Sincronização na Nuvem de Dados Sensoriais de Pecuária

Thiago de Oliveira Soares

Sincronização na Nuvem de Dados Sensoriais de Pecuária

Thiago de Oliveira Soares

Orientadora: *Profª Drª Hana Karina Salles Rubinsztejn*
Coorientador: *Drº Camilo Carromeu*

Dissertação entregue a Faculdade de Computação da Universidade Federal de Mato Grosso do Sul - FACOM-UFMS como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada.

FACOM - Campo Grande
Março/2024

Agradecimentos

Primeiramente gostaria de agradecer aos meus pais, pelo constante apoio e incentivo, durante essa jornada.

À minha esposa Erika, que sempre me apoiou e incentivou durante todo o ciclo de desenvolvimento do projeto. Sem esse apoio eu não conseguiria concluir esse projeto.

À minha orientadora, professora Drª Hana Karina Salles Rubinsztejn, pela paciência, orientações e por toda ajuda durante as fases do projeto.

Ao meu coorientador, Drº Camilo Carromeu, pela paciência e contribuições ao meu trabalho, dando dicas e ideias durante o desenvolvimento.

Ao meu companheiro de mestrado Ygo Brito, que me ajudou durante a fase de desenvolvimento do projeto.

Por fim, agradeço a Deus, que me sustentou todo esse tempo, fisicamente e espiritualmente, passando por uma pandemia e alcançando essa vitória.

Abstract

Cattle farming plays a crucial role in the Brazilian economy, being essential to the country's Gross Domestic Product (GDP). Therefore, the efficiency and sustainability of this sector are fundamental to maintaining its competitiveness. In this context, technological innovations represent indispensable tools for enhancing livestock monitoring and management. The e-Cattle platform emerges as an integrated solution for precision livestock farming, offering advanced features for effective herd monitoring and data-driven decision support. This work focuses on the development of two critical components for the e-Cattle platform: the first aimed at synchronization of locally collected sensory data with the cloud, ensuring the uniformity and integrity of the information essential for accurate analyses and informed decisions. The second component aims at implementing a dedicated memory manager, designed to prevent excessive data accumulation, ensuring operational continuity and efficiency of the platform. Both innovations are fundamental for optimizing cattle management, representing significant steps toward smarter and more sustainable production.

Resumo

A pecuária bovina desempenha um papel crucial na economia brasileira, sendo essencial para o Produto Interno Bruto (PIB) do país. A eficiência e sustentabilidade deste setor são, portanto, fundamentais para manter sua competitividade. Neste contexto, inovações tecnológicas representam ferramentas indispensáveis para aprimorar o monitoramento e a gestão da produção pecuária. A plataforma e-Cattle surgiu como uma solução integrada para a pecuária de precisão, oferecendo recursos avançados para o monitoramento efetivo do rebanho e suporte à tomada de decisão baseada em dados. Este trabalho foca no desenvolvimento de dois componentes críticos para a plataforma e-Cattle: o primeiro direcionado à sincronização de dados sensoriais coletados localmente com a nuvem, garantindo a uniformidade e a integridade das informações essenciais para análises precisas e decisões informadas. O segundo componente visa a implementação de um gerenciador de memória dedicado, projetado para prevenir a acumulação excessiva de dados, assegurando a continuidade e a eficiência operacional da plataforma. Ambas as inovações são fundamentais para otimizar a gestão da pecuária bovina, representando passos significativos em direção a uma produção mais inteligente e sustentável.

Conteúdo

Lista de Figuras

Listings

Lista de Abreviaturas

API *Application Programming Interface*

BSD Berkeley Software Distribution

EMBRAPA Empresa Brasileira de Pesquisa Agropecuária

FACOM Faculdade de Computação

HTTP *Hypertext Transfer Protocol*

IoT *Internet of Things*

JSON *JavaScript Object Notation*

JWT *JSON Web Token*

NoSQL *Not Only SQL*

MAC *Media Access Control*

UFMS Fundação Universidade Federal de Mato Grosso do Sul

VM *Virtual Machine*

CI *Integração Contínua*

YAML *YAML Ain't Markup Language*

GC *Garbage Collector*

VM *Máquina Virtual*

GCP *Google Cloud Platform*

CAPÍTULO

1

Introdução

A pecuária bovina no Brasil desempenha um papel fundamental na economia do país, com o rebanho bovino crescendo a cada ano. Segundo um relatório divulgado pela Associação Brasileira das Indústrias Exportadoras de Carnes (ABIEC), o Brasil possui o segundo maior rebanho do mundo, aproximadamente 202 milhões de cabeças, representando 12,18% do rebanho bovino mundial. Além disso, é o maior exportador de carne bovina do mundo, com 27,7% em 2022, superando as exportações de 2021 por 471 mil toneladas, totalizando cerca de 12,97 bilhões de dólares, um aumento de 40,8% em relação a 2021 (ABIEC, 2023).

Com o aumento do rebanho, mudanças são necessárias para uma gestão aprimorada, impondo desafios relacionados ao bem-estar animal. Neste contexto, a Embrapa Gado de Corte desempenha um papel crucial no desenvolvimento de soluções voltadas à pecuária de precisão, buscando otimizar insumos e aprimorar a produção bovina.

A utilização de tecnologias possibilita o desenvolvimento de sistemas de produção mais sustentáveis, ampliando a capacidade de monitoramento do rebanho, de forma a oferecer indicadores produtivos, comportamentais e fisiológicos, beneficiando o bem-estar dos animais (Embrapa, 2020).

Segundo Milanez et al. (2020) para que o Brasil continue atendendo à demanda global por alimentos, torna-se imprescindível a adoção de tecnologias avançadas. Nesse panorama, a incorporação de dispositivos IoT (Internet das Coisas) no setor agroindustrial não apenas possibilita a expansão da produção, mas também assegura a redução de custos e o aprimoramento da produtividade. Além disso, promove uma gestão mais eficiente de recursos, contribuindo para a sustentabilidade do processo produtivo.

1.1 Motivação

De acordo com Nery and Britto (2022), em 2021, 90% dos lares brasileiros tinham acesso à internet, sendo que em propriedades rurais houve um aumento de 16,9% em relação ao ano de 2019, alcançando 74,7% dos domicílios. Esse aumento pode ser atribuído à expansão dos serviços de internet, como o Starlink[1], que oferece conexões de alta qualidade em áreas remotas ou rurais.

Em colaboração com a Faculdade de Computação (FACOM) da Universidade Federal de Mato Grosso do Sul (UFMS), a Embrapa Gado de Corte vem desenvolvendo trabalhos focados em pecuária de precisão. Esses projetos coletam informações por meio de sensores, abrangendo dados comportamentais, microclimáticos, fisiológicos e contextuais, que são processados localmente de forma individualizada.

Visando aprimorar o gerenciamento desses dados, Carromeu (2019) propôs a plataforma e-Cattle (Uma Plataforma de IoT para Pecuária de Precisão), que integra diversos serviços, desde a coleta de dados até a implementação de softwares para processamento e disponibilização das informações aos usuários finais.

A plataforma foi idealizada para ser implementada em propriedades rurais, que muitas vezes possuem um ambiente inóspito e hostil, com o mínimo de esforço. Além de ser acessível, possuindo um baixo custo, fácil implantação, com uma interface simples para interpretação o usuário, possuindo escalabilidade, onde a propriedade pode ter diversos nós implantados, e por fim, ser de código aberto, garantido sua confiabilidade e constante evolução.

A implementação é realizada diretamente nas fazendas por meio de um *middleware* intitulado "BigBoxx". Atendendo às premissas mencionadas anteriormente, optou-se pelo emprego de tecnologias de baixo custo, sendo utilizado como *hardware* o RaspBerry Pi, que atua fazendo a comunicação entre os sensores instalados no campo e os dispositivos que irão consumir os dados coletados.

É importante salientar que o Raspberry Pi, por ser um *hardware* de pequeno porte, possui algumas limitações, principalmente no que diz respeito à sua capacidade de armazenamento, que não é muito elevada, pois utiliza cartões de memória. Além disso, é crucial garantir a sincronização dos dados com a nuvem para evitar perdas e garantir a acessibilidade, especialmente em pesquisas e projetos acadêmicos que demandam o armazenamento de grandes volumes de dados.

[1] https://www.starlink.com/map

Este trabalho propõe mitigar os desafios de disponibilizar os dados sensoriais armazenados localmente para uma aplicação na nuvem. Isso permitirá ampliar o uso do e-Cattle para aplicações remotas, possibilitando o acesso aos dados sensoriais mesmo não estando na rede da propriedade. Isso, por sua vez, permitirá gerenciar e tomar decisões remotamente.

1.2 Objetivos

O objetivo geral deste trabalho foi possibilitar a sincronização dos dados sensoriais armazenados localmente no dispositivo BigBoxx com a plataforma e-Cattle disponibilizada na nuvem, bem como gerenciar a memória disponível do BigBoxx. Com essas soluções, é possível disponibilizar os dados na nuvem, assegurando o acesso às informações por todos os usuários já cadastrados na plataforma e também evitar a sobrecarga do sistema local com os registros armazenados.

Para alcançar os objetivos deste projeto, foram definidos os seguintes objetivos específicos:

- Implementação do módulo, denominado Agente de Sincronismo, responsável pela sincronização dos dados sensoriais com a nuvem;
- Desenvolvimento do módulo para o gerenciamento da memória disponível no BigBoxx;
- Disponibilização dos módulos para instalação no BigBoxx utilizando pacotes Snap;
- Atualização dos módulos *Kernel* e *Totem* do BigBoxx para uma versão recente, incluindo correções de *bugs* e melhorias no gerenciamento das variáveis de ambiente;
- Validação do funcionamento do Agente de Sincronismo por meio da realização de testes no BigBoxx.

Todos os objetivos propostos durante o desenvolvimento deste trabalho foram alcançados. Foi implementado um módulo eficaz para sincronização dos dados com a nuvem, além da criação de um módulo para o gerenciamento da memória, que previne a sobrecarga de dados e garante a continuidade das operações. Também realizamos a disponibilização dos módulos para uma instalação simplificada, utilizando os pacotes Snap, e efetuamos correções nos módulos *Kernel* e *Totem* para garantir seu funcionamento em futuras implementações. Por fim, foi realizado o teste para validar o funcionamento do Agente de Sincronismo no BigBoxx.

Visando promover a transparência, a confiabilidade e o contínuo aprimoramento, os componentes desenvolvidos neste trabalho, bem como toda a plataforma e-Cattle[2], estão disponíveis publicamente no GitHub, sob a licença Berkeley Software Distribution (BSD).

1.3 Organização do Texto

Este trabalho está organizado em 5 capítulos. O Capítulo 2 apresenta as tecnologias utilizadas no desenvolvimento do projeto, além de discutir trabalhos que abordam a sincronização de dados de sensores com a nuvem, utilizando os conceitos de conectividade, microsserviços e *multi-tenant*, com foco na temática Agro. O Capítulo 3 descreve a plataforma e-Cattle e as metodologias utilizadas para alcançar os objetivos propostos. O Capítulo 4 apresenta os resultados obtidos por essa pesquisa. Por fim, o Capítulo 5 oferece as conclusões, destacando as principais contribuições e sugerindo direções para trabalhos futuros.

[2] https://github.com/e-cattle/synchronize

CAPÍTULO 2

Fundamentação Teórica

Este capítulo tem como foco apresentar os conceitos das tecnologias utilizadas no desenvolvimento deste trabalho. A Seção 2.1 traz uma visão das tecnologias empregadas durante o desenvolvimento; em seguida, apresenta-se, na Seção 2.2, os conceitos da arquitetura *multi-tenant* e computação em nuvem. A Seção 2.3 aborda uma revisão dos trabalhos relacionados à sincronização. Por fim, na Seção 2.4, são apresentadas as considerações finais.

2.1 Tecnologias Utilizadas

Esta seção apresenta as definições e conceitos básicos das tecnologias empregadas no projeto, incluindo o sistema operacional voltado para IoT utilizado na execução da aplicação, assim como as tecnologias e metodologias adotadas durante o desenvolvimento.

2.1.1 Snapcraft

O Snapcraft é a ferramenta responsável pelo empacotamento e distribuição de aplicações em formato de pacotes Snaps. Utiliza uma máquina virtual (VM), como LXD ou Multipass, para compilar esses pacotes, isolando bibliotecas e outros arquivos que poderiam causar conflitos no sistema, garantindo independência entre eles (Canonical, 2024a).

Snaps são pacotes de software criados e instalados em forma de contêiner Linux independentes, sendo seguros, multiplataformas e livres de dependências (Canonical, 2024b).

O Snapcraft utiliza um arquivo de configuração YAML para definir os requisitos e as dependências da aplicação. Esse arquivo facilita o processo de criação e automatiza a construção do Snap, permitindo a integração com sistemas de integração contínua (CI) para um desenvolvimento eficiente (Canonical, 2024a).

Diante das características apresentadas, a adoção dos snaps para o empacotamento do módulos permite a instalação no *hardware* do BigBoxx de maneira simples e rápida.

2.1.2 Ubuntu Core

O Ubuntu Core é um sistema operacional Linux leve e modular, especialmente desenvolvido para dispositivos IoT. Sua arquitetura minimalista garante alta segurança, confiabilidade e baixo consumo de recursos, tornando-o ideal para aplicações em ambientes com restrições de conectividade e recursos computacionais (Canonical, 2023).

Vantagens para sua utilização:

Segurança: O Ubuntu Core utiliza pacotes Snap, que são isolados e autocontidos, garantindo um ambiente de execução seguro e minimizando os riscos de ataques cibernéticos.

Robustez: Ubuntu Core é projetado para ser confiável e estável. Ele foi testado em uma variedade de hardware e condições operacionais.

Facilidade de uso: O sistema é projetado para ser fácil de instalar, configurar e atualizar. Os pacotes Snap são instalados e atualizados automaticamente, incluindo patches de segurança e correções de bugs, reduzindo a necessidade de intervenção manual e garantindo a confiabilidade do sistema ao longo do tempo.

O Ubuntu Core é utilizado como base deste trabalho. Sua segurança, confiabilidade, gerenciamento simplificado e ausência de pacotes desnecessários facilitam a instalação e o *deploy* dos módulos.

2.1.3 Docker e Docker Compose

Docker é uma plataforma de código aberto projetada para o desenvolvimento, envio e execução eficiente de aplicativos. Ele isola os aplicativos da infraestrutura subjacente, facilitando sua entrega rápida (Docker, 2024b).

Além disso, ele unifica o gerenciamento de infraestrutura e aplicativos, permitindo o uso das mesmas ferramentas e processos para ambos, otimizando o fluxo de trabalho de desenvolvimento.

O Docker permite a criação de contêineres, que são unidades leves, portáteis e autossuficientes que encapsulam o software e todas as suas depen-

dências. Os contêineres podem ser executados em ambientes isolados, livres de conflitos com outras aplicações ou com o sistema operacional, tornando-as mais portáteis e fáceis de mover entre diferentes ambientes (Docker, 2024b).

Os contêineres Docker são eficientes e autossuficientes, contendo todo o necessário — código, bibliotecas e arquivos de configuração — para a execução dos aplicativos, garantindo sua portabilidade e consistência entre diferentes ambientes (Docker, 2024b).

O gerenciamento de múltiplos contêineres é simplificado pelo Docker Compose, uma ferramenta complementar ao Docker que facilita a definição e execução de aplicativos multicontêiner. Com ele, é possível centralizar a gestão da pilha de aplicativos, definindo serviços, redes e volumes em um único arquivo de configuração YAML. Assim, todos os serviços podem ser ativados simultaneamente com um único comando (Docker, 2024a).

2.2 Computação em Nuvem e Multi-tenant

Nesta seção, apresentamos os trabalhos relacionados às arquiteturas *multi-tenant* e computação em nuvem aplicadas ao setor agropecuário, abordando conceitos e desafios identificados pelos autores.

A *Cloud Computing*, ou computação em nuvem, representa um paradigma que oferece aos usuários serviços baseados no modelo *pay-per-use* (pague pelo uso), eliminando a necessidade de preocupações com infraestrutura ou armazenamento por parte dos clientes (Odun-Ayo et al., 2017).

A adoção da computação em nuvem tem crescido substancialmente, sendo implementada em diversos setores. Essa tecnologia garante acesso onipresente e sob demanda a recursos de computação configuráveis e praticamente ilimitados. Como resultado, simplifica a entrega de serviços, permitindo seu uso em qualquer local com acesso à Internet e favorecendo o surgimento de novos aplicativos e redes inteligentes (Matharu et al., 2014; Ahmed et al., 2017).

O conceito de *multi-tenant*, essencial para a computação em nuvem, possibilita que múltiplos usuários compartilhem os mesmos recursos, promovendo eficiência e escalabilidade (Odun-Ayo et al., 2017). Segundo Nguyen et al. (2019), a adoção de microsserviços pode promover uma personalização mais eficaz do ambiente na nuvem. Isso porque eles podem ser desenvolvidos e operados de maneira isolada, atendendo a um dos requisitos críticos de *multi-tenant*. Ademais, os microsserviços facilitam a integração e a entrega contínua de produtos.

Esta seção apresentou trabalhos que demonstram a importância das arquiteturas *multi-tenant* no manejo do acesso às informações e no tratamento do

volume de dados gerados por dispositivos, assegurando a proteção dos dados armazenados na nuvem. As vantagens dessas arquiteturas, inclui a simplificação na entrega de produtos e a garantia de acesso remoto às informações que serão sincronizadas, sem a necessidade de gestão de infraestrutura.

2.3 Trabalhos Relacionados

Esta seção apresenta uma revisão dos trabalhos relacionados à sincronização de dados sensoriais com a nuvem, identificando projetos similares que possam contribuir para o aprimoramento de nossa pesquisa.

Neste contexto, são apresentados alguns trabalhos pertinentes à sincronização dos dados na nuvem no domínio agropecuário. Buscamos explorar as soluções adotadas e identificar oportunidades para aprimorar nossa pesquisa, visando um melhor gerenciamento dos dados da plataforma e-Cattle.

De acordo com Embrapa (2020), o uso de tecnologias pode auxiliar no desenvolvimento e na observação do rebanho. A implantação de uma ampla variedade de sensores, integrados a dispositivos para otimização de processos, introduz o desafio e a necessidade de sincronização dos dados, permitindo que as informações sejam interpretadas pelos usuários ou por uma aplicação na tomada de decisão (Zervopoulos et al., 2020).

Neste contexto, Dieng et al. (2017) propôs uma solução para prevenir o roubo de gado. A solução envia os sinais coletados a um *gateway* LoRa, realizando a sincronização dos dados com a nuvem, sempre que a conectividade com a internet estiver disponível. Para fazendas isoladas e sem conexão com a internet, os autores sugeriram a disponibilização dessas informações em *smartphone* ou *tablet* via *WiFi* ou *Bluetooth*.

Por sua vez, Taneja et al. (2019) apresenta uma plataforma IoT assistida por computação em nuvem. O projeto adota um design orientado a microsserviços para desenvolver uma aplicação distribuída que supera cenários de conectividade restrita à internet. Os dados coletados são enviados para uma plataforma baseada em névoa (*fog computing*), onde são classificados e analisados, oferecendo percepções sobre o bem-estar dos animais. Segundo os autores, esse processo resultou em uma redução de 84% no total de dados sincronizados, em comparação com uma abordagem tradicional baseada em nuvem.

Outra abordagem para a sincronização de dados é proposta por Bhargava et al. (2019). Neste estudo, os autores sugerem uma comunicação orientada a eventos com o *gateway*, na qual os dados são transmitidos apenas quando ocorre alguma alteração no dispositivo. Essa estratégia, segundo os autores, melhora a eficiência do sistema por meio da redução de transmissões e do

custo operacional, eliminando a necessidade de conectividade contínua com a internet e reduzindo a transmissão de pacotes desnecessários para a nuvem.

No trabalho de Vani and Rao (2016), os autores desenvolveram um sistema de monitoramento da umidade do solo. Os dados coletados pelos sensores são transmitidos a um *gateway* e sincronizados com a nuvem utilizando uma conexão WiFi.

Outra solução a sincronização dos dados pode ser vista em Stolárik (2021). O autor aborda a implementação de um *gateway* móvel para possibilitar a conexão entre dispositivos IoT e sistema em nuvem. Foi desenvolvido um aplicativo que garante uma transferência bidirecional de dados e oferece uma interface gráfica para o monitoramento dos dados por parte dos usuários.

Diversas abordagens para a sincronização de dados com a nuvem foram propostas, cada uma atendendo a requisitos específicos para facilitar o acesso à informação em fazendas com variados níveis de conectividade. As soluções que empregam dispositivos intermediários para processamento ou sincronização dos dados, como mencionado em Dieng et al. (2017), Taneja et al. (2019), e Stolárik (2021), são particularmente relevantes para propriedades com conectividade limitada ou inexistente.

Contudo, nenhuma das soluções estudadas aborda todos os requisitos deste trabalho. Entre as necessidades não atendidas estão o gerenciamento de memória para dispositivos de baixo custo como Raspberry Pi e a flexibilização da sincronização de dados na nuvem.

2.4 Considerações Finais

Este capítulo destacou a importância e o impacto das tecnologias emergentes, como a computação em nuvem, a sincronização de dados e as arquiteturas *multi-tenant*, no setor agropecuário. Demonstrou-se como essas inovações podem otimizar processos, facilitar a tomada de decisões baseada em dados e promover práticas sustentáveis, enfrentando desafios como a conectividade limitada em áreas rurais e a necessidade de infraestrutura escalável para gerenciamento de dados.

As discussões apresentadas neste capítulo sublinham a importância da inovação tecnológica no agro. A integração dessas tecnologias não apenas melhora o desempenho operacional das fazendas, mas também contribui para a adoção de práticas agrícolas mais sustentáveis e responsáveis.

Por fim, realizamos um comparativo entre os trabalhos encontrados, a fim de buscar possíveis melhorias para o desenvolvimento dos módulos. Além de destacar algumas das tecnologias utilizadas na implementação deste trabalho.

CAPÍTULO

3

Metodologia

O e-Cattle é uma infraestrutura baseada na Internet das Coisas (IoT), voltada para a integração, monitoramento e automação dos serviços na pecuária de precisão.

A revisão dos trabalhos relacionados evidenciou a necessidade de uma camada intermediária para a sincronização dos dados oriundos de sistemas IoT com a nuvem. Este estudo foca na sincronização de dados sensoriais do e-Cattle para a nuvem.

Neste capítulo, é apresentada uma visão geral dos módulos que são o foco desta pesquisa, bem como a motivação e os requisitos que norteiam seu desenvolvimento. Na Seção 3.1 é apresentada a arquitetura da plataforma e-Cattle e, em seguida, na Seção 3.2 é discutida a sincronização dos dados sensoriais com a nuvem. A Seção 3.3 aborda o módulo de gerenciamento da memória do BigBoxx, o *Garbage Collector (GC)*. Por fim, a Seção 3.4 apresenta as considerações finais do capítulo.

3.1 Arquitetura

A Plataforma e-Cattle foi projetada visando a integração de diversas soluções IoT, voltadas para área da pecuária de precisão, desenvolvidas pela parceria da EMBRAPA e FACOM.

A plataforma oferece uma interface amigável e intuitiva para o monitoramento centralizado dos componentes da fazenda, tanto os espalhados pelo terreno quanto aqueles conectados aos animais, como pode ser observado na Figura 3.1.

O desenvolvimento desta plataforma foi projetado em uma arquitetura de

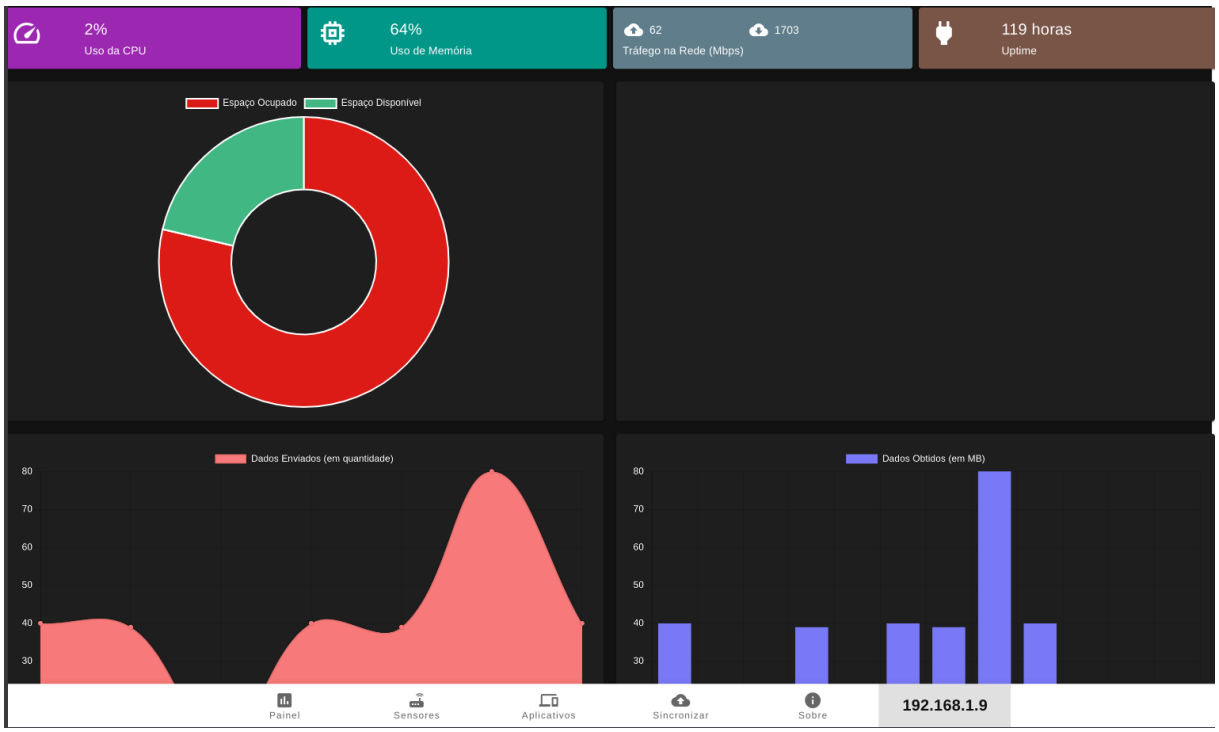


Figura 3.1: Tela inicial do BigBoxx.

seis camadas, visando simplificar o desenvolvimento de softwares, aumentar a flexibilidade nas atualizações e promover a interoperabilidade dos componentes de hardware (Carromeu, 2019). No entanto, Brito (2023) introduziu uma nova camada, denominada "Ambiente em Nuvem", inserida entre as camadas de Serviços e Aplicação (Figura 3.2).

- **Camada de Sensores** refere-se aos componentes usados na coleta de dados ambientais, comportamentais e fisiólogos.
- **Camada de Agregação** inclui coordenadores que são responsáveis pelo envio dos dados coletados pelos sensores à **Camada de Coleta de Dados**.
- **Camada de Coleta de Dados** pode receber requisições HTTP contendo as informações dos dados a serem armazenados, ou por meio de tecnologias de comunicação IoT, como MQTT e LoRA.
- **Camada de Persistência** é encarregada da classificação e armazenamento das informações em um banco de dados NoSQL, onde os dados são organizados como documentos em coleções (*collections*).
- **Camada de Serviços** tem a função de disponibilizar os dados de forma dinâmica para as aplicações consumidoras.
- **Ambiente em Nuvem** é responsável pelo gerenciamento das propriedades e dos usuários, além do armazenamento dos dados do BigBoxx na nuvem.

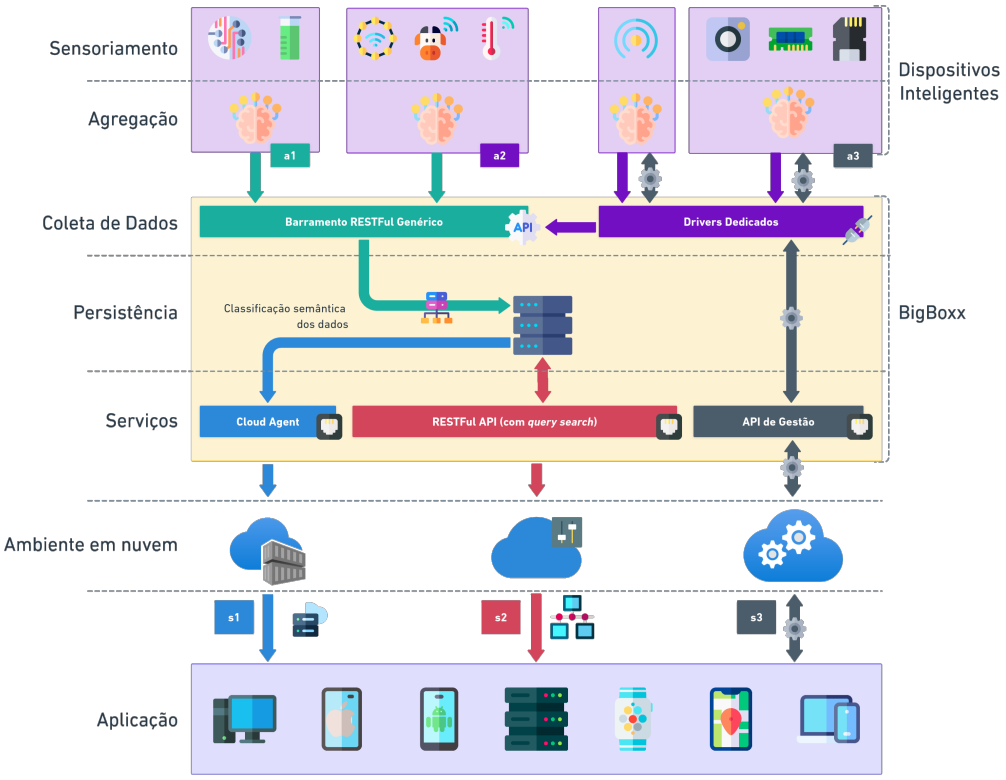


Figura 3.2: Arquitetura da plataforma e-Cattle.
Fonte: Brito (2023)

- **Camada de Aplicação** engloba os softwares que utilizam os dados providos pela **Camada de Serviços**, visando a automação da fazenda e auxiliando o produtor na tomada de decisões.

O principal *software* da plataforma e-Cattle é um *middleware* denominado BigBoxx (Carromeu, 2019). Ele é composto por múltiplas camadas que asseguram a coleta, persistência e disponibilização dos dados dos sensores para aplicações de alto nível, facilitando a tomada de decisões estratégicas. O *hardware* escolhido para a implantação do BigBoxx foi o Raspberry Pi, por se tratar de um computador de baixo custo. Contudo, em razão do tamanho compacto do *hardware*, sua capacidade de armazenamento é limitada e depende de um cartão de memória. Para contornar isso, propomos a implementação do "*Garbage Collector* (GC)" para o BigBoxx, que será responsável por prevenir a sobrecarga de dados e garantir a eficiência operacional.

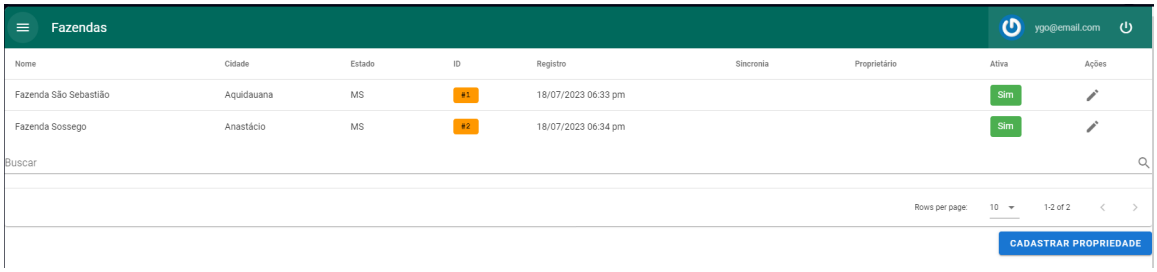
Para a gestão das propriedades e a persistência dos dados na nuvem na camada "Ambiente em Nuvem", três aplicações foram desenvolvidas (Brito, 2023):

- **Gestor de Inquilinos** permite o gerenciamento das propriedades e dos usuários da fazenda, atribuindo papéis de *viewer*, *manager* e *owner* (Fi-

gura 3.3-a);

- **Portal *Web*** apresenta as informações das fazendas, assim como os dados sincronizados do BigBoxx com a nuvem, e é responsável também pela aprovação do vínculo do BigBoxx à propriedade (Figura 3.3-b);

- **Scheduler Job** é responsável pela criação das propriedades no ambiente em nuvem.



a) Tela da aplicação Gestor de Inquilinos.

b) Tela da aplicação Portal *Web*.

Figura 3.3: Telas das aplicações web.
Fonte: Adaptada de Brito (2023)

Para realizar a sincronização dos dados com a nuvem, este trabalho implementou o *Cloud Agent*, denominado nesta pesquisa como "Agente de Sincronismo", um serviço da Camada de Serviços essencial para garantir a persistência na nuvem dos dados coletados localmente por sensores.

3.2 Sincronização dos Dados Sensoriais

A camada de foco deste projeto é a Camada de Serviços (Figura 3.2), que disponibiliza três serviços essenciais do tipo *pull/push*. O primeiro serviço (*s1*) é encarregado de transmitir os dados coletados pelo e-Cattle para a nuvem. O segundo serviço (*s2*) relaciona-se ao barramento de consulta de dados, que permite aos aplicativos externos consumir esses dados. Por último, o terceiro serviço (*s3*) serve como interface para a gestão dos dispositivos (Carromeu, 2019).

O serviço primordial para este projeto é o *s1*, o *Cloud Agent*, que facilita a sincronização dos dados do e-Cattle com a nuvem. Essa sincronização ocorre assim que a rede da propriedade estabelece conexão com a internet, permitindo o acesso remoto aos dados da fazenda por meio de aplicativos externos.

A Figura 3.4 apresenta o fluxograma que o gestor da propriedade deverá realizar para permitir a sincronização dos dados no ambiente em nuvem do e-Cattle.

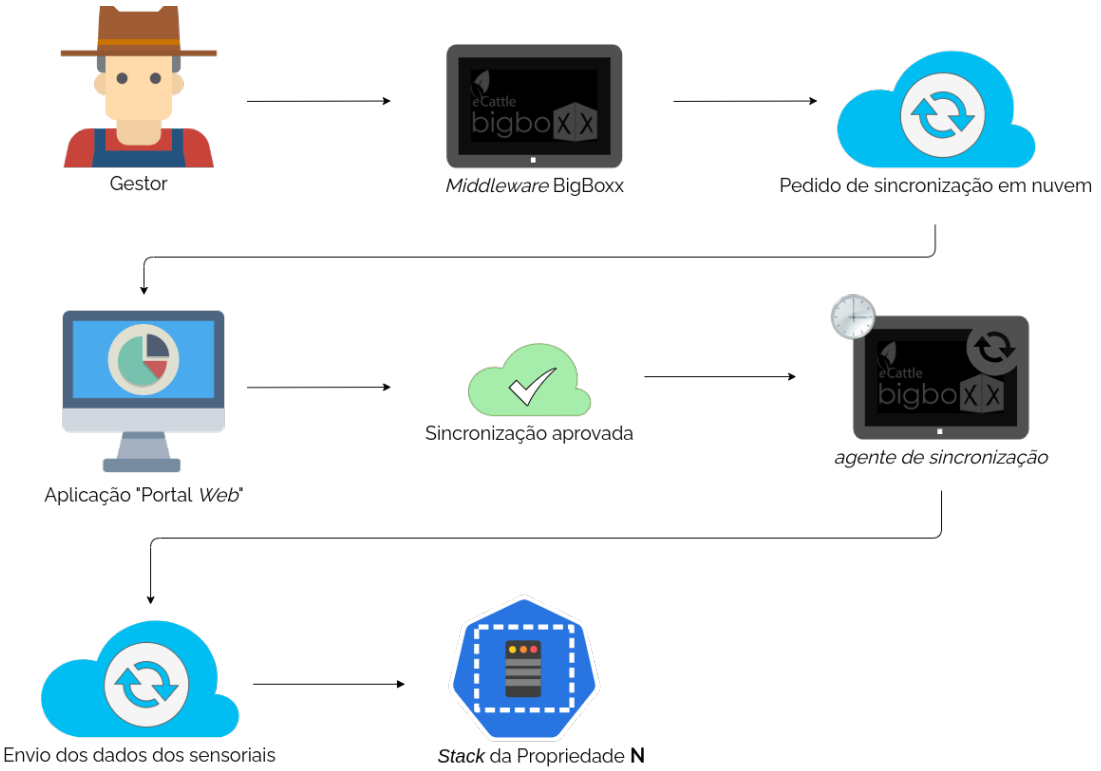


Figura 3.4: Fluxograma da sincronização no ambiente em nuvem.
Fonte: Adaptada de Brito (2023)

Inicialmente, o gestor da propriedade precisa indicar no *middleware* BigBoxx o ID da fazenda (Figura 3.5) que deseja realizar a sincronização com a nuvem, aguardando, em seguida, a aprovação na aplicação Portal Web. Essa aprovação se faz necessária para garantir que aquele ID é da fazenda em questão, evitando o risco de inserção de um ID incorreto. O pedido deve ser aceito por um usuário com o papel de *Manager* ou *Owner* (Brito, 2023). Com a aprovação, o Agente de Sincronismo obtém os dados necessários para a sincronização com a *stack*[1] específica da propriedade, como o ID da Fazenda e o *token* de autenticação da requisição.

Os registros da propriedade são armazenados em uma aplicação *multi-tenant* na nuvem, o que significa que cada propriedade mantém seus dados de forma isolada das demais.

A sincronização desses dados é realizada por um agente de sincronização na nuvem, que comunica com o módulo de sincronização do BigBoxx. Esta sincronização se aplica somente em ambientes com acesso à internet, permitindo que dados coletados, especialmente aqueles de maior relevância, sejam

[1] *stack* é uma junção de vários serviços que compõem uma aplicação ou uso (Portainer, 2023)

Figura 3.5: Tela de solicitação do sincronismo em nuvem do BigBoxx.

sincronizados com a nuvem para uma visualização remota das condições da propriedade.

Para garantir a integridade e segurança dos dados que serão sincronizados na aplicação na nuvem (*Cloud API*), todas as transações requerem autenticação. Essa autenticação dos dispositivos é efetuada por meio de um *token JSON Web Token* (JWT)[2], fornecido pelo *kernel* do BigBoxx ao estabelecer conexão com a nuvem. Além do JWT, o ID da fazenda também é providenciado, o que é necessário para a sincronização dos dados.

3.3 Garbage Collector (GC)

Diante do grande volume de dados sensoriais coletados, existe o risco de esgotamento do espaço de armazenamento disponível no BigBoxx, uma vez que o *hardware* utilizado é de pequeno porte e possui limitações na sua capacidade de armazenamento. Para enfrentar esse desafio, foi implementado o *Garbage Collector* (GC) na plataforma e-Cattle. O objetivo do GC é preservar a integridade do BigBoxx, prevenindo a sobrecarga de dados armazenados.

O GC será encarregado de remover os dados sincronizados com a nuvem assim que a ocupação da memória no BigBoxx for igual ou superior à 80%. Na ausência de dados sincronizados e para garantir a eficiência e o desempenho do BigBoxx, foi definido que a capacidade de armazenamento em disco não pode ser inferior a 10%, quando isso ocorrer, os N registros mais antigos das *collections* serão identificados e excluídos. Esse procedimento será repetido até que se libere pelo menos 10% da capacidade de memória.

[2]O JWT é um padrão (RFC 7519) para transmitir informações seguras entre sistemas.

As configurações do GC serão definidas em um arquivo JSON, que especificará a quantidade de registros a serem deletados por execução. Além disso, este arquivo conterá um campo para determinar quais *collections* serão priorizadas para exclusão, conforme a importância designada pelo administrador, e um campo que estabelece o limiar de memória para a primeira exclusão de registros pelo GC.

3.4 Considerações Finais

Este capítulo apresentou os módulos desenvolvidos nesta pesquisa, oferecendo uma visão abrangente da plataforma e-Cattle. Destacou-se a importância do Agente de Sincronismo, detalhando as etapas que devem ser realizadas pelo gestor da propriedade para permitir a comunicação efetiva do BigBoxx com a nuvem. Além disso, ressaltou-se a relevância do Garbage Collector (GC), que contribui significativamente para a confiabilidade do armazenamento dos dados, evitando a sobrecarga e garantindo a continuidade das operações e a sustentabilidade a longo prazo do BigBoxx.

CAPÍTULO

4

Resultados

Neste capítulo, abordaremos os resultados obtidos a partir do desenvolvimento dos projetos descritos no Capítulo 3. A Seção 4.1 discute o processo de desenvolvimento do Agente de Sincronismo e do Garbage Collector (GC), com ênfase nas tecnologias utilizadas. Na Seção 4.2, focamos no ambiente de desenvolvimento utilizado para os módulos, detalhando os procedimentos de instalação e execução. A Seção 4.3 destaca a implantação desses módulos no Raspberry Pi. A Seção 4.4 apresenta os resultados dos testes realizados para avaliar o desempenho no Raspberry Pi. Por fim, as considerações finais deste capítulo são apresentadas na Seção 4.5.

4.1 Desenvolvimento

O Agente de Sincronismo é uma aplicação *backend* responsável pela sincronização dos dados armazenados no BigBoxx com a nuvem. Sua implementação utiliza a linguagem de programação Python[1], que é linguagem de alto nível e reconhecida por sua versatilidade e adoção em diferentes áreas.

Todo o processo de sincronização do Agente de Sincronismo é realizado por meio de uma solicitação `HTTP/POST` direcionada ao agente da nuvem associado à fazenda.

Para realizar a sincronização com a nuvem dos dados armazenados no BigBoxx, incluindo contratos, dispositivos e diversos tipos de sensores, foram criados três *endpoints*, cada um destinado a um tipo de valor. Os *endpoints* contêm o ID (<FARM-ID>) da fazenda para a qual o agente deve realizar a

[1] https://www.python.org/about/apps/

sincronização dos dados:

- **Sincronização de Contratos:**
  - *http://e-cattle.cnpgc.embrapa.br/<FARM-ID>/cloud/contracts*;
- **Sincronização de Dispositivos:**
  - *http://e-cattle.cnpgc.embrapa.br/<FARM-ID>/cloud/devices*;
- **Sincronização de Sensores:**
  - *http://e-cattle.cnpgc.embrapa.br/<FARM-ID>/cloud/sensors*.

A Lista 4.1 contém um exemplo do modelo JSON enviado na requisição com a nuvem do sensor de temperatura (*type-air-temperature*). Este JSON inclui o tipo da coleção, permitindo que o agente na nuvem determine a coleção apropriada para o armazenamento dos dados.

```
{
    "type": "<SENSORS>",
    "id": "5e1362627cb1853275270dc8",
    "storage_date": "2020-01-06T15:18:16.231Z",
    "device_id":  "5e1362627c61853275270dc6",
    "value": 33,
    "created_at": "2020-01-06T16:40:22.223Z",
    "resource": "Curral da Fazenda X"
}
```

Listagem 4.1: *JSON* da Requisição do Agente de Sincronismo com a nuvem.

Com o avanço contínuo das tecnologias, o Agente de Sincronismo foi desenvolvido exclusivamente para ambientes com acesso à internet. A Figura 4.1 representa o diagrama de processos realizados pelo agente, dividido em quatro etapas principais.

Inicialmente, o Agente de Sincronismo executa validações preliminares para iniciar a sincronização dos dados com a nuvem. O primeiro passo é verificar se o BigBoxx está habilitado para sincronizar seus dados. Se a sincronização estiver ativa, valida-se o *token* de acesso e o ID da fazenda. Após essa etapa, confirma-se se a fazenda está ativa e então é realizada uma requisição, via API REST, do Agente de Sincronismo para o *container* correspondente da fazenda na nuvem, garantindo que o mesmo esteja disponível e recebendo as requisições. Qualquer falha nestas validações interrompe o processo, aguardando uma nova tentativa.

Seguindo a validação, a segunda etapa consiste na verificação de contratos não sincronizados no BigBoxx. Caso afirmativo, realiza-se a solicitação de

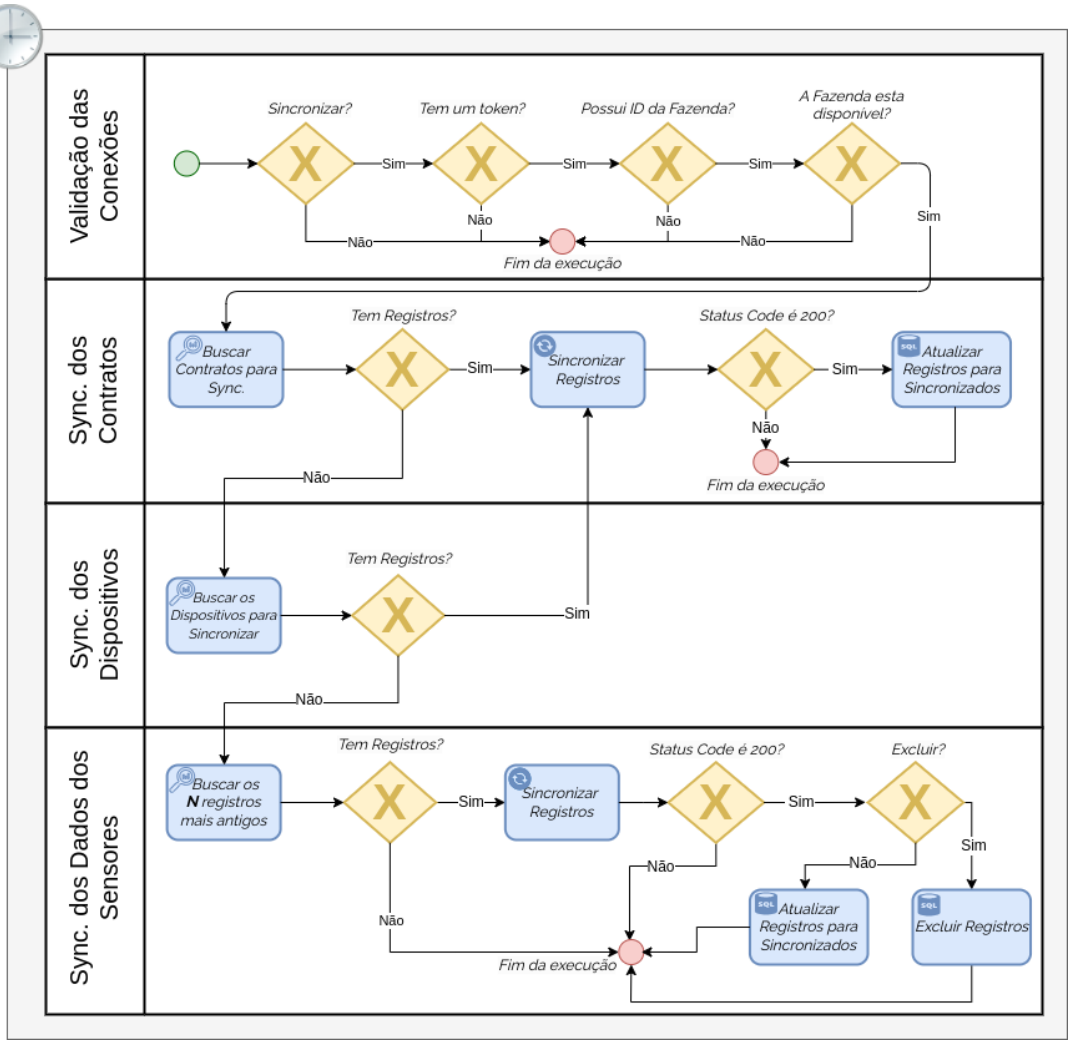


Figura 4.1: Fluxograma do Agente de Sincronismo.

atualização. Se a atualização for bem-sucedida (status_code 200), os dados são marcados como sincronizados, e o processo é finalizado até a próxima execução.

A terceira etapa consiste na verificação e sincronização dos dispositivos. Uma nova verificação é realizada no BigBoxx e uma solicitação de atualização é feita, semelhante à sincronização dos contratos.

Concluídas as etapas anteriores, na sua próxima execução, o agente inicia a última etapa: a busca e sincronização dos N dados sensoriais mais antigos não sincronizados de todas as *collections* do banco de dados. Os dados são enviados de forma assíncrona para o agente da nuvem correspondente da fazenda, que retornará um dos seguintes *status_code*:

- **200:** Indica que os dados foram sincronizados com sucesso. Uma *flag*, indicando que os dados foram sincronizados, será adicionada aos registros (Figura 4.2). Após a sincronização dos dados, caso o usuário solicite a exclusão dos registros, os mesmos serão excluídos localmente.

- **500:** Se ocorrer algum erro durante o envio dos dados, o Agente de Sin-

cronismo os enviará novamente na próxima sincronização.

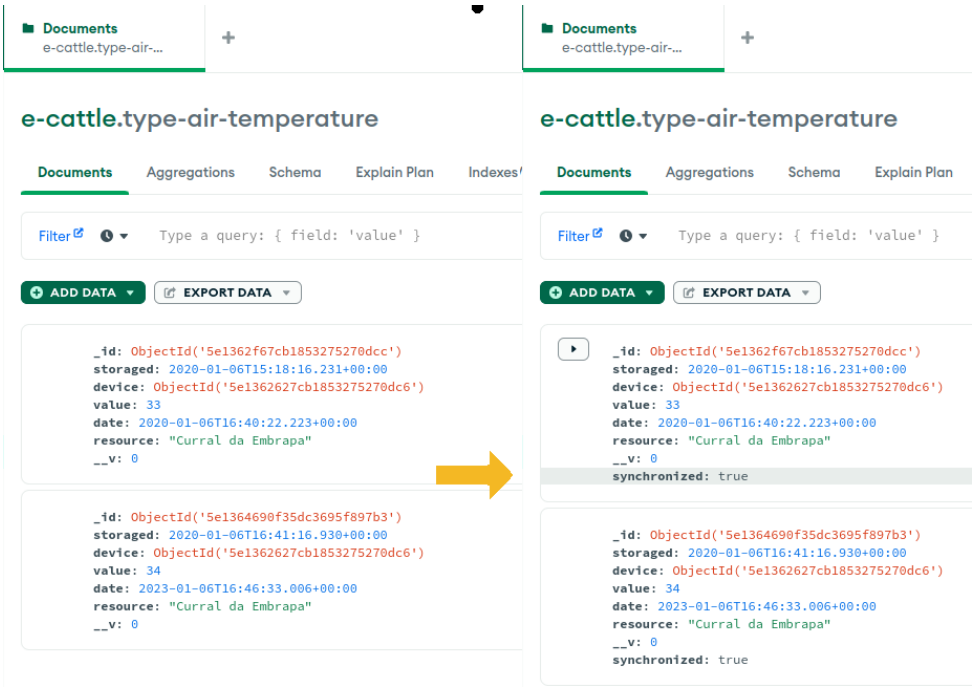


Figura 4.2: Sincronização dos dados relativo ao sensor *type-air-temperature*.

Após a conclusão do fluxo de sincronização, o Agente de Sincronismo será automaticamente encerrado. Uma nova execução é agendada para ocorrer a cada 15 minutos, garantindo a atualização contínua dos dados.

Paralelamente, desenvolveu-se *Garbage Collector* (GC), responsável pela gestão da memória do BigBoxx. A Figura 4.3 representa o diagrama de processo desse módulo.

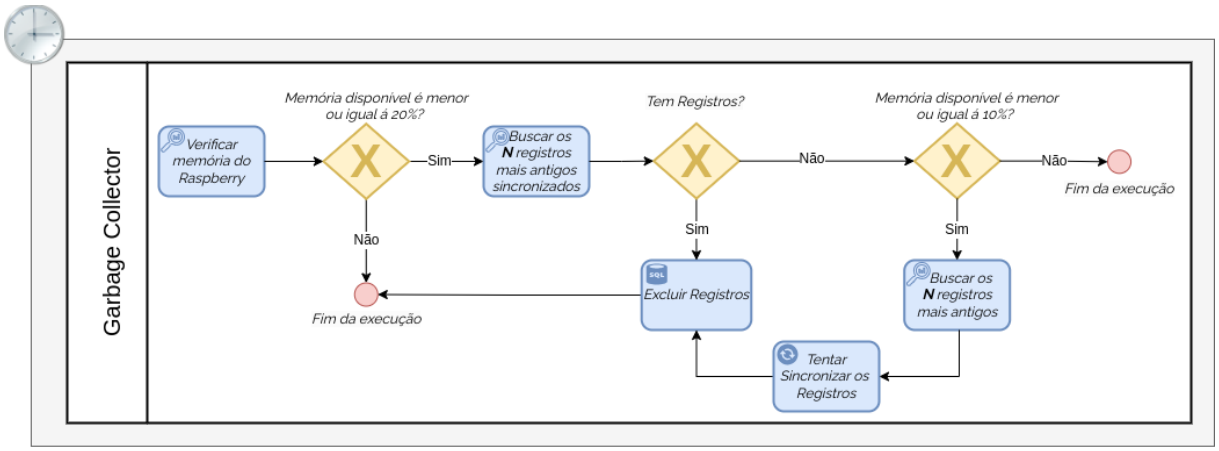


Figura 4.3: Fluxograma do *Garbage Collector*.

O GC opera automaticamente, com execuções iniciais programadas para cada 15 minutos. Sua função é verificar o status da memória e, se necessário, eliminar dados antigos já sincronizados para liberar espaço.

O GC verifica se a quantidade de memória disponível é menor ou igual a 20%, em caso afirmativo, o sistema busca os *N* dados mais antigos, que foram previamente sincronizados, em todas as *collections* do banco de dados, e os deleta se confirmada a sincronização. Por padrão, o valor é definido como 1000.

Se não houver dados sincronizados, uma nova verificação da memória utilizada é feita. Quando a memória disponível cai para 10% ou menos, o sistema seleciona os *N* registros mais antigos de todas as *collections* para exclusão, mas tenta sincronizá-los antes de efetuar a remoção. Este processo é repetido até que seja liberada uma margem acima de 10% da memória.

O Agente de Sincronismo e o GC têm suas configurações armazenadas em um arquivo no formato JSON e podem ser alterados de acordo com as necessidades e preferências do usuário.

No Apêndice A, é possível consultar este arquivo. Ele inclui o número de registros a serem sincronizados ou excluídos a cada execução, um campo usado para especificar quais *collections* devem ser priorizadas, de acordo com a importância definida pelo administrador, dois campos para determinar a porcentagem de memória a ser utilizada como critério inicial e final na exclusão de registros pelo GC, uma chave para indicar se a sincronização está ativada, entre outros aspectos relevantes.

4.2 Ambiente de Desenvolvimento Utilizando Docker

O e-Cattle possui um projeto disponível no GitHub, denominado "*middleware*"[2], o qual tem como objetivo facilitar a manutenção e o desenvolvimento de novos componentes para o BigBoxx. O projeto emprega as tecnologias Docker e Docker Compose para gerenciar e orquestrar os contêineres do BigBoxx (Cáceres, 2019). Essas tecnologias facilitam o desenvolvimento e os testes das aplicações de maneira isolada, assegurando a uniformidade de seu funcionamento em diferentes sistemas operacionais.

O desenvolvimento do Agente de Sincronismo e do GC foi realizado utilizando as tecnologias Docker para contêineres e Docker Compose como mecanismo de gerenciamento desses contêineres.

O projeto inclui um arquivo `docker-compose.yaml`, que contém todas as configurações necessárias para a execução do Agente de Sincronismo, incluindo volumes configurados para facilitar o desenvolvimento. Dessa forma, quaisquer alterações feitas nos arquivos são imediatamente refletidas no contêiner.

Para configurar o ambiente de desenvolvimento, o usuário deve fazer o

[2] https://github.com/e-cattle/middleware

clone do projeto *synchronize*[3], utilizando o comando **`git clone`**, acessar a pasta *synchronize* e, em seguida, executar o comando **`docker-compose up`**. Este comando instala as imagens do projeto, bem como a imagem do banco de dados NoSQL MongoDB, simulando o ambiente de sincronização do BigBoxx.

```
1 ~ $ mkdir BigBoxx
2 ~ $ cd BigBoxx
3 ~/BigBoxx$ git clone https://github.com/e-cattle/synchronize.git
4 ~/BigBoxx$ cd synchronize
5 ~/synchronize$ docker-compose up
```

Listagem 4.2: Iniciando o ambiente de desenvolvimento do Agente de Sincronismo

Com o objetivo de simplificar futuras implantações e manutenções, o projeto *middleware* foi atualizado. Essa atualização inclui a instalação automatizada do Agente de Sincronismo e do GC, garantindo que todo o sistema BigBoxx possa ser instalado de forma eficiente e descomplicada.

4.3 Implementação no Raspberry Pi

O projeto possui um arquivo `snapcraft.yaml` para a implantação do Agente de Sincronismo e GC no Ubuntu Core. Este arquivo lista os procedimentos e dependências necessárias para a execução da aplicação, incluindo a criação de um *daemon* que permite a inicialização automática dos módulos. Detalhes sobre a configuração podem ser encontrados no Apêndice B, onde o arquivo YAML é disponibilizado para consulta.

A Figura 4.4 mostra o processo de construção (`build`) do módulo de sincronização na *Snap Store*. O módulo foi vinculado ao repositório *GitHub* e registrado com um nome *Snap*, denominado "`BigBoxx-sync`".

A Figura 4.4 demonstra o processo de construção (`build`) do módulo de sincronização na *Snap Store*. O módulo foi vinculado ao reposit

Com o vínculo estabelecido com o repositório Git, o portal de *builds* monitora constantemente por qualquer alteração. Caso ocorra um novo `commit` na *branch master*, o *snap* é automaticamente reconstruído em várias arquiteturas. Se algum erro for detectado durante esse processo, a última versão estável é mantida.

A implementação do Agente de Sincronismo e do GC, foi realizada no Raspberry Pi 3 Model B+[4], um microcomputador equipado com um processador quad-core Broadcom BCM2837B0 (ARMv8 Cortex-A53 de 1,4 GHz com arquitetura de 64bits), 1GB de memória RAM e interfaces wireless LAN e Bluetooth Low Energy (BLE), conforme ilustrado na Figura 4.5.

[3] https://github.com/e-cattle/synchronize
[4] https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/

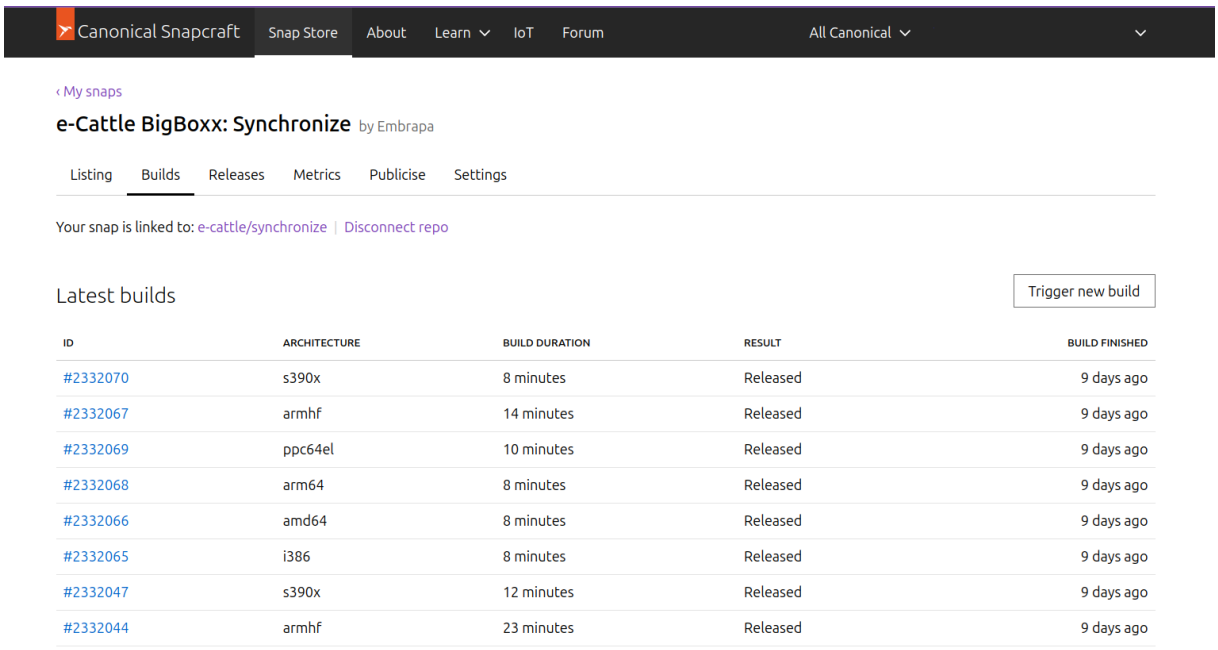


Figura 4.4: Build do módulo de Sincronização/GC na *Snap Store*.

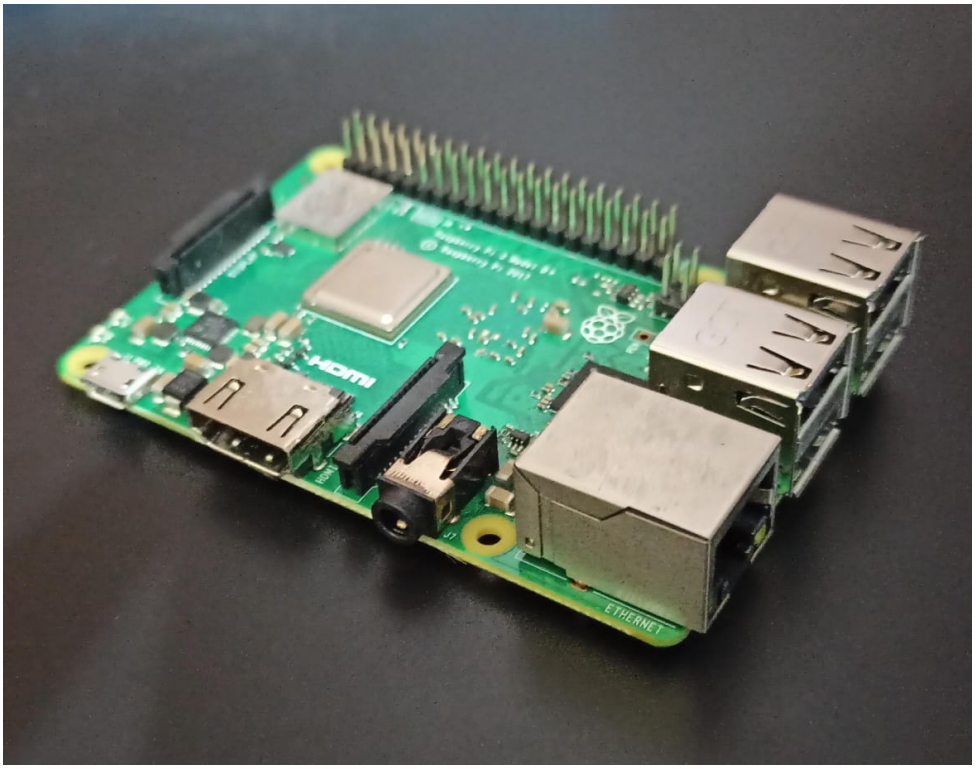


Figura 4.5: Raspberry Pi 3 Model B+ utilizado no trabalho.

O Ubuntu Core 18[5], otimizado para plataformas IoT e desenvolvido pela Canonical, foi o sistema operacional escolhido para este projeto. Este sistema operacional, compatível com os demais módulos do BigBoxx, promove um ambiente seguro e adequado para a execução de aplicações IoT.

A instalação dos módulos no Raspberry Pi, utilizando o sistema operacional Ubuntu Core 18, foi efetuada por meio de pacotes *Snaps*, detalhados na

[5] https://cdimage.ubuntu.com/ubuntu-core/18/stable/current/

Listagem 4.3. Esses pacotes facilitam a instalação do projeto em contêineres, oferecendo segurança aprimorada durante a execução, atualizações automáticas e simplificação na manutenção do sistema.

```
1 ~ $ snap install BigBoxx-kernel --devmode --edge
2 ~ $ snap install BigBoxx-totem --devmode --edge
3 ~ $ snap install BigBoxx-lora --devmode --edge
4 ~ $ snap install BigBoxx-sync --devmode --edge
5 ~ $ snap install mir-kiosk
6 ~ $ snap install wpe-webkit-mir-kiosk
7 ~ $ snap set wpe-webkit-mir-kiosk url="http://localhost:3002"
```

Listagem 4.3: Inicialização do BigBoxx no Raspberry

Para assegurar a funcionalidade completa e eficiente do BigBoxx no Raspberry Pi, foi indispensável realizar ajustes e atualizações nas configurações e dependências dos módulos. Estas modificações incluíram a atualização do NodeJs para a versão 16.20.1 no arquivo `snapcraft.yaml` de configuração do BigBoxx, devido à depreciação de algumas bibliotecas essenciais para o funcionamento dos módulos *Totem* e *Kernel*. Adicionalmente, ajustes específicos na configuração das variáveis do *Kernel* foram necessários para testar a sincronização do BigBoxx com a nuvem.

4.4 Teste do Agente de Sincronismo no Raspberry Pi

Com o propósito de avaliar a eficiência do Agente de Sincronismo, foram conduzidos testes focados no uso de processamento e de memória, bem como na capacidade de suporte à comunicação assíncrona.

Para assegurar uma análise precisa, foi desenvolvido um script específico para auxiliar na coleta dos dados. A fim de minimizar interferências externas que poderiam afetar os resultados, foram instalados no Raspberry Pi apenas os componentes essenciais: `BigBoxx-kernel`, `BigBoxx-totem` e `BigBoxx-sync`. A conexão utilizada no Raspberry Pi foi cabeada, com velocidades médias de download de 26 Mbps e upload de 31 Mbps.

O ambiente em nuvem foi simulado em uma Máquina Virtual (VM) da Google Cloud Platform (GCP)[6], com o sistema operacional Ubuntu 22.04 LTS, utilizando uma máquina da série E2[7] com 2 vCPUs (1 núcleo) e 8 GB de memória RAM, executando o ambiente por meio de *containers* através da orquestração do docker-compose.

Para a execução dos testes, foram inseridos no BigBoxx 600 mil dados simulados dos sensores, incluindo medições de temperatura (*type-air-temperature*),

[6] https://cloud.google.com/

[7] https://cloud.google.com/compute/vm-instance-pricing?e2_custommachinetypepricing

dióxido de carbono (*type-co2*) e umidade relativa (*type-relative-humidity*).

Antes da execução do Agente de Sincronismo, foram coletados dados referentes à utilização da CPU e da memória RAM. O consumo de memória RAM do BigBoxx estava em **61.7%** com uma variação de **0.4%**, enquanto o consumo de CPU estava aproximadamente em **3.8%**, com um desvio padrão de **1.9%**.

Os testes foram divididos em cinco etapas. Na primeira etapa, conduzimos o teste com 100 registros, aumentando para 500 registros na segunda etapa. Na terceira etapa, utilizamos 1000 registros, seguidos por 2500 registros na quarta etapa. Por fim, na última etapa do teste, foram sincronizados 5000 registros. Para cada etapa, foram realizadas 30 execuções de sincronização com a nuvem, com um intervalo aproximado de 1 minuto entre elas. Durante essas solicitações, foram coletados dados sobre o consumo de CPU, memória RAM, o tempo de inicialização e o processamento das requisições do agente de sincronismo.

A inicialização do Agente de Sincronismo teve uma duração média aproximada de 9 segundos após sua reinicialização. O gráfico apresentado na Figura 4.6 ilustra o tempo médio de execução do Agente de Sincronismo durante o processo de sincronização com a nuvem. Nota-se uma variação gradual no tempo necessário para sincronizar os registros na nuvem, dependendo da quantidade de registros envolvidos.

Ao sincronizar 100 registros, o processo foi concluído em 13.4 segundos, resultando em uma taxa média de sincronização de aproximadamente 7.4 registros por segundo. Em contrapartida, ao sincronizar 5000 registros, o tempo médio de sincronização aumentou para 44.68 segundos, representando um incremento de aproximadamente 3.34 vezes em relação ao tempo necessário para a sincronização inicial de 100 registros. Além disso, a quantidade de registros sincronizados aumentou em 50 vezes.

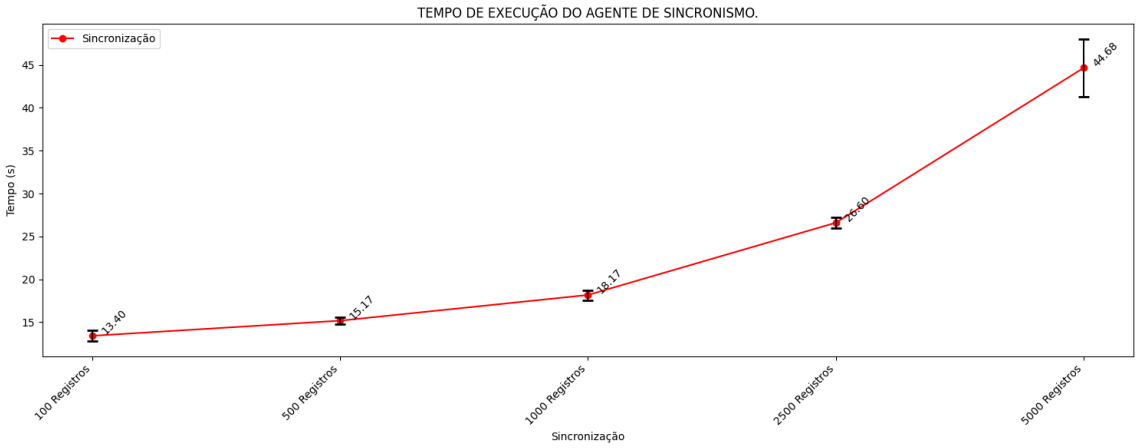


Figura 4.6: Tempo de Execução do Agente de Sincronismo.

A Figura 4.7 ilustra os resultados referentes ao consumo médio de CPU e memória RAM. Podemos observar que a média de CPU nos testes é em torno de **36,96%**, com um desvio padrão médio de **8,84%**, enquanto a memória RAM teve um aumento médio no consumo de **5,3%**, chegando a **67%**, com um desvio padrão de **2,67%**. Diferentemente do tempo de execução do Agente de Sincronismo, o consumo de memória e CPU permaneceu estável, com pouca variação em relação à quantidade de registros sincronizados.

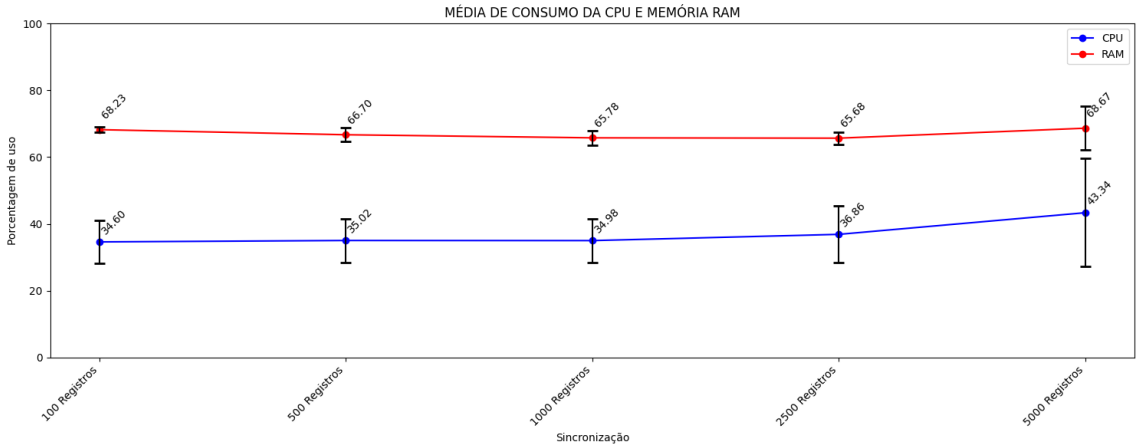


Figura 4.7: Média de Consumo da CPU e Memória RAM.

Os resultados dos testes demonstram que o módulo obteve um bom desempenho, conseguindo realizar a sincronização sem ocasionar uma sobrecarga do sistema. De modo geral, em todos os testes, a memória teve um aumento médio de **5.3%**, já a utilização média da CPU ficou abaixo de **40%** durante a sincronização dos dados. Vale salientar que essa diferença entre os picos de CPU dos testes provavelmente foi provocada por algum processo do sistema operacional.

4.5 Considerações Finais

Este capítulo teve como objetivo apresentar as principais implementações realizadas neste trabalho, destacando o Agente de Sincronismo e o GC (Garbage Collector). O Agente de Sincronismo é essencial para a sincronização dos dados com a nuvem, enquanto o GC desempenha um papel crucial no gerenciamento da memória do BigBoxx, assegurando a otimização dos recursos disponíveis.

Adicionalmente, discutiu-se a abordagem de desenvolvimento do projeto, que adota o Docker como base para facilitar a integração com outros componentes e garantir compatibilidade com diferentes sistemas operacionais. Essa estratégia visa não apenas simplificar o processo de desenvolvimento, mas

também promover uma implantação mais ágil e confiável do BigBoxx em variados ambientes operacionais.

A automatização da instalação dos módulos através do uso de *snaps* foi outra medida importante, visando facilitar a distribuição e instalação dos componentes em diversas arquiteturas. Essa abordagem contribui para a escalabilidade e a flexibilidade do sistema, permitindo que ele se adapte a diferentes contextos de uso com eficiência.

Para avaliar a robustez e a integração do Agente de Sincronismo com o BigBoxx, foram conduzidos testes específicos. Os resultados obtidos confirmam a viabilidade de uso do Agente de Sincronismo, evidenciando o sucesso de sua implementação e a capacidade de atender às necessidades de sincronização de dados na nuvem de forma eficaz.

Por fim, visando promover a transparência, a confiabilidade e o constante aprimoramento, todos os componentes desenvolvidos neste trabalho, assim como toda a plataforma e-Cattle[8], estão disponíveis publicamente no GitHub.

[8] https://github.com/e-cattle

CAPÍTULO

5

Conclusões

Neste trabalho, abordamos os desafios e soluções associadas ao armazenamento de dados e sua sincronização na nuvem, focando no contexto da plataforma e-Cattle utilizada na pecuária de precisão. Apresentamos as principais contribuições deste trabalho na Seção 5.1 , bem como perspectivas para futuras pesquisas na área a Seção 5.2.

5.1 Principais Contribuições

O desenvolvimento deste projeto teve como objetivo principal aprimorar o gerenciamento de dados dentro do *middleware* BigBoxx da plataforma e-Cattle, focando em dois aspectos críticos: armazenamento de dados persistidos e sua sincronização eficiente com a nuvem.

Conseguimos implementar um módulo eficaz para a sincronização de dados armazenados localmente, assegurando a padronização desses dados ao serem enviados para o ambiente em nuvem. Este avanço representa uma melhoria significativa na integridade e na consistência dos dados gerenciados, fundamental para a tomada de decisões baseada em dados na pecuária de precisão.

Além disso, a criação de um módulo dedicado ao gerenciamento da memória do BigBoxx endereçou o desafio imposto por sua limitada capacidade de armazenamento. Esse módulo não apenas previne a sobrecarga de dados, como também garante a continuidade e a eficiência das operações, essenciais para a sustentabilidade de longo prazo das aplicações na pecuária de precisão.

O próximo passo seria a realização de testes em ambientes reais de produção, visando validar a eficácia e a escalabilidade das soluções desenvolvidas.

Por fim, um artigo foi submetido para conferência e será encaminhado um pedido de registro de software em conjunto com a plataforma web desenvolvida por Brito (2023).

5.2 Trabalhos Futuros

Atualmente o Agente de Sincronismo foi desenvolvido para ambientes com acesso a internet. A pesquisa evidenciou oportunidades significativas para a expansão e aprimoramento do sistema. Uma delas é a adaptação do Agente de Sincronismo para operar em ambientes sem acesso contínuo à internet, por meio de uma abordagem de sincronização em névoa. Isso permitiria o armazenamento de dados provenientes do BigBoxx em um dispositivo intermediário que, ao reconectar-se à internet, sincronizaria os dados com a nuvem, uma funcionalidade especialmente valiosa para fazendas em locais remotos.

Durante o desenvolvimento dos módulos, foram necessários alguns ajustes nas aplicações já desenvolvidas como o *Totem* e o *Kernel*, por estarem com bibliotecas depreciadas. Identificamos assim a necessidade de manter uma política de atualização constante para as bibliotecas utilizadas no desenvolvimento do BigBoxx, garantindo assim a compatibilidade e a segurança do sistema diante dos rápidos avanços tecnológicos.

Por fim, a implementação de uma interface gráfica para a configuração do Agente de Sincronismo e do GC, atualizando o *Totem UI*, representaria um avanço significativo na usabilidade do sistema, facilitando a gestão e a customização das operações por parte dos usuários.

5.3 Potencialidades para a Pecuária de Precisão

Os resultados e desenvolvimentos alcançados neste trabalho têm o potencial de transformar significativamente a gestão de dados na pecuária de precisão. A capacidade de coletar, armazenar de forma segura e sincronizar dados com a nuvem em tempo real ou em condições de conectividade limitada abre novas possibilidades para a análise de dados e tomada de decisões baseadas em informações precisas e atualizadas.

Essas inovações prometem não apenas melhorar a eficiência operacional das fazendas, mas também contribuir para a sustentabilidade e a produtividade do setor como um todo.

Bibliografia

ABIEC (2023). Perfil da pecuária no brasil - capítulo 3 - 2023. *Associação Brasileira das Indústrias Exportadoras de Carne*. Citado na página 1.

Ahmed, E., Ahmed, A., Yaqoob, I., Shuja, J., Gani, A., Imran, M., e Shoaib, M. (2017). Bringing computation closer toward the user network: Is edge computing the solution? *IEEE Communications Magazine*, 55(11):138–144. Citado na página 7.

Bhargava, K., Ivanov, S., McSweeney, D., e Donnelly, W. (2019). Leveraging fog analytics for context-aware sensing in cooperative wireless sensor networks. *ACM Trans. Sen. Netw.*, 15(2). Citado na página 8.

Brito, Y. A. (2023). Ambiente em nuvem para a plataforma e-cattle utilizando multi-tenant. Dissertação (Mestrado) — Universidade Federal de Mato Grosso do Sul, UFMS. Citado nas páginas 12, 13, 14, 15, e 32.

Canonical (2023). Ubuntu core documentation. Disponível em: https://ubuntu.com/core/docs. Acesso em: 20 de dezembro de 2023. Citado na página 6.

Canonical (2024a). Snap documentation. Disponível em: https://snapcraft.io/docs/snapcraft-overview. Acesso em: 13 de janeiro de 2024. Citado nas páginas 5 e 6.

Canonical (2024b). Snap documentation. Disponível em: https://snapcraft.io/docs. Acesso em: 13 de janeiro de 2024. Citado na página 5.

Carromeu, C. (2019). Uma plataforma de iot para pecuária de precisão. Tese (Doutorado) — Universidade Federal de Mato Grosso do Sul, UFMS. Citado nas páginas 2, 12, 13, e 14.

Cáceres, B. d. A. (2019). Barramento de serviços para consulta de dados sensoriais em iot na plataforma e-cattle. Dissertação (Mestrado) — Universidade Federal de Mato Grosso do Sul, UFMS. Citado na página 23.

Dieng, O., Diop, B., Thiare, O., e Pham, C. (2017). A study on iot solutions for preventing cattle rustling in african context. In *Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing*, ICC '17, New York, NY, USA. Association for Computing Machinery. Citado nas páginas 8 e 9.

Docker (2024a). Docker compose overview. Disponível em: https://docs.docker.com/compose/. Acesso em: 20 de janeiro de 2024. Citado na página 7.

Docker (2024b). Docker overview. Disponível em: https://docs.docker.com/get-started/overview/. Acesso em: 20 de janeiro de 2024. Citado nas páginas 6 e 7.

Embrapa (2020). Pecuária de precisão contribui para desenvolvimento de sistemas de produção sustentáveis e eficientes. Citado nas páginas 1 e 8.

Matharu, G. S., Mishra, A., e Chhikara, P. (2014). A framework to leverage cloud for modernization of indian agricultural produce marketing system. In *Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies*, ICTCS '14, New York, NY, USA. Association for Computing Machinery. Citado na página 7.

Milanez, A. Y., Mancuso, R. V., Maia, G. B. d. S., Guimarães, D. D., Alves, C. E. A., Madeira, R. F., et al. (2020). Conectividade rural: situação atual e alternativas para superação da principal barreira à agricultura 4.0 no brasil. Citado na página 1.

Nery, C. e Britto, V. (2022). Pesquisa nacional por amostra de domicílios. Disponível em: https://www.ibge.gov.br/estatisticas/sociais/trabalho/9171-pesquisa-nacional-por-amostra-de-domicilios-continua-mensal.html?=&t=publicacoes. Acesso em: 25 de julho de 2023. Citado na página 2.

Nguyen, P. H., Song, H., Chauvel, F., Muller, R., Boyar, S., e Levin, E. (2019). Using microservices for non-intrusive customization of multi-tenant saas. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, pagina 905–915, New York, NY, USA. Association for Computing Machinery. Citado na página 7.

Odun-Ayo, I., Misra, S., Abayomi-Alli, O., e Ajayi, O. (2017). Cloud multitenancy: Issues and developments. In *Companion Proceedings of The10th International Conference on Utility and Cloud Computing*, UCC '17 Companion, pagina 209–214, New York, NY, USA. Association for Computing Machinery. Citado na página 7.

Portainer (2023). Stacks. Disponível em: https://docs.portainer.io/user/docker/stacks. Acesso em: 08 de janeiro de 2024. Citado na página 15.

Stolárik, B. M. (2021). A mobile system enabling cloud connectivity of iot devices in smart food industry. Citado na página 9.

Taneja, M., Jalodia, N., Byabazaire, J., Davy, A., e Olariu, C. (2019). Smartherd management: A microservices-based fog computing–assisted iot platform towards data-driven smart dairy farming. *Software: practice and experience*, 49(7):1055–1078. Citado nas páginas 8 e 9.

Vani, P. D. e Rao, K. R. (2016). Measurement and monitoring of soil moisture using cloud iot and android system. *Indian Journal of Science and Technology*, 9(31):1–8. Citado na página 9.

Zervopoulos, A., Tsipis, A., Alvanou, A. G., Bezas, K., Papamichail, A., Vergis, S., Stylidou, A., Tsoumanis, G., Komianos, V., Koufoudakis, G., et al. (2020). Wireless sensor network synchronization for precision agriculture applications. *Agriculture*, 10(3):89. Citado na página 8.

APÊNDICE

A

Arquivo de Configuração do Agente de Sincronismo e do GC

O arquivo `config.json` contém as os parâmetros essenciais para a execução dos módulos Agente de Sincronismo e *Garbage Collector*. Esses parâmetros possibilitam a personalização e adaptação dos módulos às necessidades específicas do usuário.

```
{
    "is_sync_active": true,
    "first_sync ": true,
    "del_record_after_sync": false,
    "max_record_sync": 100,
    "prioritize_sync_collections": [],
    "port": 52000,
    "url_farm": "http://e-cattle.cnpgc.embrapa.br/$PORT",
    "url_db": "mongodb://localhost:27017/",
    "db_name": "e-cattle",
    "available_memory_sync": 20,
    "available_memory_not_sync": 10,
    "total_records_to_delete": 1000,
    "prioritize_delete_collections": []
}
```

Listagem A.1: Arquivo config.json

APÊNDICE

B

Snapcraft do módulo de Sincronização

O arquivo snapcraft.yml contém as dependências individuais do módulo de sincronização bem como do *garbage collector*.

```
1 name: bigboxx-sync
2 base: core18
3 version: git
4 summary: IoT Middleware Synchronization Module for the e-Cattle Platform
      for Cattle Farms
5
6 description: |
7     e-Cattle BigBoxx is a middleware to receive, rank, segment, persist and
       provide sensory data for IoT applications in livetstock farms. This
      module, named Synchronize, has the role to send local data to cloud.
8
9 grade: devel
10 confinement: devmode
11
12 apps:
13   synchronize:
14     command: synchronize
15     daemon: simple
16     plugs: [home, network, network-bind]
17     timer: 00:00-24:00/96
18
19   gc:
20     command: gc
21     daemon: simple
22     plugs: [home, network, network-bind]
```

```
23     timer: 00:00-24:00/96
24
25 parts:
26   synchronize:
27     plugin: python
28     python-version: python3
29     source: .
30     requirements: ['requirements.txt']
```

Listagem B.1: Arquivo snapcraft.yml