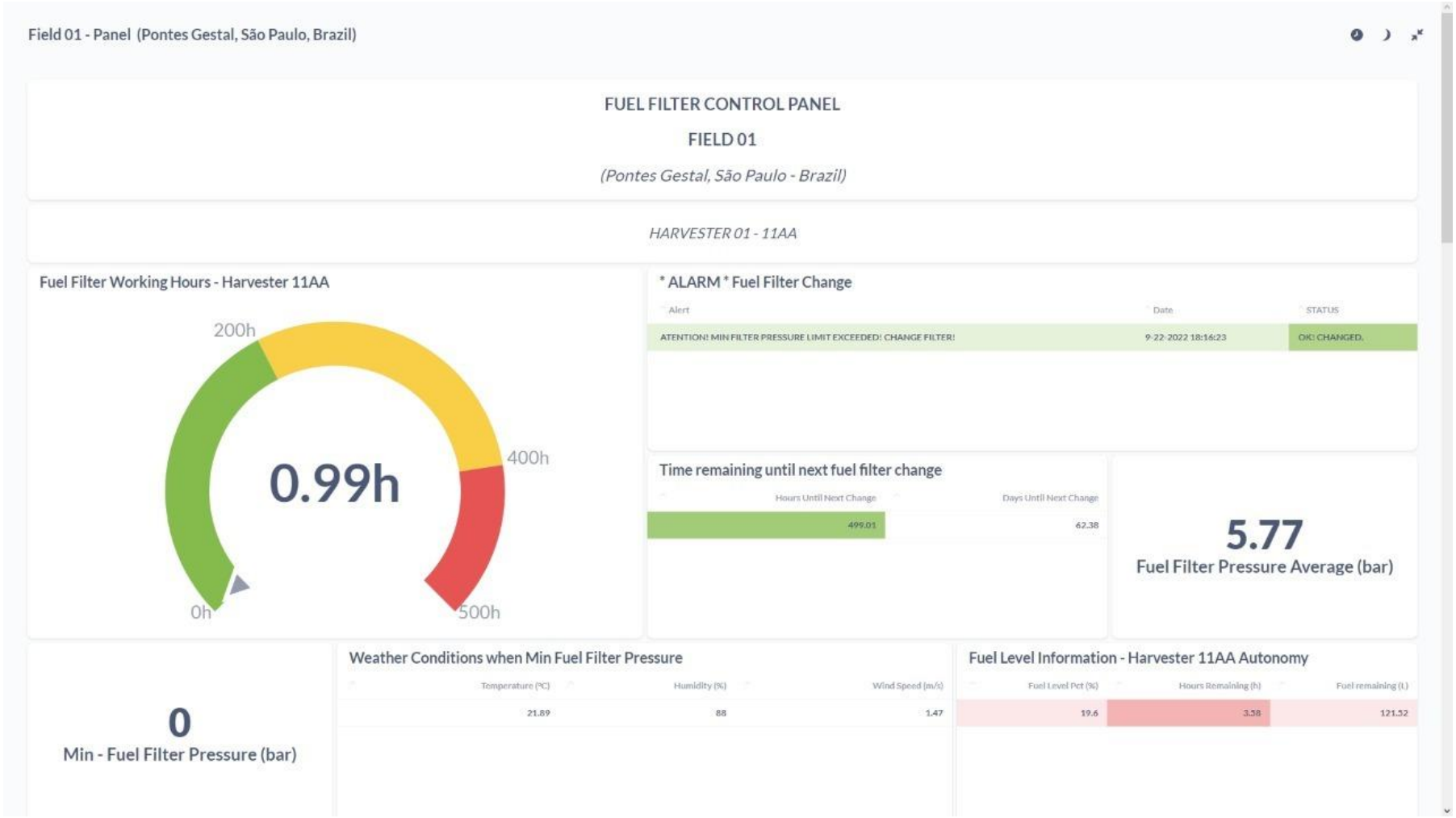
```
var tag1 = msg.payload.Part_Number;
var tag2 = msg.payload.Qtd;
msg.payload  = [tag1, tag2];
msg.topic = 'CREATE TABLE IF NOT EXISTS field03stocks (Part_Number
VARCHAR(50), Qtd INT); INSERT INTO field03stocks (Part_Number, Qtd)
VALUES (?,?)';
return msg;
```

- Quality

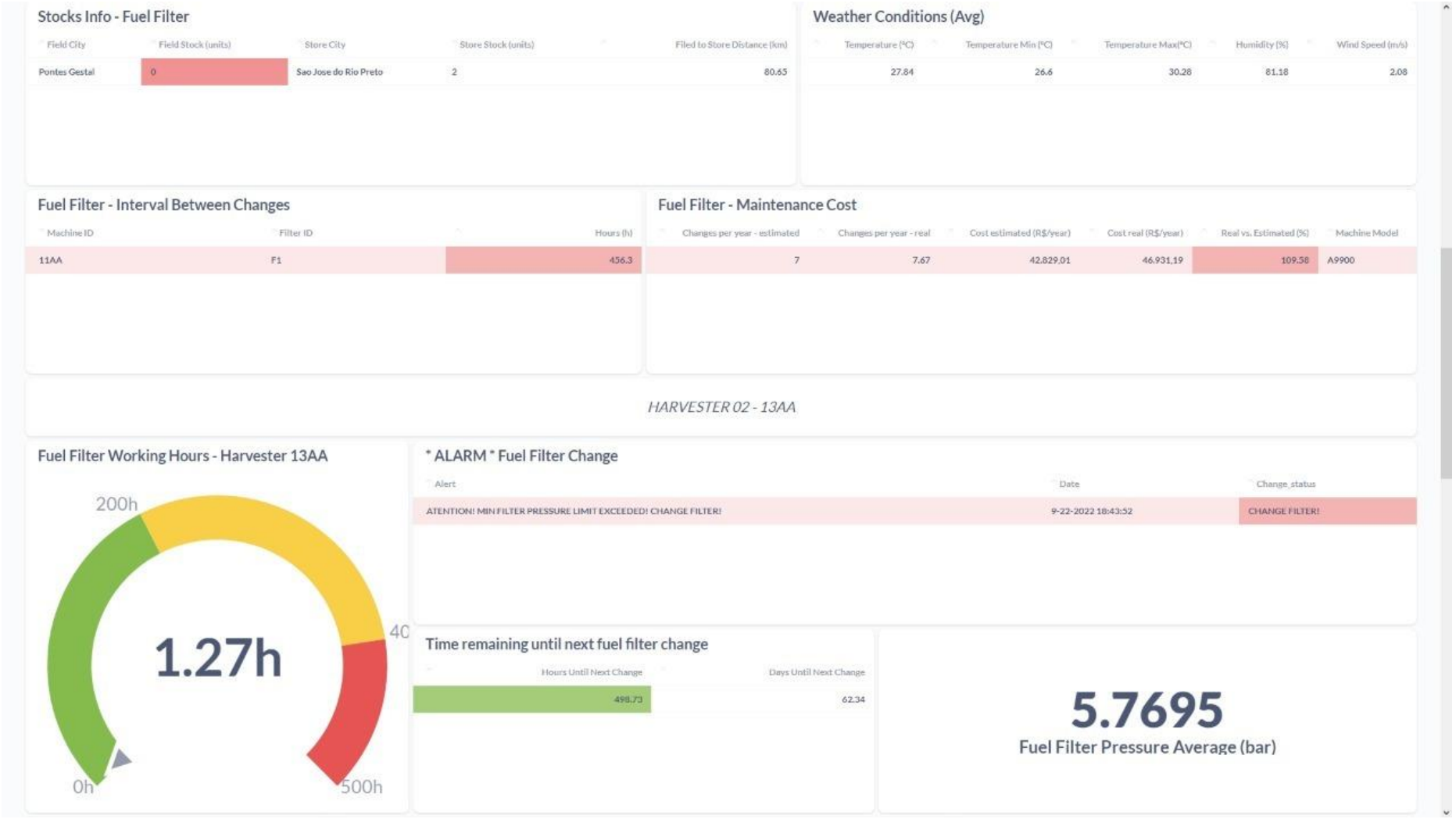
```
var tag1 = msg.payload.Date;
var tag2 = msg.payload.City;
var tag3 = msg.payload.Density;
var tag4 = msg.payload.Particles;
msg.payload  = [tag1, tag2, tag3, tag4];
msg.topic = 'CREATE TABLE IF NOT EXISTS quality (Date VARCHAR(50),
City VARCHAR(50), Density INT, Particles INT); INSERT INTO quality
(Date, City, Density, Particles) VALUES (?,?,?,?)';
return msg;
```

APÊNDICE F – Modelo de informação – *dashboard*

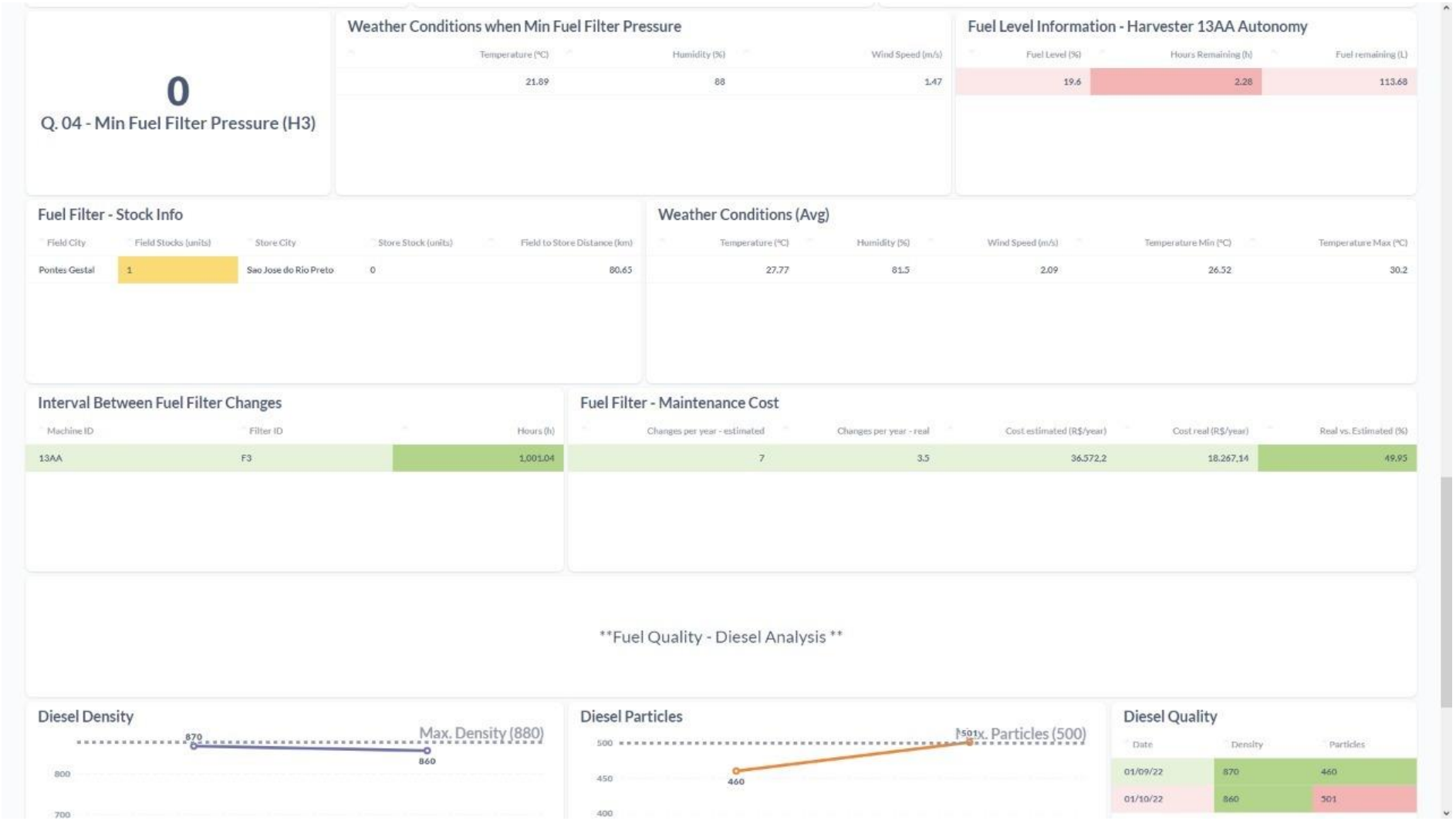
- Parte I



- Parte II


Stocks Info - Fuel Filter
Field City
Field Stock (units)
Store City
Store Stock (units)
Filed to Store Distance (km)
Pontes Gestal
0
Sao Jose do Rio Preto
2
80.65
Weather Conditions (Avg)
Temperature (°C)
Temperature Min (°C)
Temperature Max(°C)
Humidity (%)
Wind Speed (m/s)
27.84
26.6
30.28
81.18
2.08
Fuel Filter - Interval Between Changes
Machine ID
Filter ID
Hours (h)
11AA
F1
456.3
Fuel Filter - Maintenance Cost
Changes per year - estimated
Changes per year - real
Cost estimated (R$/year)
Cost real (R$/year)
Real vs. Estimated (%)
Machine Model
7
7.67
42.829,01
46.931,19
109.58
A9900
HARVESTER 02 - 13AA
Fuel Filter Working Hours - Harvester 13AA
200h
1.27h
0h
500h
* ALARM * Fuel Filter Change
Alert
Date
Change_status
ATENTION! MIN FILTER PRESSURE LIMIT EXCEEDED! CHANGE FILTER!
9-22-2022 18:43:52
CHANGE FILTER!
Time remaining until next fuel filter change
Hours Until Next Change
Days Until Next Change
498.73
62.34
5.7695
Fuel Filter Pressure Average (bar)

- Parte III


0
Q. 04 - Min Fuel Filter Pressure (H3)
Weather Conditions when Min Fuel Filter Pressure
Temperature (°C) Humidity (%) Wind Speed (m/s)
21.89 88 1.47
Fuel Level Information - Harvester 13AA Autonomy
Fuel Level (%) Hours Remaining (h) Fuel remaining (L)
19.6 2.28 113.68
Fuel Filter - Stock Info
Field City Field Stocks (units) Store City Store Stock (units) Field to Store Distance (km)
Pontes Gestal 1 Sao Jose do Rio Preto 0 80.65
Weather Conditions (Avg)
Temperature (°C) Humidity (%) Wind Speed (m/s) Temperature Min (°C) Temperature Max (°C)
27.77 81.5 2.09 26.52 30.2
Interval Between Fuel Filter Changes
Machine ID Filter ID Hours (h)
13AA F3 1,001.04
Fuel Filter - Maintenance Cost
Changes per year - estimated Changes per year - real Cost estimated (R$/year) Cost real (R$/year) Real vs. Estimated (%)
7 3.5 36.572,2 18.267,14 49,95
**Fuel Quality - Diesel Analysis **
Diesel Density
Max. Density (880)
870
860
800
700
Diesel Particles
Max. Particles (500)
501
460
500
450
400
Diesel Quality
Date Density Particles
01/09/22 870 460
01/10/22 860 501

- Parte IV


Interval Between Fuel Filter Changes
Machine ID
Filter ID
Hours (h)
13AA
F3
1,001.04
Fuel Filter - Maintenance Cost
Changes per year - estimated
Changes per year - real
Cost estimated (R$/year)
Cost real (R$/year)
Real vs. Estimated (%)
7
3.5
36.572,2
18.267,14
49.95
**Fuel Quality - Diesel Analysis **
Diesel Density
Max. Density (880)
870
860
Density (kg/m³)
Date
01/09/22
01/10/22
Diesel Particles
Max. Particles (500)
501
460
Particles (ppm)
Date
Diesel Quality
Date
Density
Particles
01/09/22
870
460
01/10/22
860
501
General Panel (click here to return)
Cuiabá
Brasília
Goiânia
Uberlândia
Belo Horizonte
Ribeirão Preto
São José do Rio Preto
Londrina
São Paulo
Rio de Janeiro
Curitiba

APÊNDICE G – Modelo de informação – *machine learning*

```
import sqlalchemy

import pandas as pd

import mysql.connector

import pymysql

import matplotlib.pyplot as plt

%matplotlib inline

import numpy as np

import seaborn as sns


mydb = mysql.connector.connect(host='mysqldockerhost',database='sensordatabase',user='master',password='supersecret', use_pure=True)


mycursor=mydb.cursor()


mycursor.execute('SHOW TABLES')

for x in mycursor:

    print(x)


mycursor.execute('SELECT * FROM harvester01')


harvester01db = pd.DataFrame(data=mycursor, columns = ['Date','Timestamp','Prim_Extr_Rpm','Prim_Extr_Rpm_Setp','Basecutter_pressure','Chopper_Hydr_Press','Eng_Load','Basecutter_height','Cruise_Command','Ground_Speed','BaseCutter_Rpm','Chopper_Rpm','Machine_Seconds','Elevator_Seconds','Fuel_Level_Pct','Machine_Model','Machine_ID'])


mycursor.execute('SELECT * FROM harvester02')


harvester02db = pd.DataFrame(data=mycursor, columns = ['Date','Timestamp','Prim_Extr_Rpm','Prim_Extr_Rpm_Setp','Basecutter_pressure','Chopper_Hydr_Press','Eng_Load','Basecutter_height','Cruise_Command','Ground_Speed','BaseCutter_Rpm','Chopper_Rpm','Machine_Seconds','Elevator_Seconds','Fuel_Level_Pct','Machine_Model','Machine_ID'])
```

```
mycursor.execute('SELECT * FROM harvester03')


harvester03db      =      pd.DataFrame(data=mycursor,      columns      =
['Date','Timestamp','Prim_Extr_Rpm','Prim_Extr_Rpm_Setp','Basecutter_pressu
re','Chopper_Hydr_Press','Eng_Load','Basecutter_height','Cruise_Command','G
round_Speed','BaseCutter_Rpm','Chopper_Rpm','Machine_Seconds','Elevator_Sec
onds','Fuel_Level_Pct','Machine_Model','Machine_ID'])


mycursor.execute('SELECT * FROM harvester04')


harvester04db      =      pd.DataFrame(data=mycursor,      columns      =
['Date','Timestamp','Prim_Extr_Rpm','Prim_Extr_Rpm_Setp','Basecutter_pressu
re','Chopper_Hydr_Press','Eng_Load','Basecutter_height','Cruise_Command','G
round_Speed','BaseCutter_Rpm','Chopper_Rpm','Machine_Seconds','Elevator_Sec
onds','Fuel_Level_Pct','Machine_Model','Machine_ID'])


mycursor.execute('SELECT * FROM harvester05')


harvester05db      =      pd.DataFrame(data=mycursor,      columns      =
['Date','Timestamp','Prim_Extr_Rpm','Prim_Extr_Rpm_Setp','Basecutter_pressu
re','Chopper_Hydr_Press','Eng_Load','Basecutter_height','Cruise_Command','G
round_Speed','BaseCutter_Rpm','Chopper_Rpm','Machine_Seconds','Elevator_Sec
onds','Fuel_Level_Pct','Machine_Model','Machine_ID'])


mycursor.execute('SELECT * FROM harvester06')


harvester06db      =      pd.DataFrame(data=mycursor,      columns      =
['Date','Timestamp','Prim_Extr_Rpm','Prim_Extr_Rpm_Setp','Basecutter_pressu
re','Chopper_Hydr_Press','Eng_Load','Basecutter_height','Cruise_Command','G
round_Speed','BaseCutter_Rpm','Chopper_Rpm','Machine_Seconds','Elevator_Sec
onds','Fuel_Level_Pct','Machine_Model','Machine_ID'])


harvester01db.head(5)


harvester02db.head(5)


harvester03db.head(5)
```

```
harvester04db.head(5)


harvester05db.head(5)


harvester06db.head(5)



harvesters                                                              =
pd.concat([harvester01db,harvester02db,harvester03db,harvester04db,harveste
r05db,harvester06db])

harvesters


harvesters = harvesters[harvesters.columns[2:15]]

harvesters


harvesters.info()


harvesters.describe()


harvesters.corr()


plt.figure(figsize=(20, 15))

sns.heatmap(harvesters.corr(), annot=True)


from sklearn.model_selection import train_test_split


X_train,           X_test,           y_train,           y_test           =
train_test_split(harvesters[['Ground_Speed',
'Cruise_Command','Basecutter_height']].values,         harvesters.Eng_Load,
test_size=0.33, random_state=0)


from sklearn.ensemble import RandomForestRegressor
```

```
# create regressor object
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)


# fit the regressor with x and y data
regressor.fit(X_train, y_train)


y_pred =regressor.predict(X_test)


from sklearn.metrics import mean_squared_error


mean_squared_error(y_pred,y_test)


mean_squared_error(y_pred,y_test)


harvesters['Eng_Load'].plot()


Ground_Speed = float(input("Enter Ground_Speed: "))

Cruise_Command = float(input("Enter Cruise_Command: "))

Basecutter_height = float(input("Enter Basecutter_height: "))


Enter Ground_Speed:  5.25
Enter Cruise_Command:  4.49
Enter Basecutter_height:  124


result                                                                  =
regressor.predict([[Ground_Speed,Cruise_Command,Basecutter_height]]).reshap
e(1, -1)

result = result.round(2)# input must be 2D array

type(result)

print(result)


[[71.08]]
```