UNIVERSIDADE FEDERAL DO PAMPA

Luís Fernando Alves da Silva

Arquitetura Híbrida Nuvem-Borda para Sistemas de Irrigação IoT Resilientes

Alegrete
2025

Luís Fernando Alves da Silva

Arquitetura Híbrida Nuvem-Borda para Sistemas de Irrigação IoT Resilientes

Dissertação apresentado ao Programa de Pós-graduação Stricto Sensu em Engenharia de Software da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia de Software.

Orientador: Prof. Dr. Claudio Schepke

Alegrete
2025

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

S586a Silva, Luís Fernando Alves da
Arquitetura Híbrida Nuvem-Borda para Sistemas de Irrigação IoT Resilientes / Luís Fernando Alves da Silva.
100 p.

Dissertação(Mestrado)-- Universidade Federal do Pampa, MESTRADO EM ENGENHARIA DE SOFTWARE, 2025.
"Orientação: Claudio Schepke".

1. Sistemas Distribuídos. 2. IoT. 3. Agricultura Inteligente. 4. Computação em Nuvem. 5. Computação de Borda. I. Título.

LUÍS FERNANDO ALVES DA SILVA

ARQUITETURA HÍBRIDA NUVEM-BORDA PARA SISTEMAS DE IRRIGAÇÃO IOT RESILIENTES

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Software da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia de Software.

Dissertação defendida e aprovada em: 11/08/2025

Banca examinadora:

_____

Prof. Dr. Claudio Schepke

Orientador

(Unipampa)

_____

Prof. Dr. Rodrigo Brandão Mansilha

(Unipampa)

_____

Prof. Dr. Fabio Paulo Basso

(Unipampa)

_____

Prof. Dr. Guilherme Galante

(Unioeste)

Assinado eletronicamente por **FABIO PAULO BASSO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 11/08/2025, às 22:49, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **CLAUDIO SCHEPKE**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 12/08/2025, às 22:40, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Guilherme Galante**, **Usuário Externo**, em 13/08/2025, às 09:07, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **RODRIGO BRANDAO MANSILHA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 13/08/2025, às 18:08, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1803249** e o código CRC **4FBCA6F9**.

Dedicado à memória de meus pais,
Luiz Carlos da Silva e **Cleuza Alves da Silva**.

RESUMO

Este trabalho apresenta o desenvolvimento de uma arquitetura híbrida Nuvem-Borda para sistemas de irrigação inteligente, no contexto da Internet das Coisas (IoT) aplicada à agricultura. A proposta busca enfrentar os desafios característicos de ambientes rurais, marcados por falhas recorrentes de energia e conectividade, garantindo a continuidade operacional do sistema. A arquitetura adota mecanismos de replicação assíncrona bidirecional e consistência eventual, priorizando atualizações realizadas na borda para preservar a integridade dos dados após a reconexão. A prova de conceito demonstrou a viabilidade da solução, evidenciando sua resiliência e sua capacidade de atender às necessidades de pequenos produtores rurais, além de oferecer um modelo escalável e adaptável a outros cenários de infraestrutura restrita. Dessa forma, a arquitetura proposta se configura como uma contribuição relevante para sistemas IoT, ao consolidar uma solução distribuída e robusta em que a camada de borda provê autonomia computacional local, assegurando a continuidade das operações de irrigação mesmo durante períodos de desconexão da nuvem.

Palavras-chave: Sistemas Distribuídos, IoT, Agricultura Inteligente, Computação em Nuvem, Computação de Borda, Tolerância a Falhas.

ABSTRACT

This work presents the development of a hybrid Cloud-Edge architecture for smart irrigation systems in the context of the Internet of Things (IoT) applied to agriculture. The proposal seeks to address the challenges inherent in rural environments, marked by recurrent power and connectivity failures, while ensuring the system's operational continuity. The architecture adopts bidirectional asynchronous replication and eventual consistency mechanisms, prioritizing updates performed at the edge to preserve data integrity after reconnection. The proof of concept demonstrated the viability of the solution, highlighting its resilience and its ability to meet the needs of small rural producers, as well as providing a scalable model adaptable to other infrastructure-constrained scenarios. Thus, the proposed architecture stands as a significant contribution to IoT systems, consolidating a distributed and robust solution in which the edge layer plays a fundamental role by providing local computing autonomy and ensuring the continuity of irrigation operations even during periods of cloud disconnection.

Key-words: Distributed Systems, IoT, Smart Agriculture, Cloud Computing, Edge Computing, Fault Tolerance.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A agricultura é uma atividade vital para a subsistência humana, sendo responsável pela maior parte dos alimentos consumidos no mundo. A irrigação, por sua vez, desempenha um papel essencial na produtividade agrícola, especialmente em culturas que dependem diretamente da disponibilidade hídrica. A aplicação de tecnologias de IoT pode gerar eficiências significativas na agricultura, sendo as técnicas inteligentes de irrigação e fertilização para melhorar a produtividade um de seus principais casos de uso (LEA, 2020, p. 16-17). No entanto, a operação de irrigação tradicional exige intervenção manual frequente para iniciar, monitorar e interromper os ciclos de irrigação, o que implica custos elevados, um desafio particularmente relevante para pequenos produtores.

Com o avanço das tecnologias de Internet das Coisas (IoT) e o surgimento da chamada Agricultura 4.0, tornou-se possível automatizar processos agrícolas, incluindo a irrigação. Essas soluções integram sensores, atuadores e plataformas de software para monitoramento e controle em tempo real. O valor da IoT não reside em um único sensor, mas na agregação de dados de centenas, milhares, potencialmente milhões de sensores (LEA, 2020, p. 33), onde a nuvem atua como o denominador comum que centraliza as informações. Apesar disso, os sistemas disponíveis no mercado são, em sua maioria, projetados para grandes propriedades, como os baseados em pivô central, e não atendem adequadamente às necessidades de pequenos produtores. Além do custo elevado, essas soluções geralmente assumem a existência de conexão contínua com a internet e infraestrutura estável, condições frequentemente ausentes em áreas rurais remotas. O segmento agrícola se destaca justamente por estar localizado em áreas remotas e centros populacionais esparsos, o que gera impactos nos sistemas de comunicação de dados (LEA, 2020, p. 16)

Nesse contexto, torna-se necessária a criação de soluções específicas, de baixo custo, resilientes a falhas de energia e conectividade, e adaptadas às realidades operacionais de pequenos produtores rurais. O desenvolvimento do sistema proposto teve início a partir da demanda de um cliente da empresa *Tulkas Desenvolvimento*[1], especializada em sistemas de informação para pequenos produtores do nordeste de São Paulo e sul de Minas Gerais. Em 2020, esse cliente relatou falhas recorrentes em seu sistema de irrigação hidropônico, o que motivou uma visita técnica para diagnóstico e levantamento de requisitos.

A propriedade em questão é uma unidade de agricultura familiar que adota cultivo hidropônico em canos de PVC elevados. As mudas são irrigadas por meio da circulação de água e nutrientes impulsionada por uma motobomba. Embora o sistema contasse com acionamento automático, apresentava falhas operacionais frequentes, provocadas tanto pelo acúmulo de gases na bomba quanto por quedas constantes de energia elétrica. Tais falhas, quando ocorriam durante o dia, comprometiam rapidamente a produção, resultando na perda de plantas pela interrupção do fluxo hídrico. O problema era agravado

[1] <http://tulkas.com.br>

pela distância entre a plantação e a sede do sítio, exigindo o deslocamento constante de trabalhadores já sobrecarregados. A situação era ainda mais crítica devido à localização da propriedade em uma área montanhosa e de difícil acesso, com conexão de internet limitada a um serviço via satélite de baixa qualidade e alta instabilidade. Outro fator importante identificado foi o baixo nível de familiaridade tecnológica dos responsáveis pela propriedade, embora já fossem atendidos pela *Tulkas* com sistemas de informação voltados à gestão agrícola.

Diante desse cenário, iniciou-se o desenvolvimento de um sistema de irrigação baseado em IoT, originalmente projetado para atender às demandas específicas desse cliente. No decorrer do projeto, a solução foi ampliada e generalizada para abranger outros clientes da *Tulkas Desenvolvimento*, com perfis e desafios operacionais semelhantes, extrapolando o contexto restrito de sistemas hidropônicos.

A arquitetura da solução proposta é composta por dispositivos IoT, organizados em módulos sensores e atuadores, um servidor de borda, responsável por manter o funcionamento local mesmo sem conexão à internet, e um servidor em nuvem, acessível por meio de uma interface web intuitiva. Esse arranjo reflete um padrão recorrente em sistemas IoT, no qual a maioria dos sensores não se conecta diretamente à internet, mas depende de gateways e computadores de borda em um modelo *hub-and-spoke* (LEA, 2020, p. 12). Nesse contexto, o servidor de borda funciona como um nó computacional gerenciado que se estende para mais perto das fontes de dados, estratégia essencial para lidar com falhas de conectividade e prover redundância ao sistema. Além da definição dos componentes de hardware e software, foi necessário projetar uma infraestrutura de comunicação robusta, predominantemente sem fio, capaz de operar de forma resiliente frente às limitações típicas de ambientes rurais, como quedas de energia, instabilidade de conexão e exposição a condições climáticas adversas (LEA, 2020, p. 72-73). Para isso, foram propostas soluções de alimentação energética autônoma, com uso de painéis solares e baterias, bem como mecanismos de proteção física, como invólucros resistentes à água, poeira e intempéries (LEA, 2020, p. 343), com o objetivo de garantir a integridade, a durabilidade e o funcionamento contínuo dos dispositivos em campo.

1.1 Objetivo

O objetivo desta dissertação é desenvolver um sistema de irrigação baseado em Internet das Coisas (IoT), voltado ao contexto de propriedades rurais, com foco inicial nas necessidades de pequenos produtores, sem restringir-se exclusivamente a esse público. A proposta consiste na definição de uma arquitetura capaz de operar em condições adversas, como conectividade instável com a internet e interrupções frequentes no fornecimento de energia elétrica, limitações identificadas nos estudos de campo realizados.

O sistema também foi projetado para operar eficientemente em dispositivos com recursos computacionais limitados, mantendo todas as funcionalidades essenciais. Além

disso, é capaz de identificar situações de falha relacionadas à interrupção ou à intermitência da conexão com a internet, assegurando que a irrigação continue funcionando mesmo em cenários de conectividade instável.

1.2 Justificativa

A busca por maior eficiência na agricultura, aliada à escassez de mão de obra e aos desafios ambientais contemporâneos, reforça a necessidade de automatização de tarefas essenciais, como a irrigação. No entanto, pequenos produtores rurais enfrentam diversos obstáculos para adotar essas tecnologias, especialmente devido ao alto custo de implementação, à precariedade da infraestrutura elétrica e de conectividade, e à baixa familiaridade com recursos digitais. Embora a Agricultura 4.0 e as soluções baseadas em Internet das Coisas (IoT) estejam em expansão, grande parte das tecnologias disponíveis é direcionada a grandes propriedades e sistemas de produção em larga escala, tornando-se inacessíveis ou inadequadas para o contexto das pequenas explorações agrícolas (LEA, 2020, p. 16). Essa disparidade evidencia uma lacuna tecnológica e de mercado que precisa ser endereçada.

Em pesquisa exploratória interna realizada com pequenos produtores rurais durante o segundo semestre de 2023, observou-se um interesse significativo por sistemas de irrigação automatizados. No entanto, os entrevistados apontaram o custo e a complexidade de uso como os principais entraves à adoção dessas tecnologias. Diante desse cenário, justifica-se o desenvolvimento de uma arquitetura para um sistema de irrigação inteligente, de baixo custo, robusto e adaptado às condições reais enfrentadas por pequenos produtores. A proposta apresentada nesta dissertação busca preencher essa lacuna por meio de uma arquitetura modular, com interface simplificada e tolerância a falhas, contribuindo para a sustentabilidade e a autonomia da agricultura familiar.

2 ASPECTOS CONCEITUAIS

Este capítulo apresenta os fundamentos conceituais que sustentam o desenvolvimento da solução proposta, abrangendo as tecnologias, arquiteturas e práticas utilizadas.

2.1 Internet das Coisas e Agricultura 4.0

A Internet das Coisas (IoT) é um paradigma tecnológico baseado na interconexão de objetos físicos que traduzem efeitos físicos analógicos em sinais digitais (LEA, 2020, p. 33). Para ser considerado parte da IoT, um dispositivo deve possuir capacidade computacional para hospedar uma pilha de software de protocolo de internet e hardware para utilizar um transporte de rede, sem ser um dispositivo tradicional como um PC ou *smartphone* (LEA, 2020, p. 10). No contexto agrícola, essa abordagem tem-se mostrado fundamental, pois eficiências significativas podem ser alcançadas (LEA, 2020, p. 16), viabilizando a automação e o monitoramento remoto de atividades críticas, como o uso de técnicas inteligentes de irrigação e fertilização para melhorar a produtividade.

A Agricultura 4.0 representa a incorporação de tecnologias digitais aos processos produtivos do campo. Por meio da integração de sensores, atuadores, sistemas de comunicação, computação de borda e de nuvem, além de análise de dados (LEA, 2020, p. 34), é possível aumentar a eficiência produtiva, reduzir desperdícios e tomar decisões mais precisas com base em dados em tempo real, já que uma parte significativa do valor da IoT está na interpretação e nas decisões baseadas nos dados (LEA, 2020, p. 467). No entanto, a implementação dessas tecnologias enfrenta desafios únicos no setor agrícola, que se destaca por estar em áreas remotas ou com baixa densidade populacional, o que gera impactos nos sistemas de comunicação de dados e no fornecimento de energia confiável.

2.2 Arquitetura Distribuída: Nuvem e Borda

Um sistema distribuído é um conjunto de computadores interconectados em rede, no qual processos e recursos estão suficientemente distribuídos entre múltiplas máquinas (STEEN; TANENBAUM, 2023, p. 4). Nesse sentido, os sistemas de IoT podem ser compreendidos como exemplos característicos de sistemas distribuídos.

A arquitetura do sistema proposto adota o modelo distribuído baseado em computação em nuvem (*cloud computing*) e computação de borda (*edge computing*), uma abordagem cada vez mais comum em sistemas IoT modernos. Essa organização se baseia no modelo *edge-cloud*, que se refere aos componentes posicionados entre os dispositivos da Internet das Coisas (IoT) e os serviços centralizados típicos da nuvem (STEEN; TANENBAUM, 2023, p. 100).

O servidor de borda é responsável por executar a lógica local e garantir a operação do sistema, mesmo diante da ausência de conectividade com a internet. Esta capacidade de operação autônoma é um dos principais padrões de uso da computação de borda,

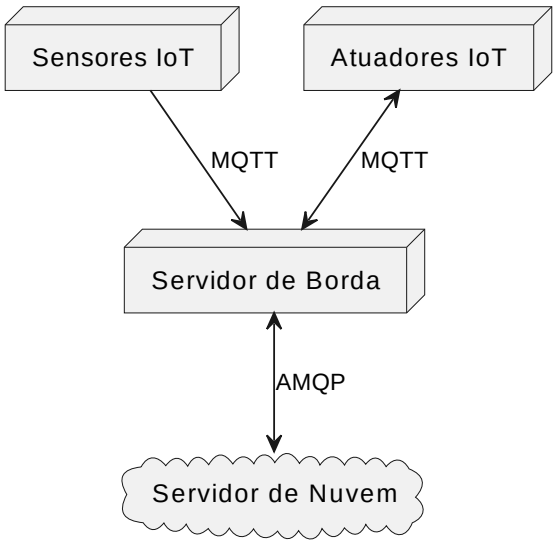

Figura 1 – Arquitetura Distribuída - Nuvem e Borda

conhecida como computação resiliente (LEA, 2020, p. 313). Em cenários com comunicação não confiável, como em áreas rurais, o servidor de borda utiliza armazenamento local (*caching*) para guardar dados até que a conexão seja restabelecida, evitando perdas (LEA, 2020, p. 313). Essa característica reflete o princípio da falha parcial, no qual parte do sistema (a borda) continua a operar mesmo quando a comunicação com a outra parte (a nuvem) falha (STEEN; TANENBAUM, 2023, p. 462).

O servidor na nuvem centraliza a supervisão e a integração remota. A nuvem é descrita como uma infraestrutura de serviços de computação sob demanda que oferece ao usuário uma sensação de independência de localização, permitindo que o sistema seja acessado de qualquer local com conexão disponível (LEA, 2020, p. 431). Assim, enquanto a borda garante a operação local e em tempo real, a nuvem atua como o denominador comum, onde os serviços centralizados são executados.

A Figura 1 representa o *Diagrama de Implantação* do sistema, ilustrando a arquitetura distribuída baseada no modelo nuvem-borda.

2.3 Tolerância a Falhas

A tolerância a falhas é um princípio fundamental no projeto de sistemas distribuídos (COULOURIS et al., 2013, p. 23–25), sendo responsável por permitir que o sistema continue a fornecer os seus serviços mesmo na presença de falhas (STEEN; TANENBAUM, 2023, p. 20). Este é um desafio central, visto que sistemas distribuídos sofrem quase continuamente com falhas parciais, nas quais um componente falha enquanto o restante do sistema continua a operar (STEEN; TANENBAUM, 2023, p. 7, 9). O objetivo ideal é atingir a transparência de falhas, na qual um utilizador ou aplicação não percebe que uma parte do sistema falhou e que este recuperou automaticamente (STEEN;

TANENBAUM, 2023, p. 13). No entanto, é reconhecido que mascarar falhas é uma das questões mais difíceis e, em alguns casos, comprovadamente impossível de se alcançar plenamente (STEEN; TANENBAUM, 2023, p. 9, 13). Dessa forma, este projeto adotou uma abordagem baseada no melhor esforço, priorizando as falhas mais recorrentes no ambiente rural em que o sistema será implantado.

Dentre os diversos modelos de falha, as interrupções de energia elétrica e a perda de conectividade com a internet foram identificadas como as mais críticas no contexto analisado. De acordo com a classificação das fontes, as interrupções de energia podem ser categorizadas como falhas de parada (*crash failures*), nas quais um componente para de funcionar abruptamente, embora estivesse a operar corretamente até esse momento (STEEN; TANENBAUM, 2023, p. 467). A perda de conectividade com a internet, por sua vez, representa uma falha de omissão, na qual um servidor falha em responder a requisições ou em enviar mensagens (STEEN; TANENBAUM, 2023, p. 467). Dependendo de sua duração, essas falhas podem ser consideradas transitórias, intermitentes ou permanentes (STEEN; TANENBAUM, 2023, p. 20). Essas limitações influenciaram diretamente o desenho arquitetural do sistema, orientando a escolha de soluções que favorecessem a resiliência operacional.

Nesse sentido, a arquitetura do sistema aproxima-se do modelo de computação de borda (*edge computing*), que posiciona serviços na borda da rede para aumentar a resiliência (STEEN; TANENBAUM, 2023, p. 100). Esta abordagem é justificada pela percepção de que a conectividade com a nuvem pode não ser suficientemente confiável para muitas aplicações, tornando necessário o uso de infraestruturas de borda (STEEN; TANENBAUM, 2023, p. 102). O sistema foi concebido para lidar com falhas de conectividade entre a borda e a nuvem (operação desconectada), bem como com quedas de energia nos dispositivos de campo e no servidor de borda. Estratégias como alimentação energética autônoma, estabilização elétrica e a capacidade de operar de forma desconectada foram incorporadas como requisitos arquiteturais. Essas decisões visam proporcionar robustez sem comprometer a simplicidade do sistema, promovendo continuidade operacional mesmo em ambientes com infraestrutura limitada.

2.4 Operação Desconectada

A operação desconectada refere-se à capacidade do sistema de continuar funcionando corretamente mesmo quando há uma falha de comunicação entre o servidor de borda e o servidor na nuvem. Essa abordagem se encaixa na arquitetura de borda-nuvem (*edge-cloud*), um estilo híbrido que posiciona serviços na borda da rede (STEEN; TANENBAUM, 2023, p. 98). Essa característica é fundamental em ambientes com conectividade intermitente, como zonas rurais, onde a conexão com a nuvem pode não ser suficientemente confiável para muitas aplicações, justificando o uso de infraestruturas de borda.

Durante o período de desconexão, o sistema demonstra tolerância a falhas, um

objetivo de projeto que visa mascarar falhas parciais e permitir que o sistema continue a fornecer seus serviços (STEEN; TANENBAUM, 2023, p. 18). O servidor de borda continua executando localmente as funcionalidades essenciais, como a coleta de dados dos sensores e a execução dos agendamentos, garantindo o funcionamento autônomo da aplicação. Essa capacidade de operar na ausência da nuvem é um dos principais argumentos para a adoção da computação de borda, especialmente em cenários onde a latência ou a confiabilidade da rede são críticas (STEEN; TANENBAUM, 2023, p. 102).

Quando a conectividade é restabelecida, ocorre a replicação dos dados entre a borda e a nuvem, respeitando os princípios da consistência eventual (*eventual consistency*). Este modelo de consistência garante que, se não ocorrerem novas atualizações, todas as réplicas convergirão gradualmente para um estado idêntico, assegurando que os dados coletados e as ações executadas durante o período offline sejam posteriormente integrados ao sistema central, evitando perdas de informação. A replicação é uma técnica fundamental tanto para a confiabilidade quanto para o desempenho (STEEN; TANENBAUM, 2023, p. 393).

A operação desconectada é viabilizada por uma arquitetura que emprega comunicação persistente e assíncrona (STEEN; TANENBAUM, 2023, p. 191). Em vez de exigir que o emissor e o receptor estejam ativos simultaneamente (comunicação transiente), os sistemas de enfileiramento de mensagens (message-queuing systems) armazenam as mensagens até que possam ser entregues (STEEN; TANENBAUM, 2023, p. 191, 220). Protocolos como o AMQP (Advanced Message Queuing Protocol) são exemplos de middleware orientado a mensagens (MOM) que fornecem esse tipo de comunicação, ideal para cenários onde a conectividade não é garantida (STEEN; TANENBAUM, 2023, p. 227). Essa abordagem, combinada com o armazenamento local dos dados e mecanismos de replicação, constitui a base para um sistema tolerante a falhas de comunicação.

2.5 Protocolos de Comunicação

Um dos princípios fundamentais da Internet das Coisas (IoT) é a comunicação, pois a maior parte do que a torna possível é construída em torno da estrutura de comunicação que interliga os dispositivos (LEA, 2020, p. 89). Para viabilizar essa troca de dados em um sistema distribuído, que integra diferentes camadas como dispositivos, borda e nuvem, é crucial a escolha criteriosa de tecnologias que favoreçam a interoperabilidade.

Neste trabalho, foram empregados protocolos da categoria de *middleware* orientada a mensagens (MOM), onde a comunicação ocorre por meio de filas distribuídas, permitindo que alguns dispositivos produzam dados enquanto outros os consomem (LEA, 2020, p. 388). Dentre eles, destacam-se o MQTT e o AMQP, que foram utilizados em diferentes camadas do sistema para atender a requisitos distintos de comunicação, uma vez que a introdução de camadas intermediárias (como a borda) torna a orquestração do sistema significativamente mais complexa.

2.5.1 MQTT

O protocolo MQTT (*Message Queuing Telemetry Transport*) é amplamente utilizado em sistemas IoT por ser extremamente simples e leve, projetado para dispositivos com recursos restritos e para operar em redes de baixa largura de banda, alta latência ou pouco confiáveisv(LEA, 2020, p. 390). Ele opera sobre TCP/IP, o que lhe confere uma garantia base de que os pacotes são transferidos de forma confiável (LEA, 2020, p. 396).

Seu modelo *publish/subscribe* (pub/sub) desacopla o cliente que transmite a mensagem (publicador) do cliente que a recebe (assinante), sem que eles precisem conhecer os identificadores físicos um do outro, como endereço IP ou porta (LEA, 2020, p. 391). A comunicação é mediada por um servidor central chamado broker (LEA, 2020, p. 388). No entanto, é importante notar que, por padrão, o MQTT não é uma fila de mensagens. Se uma mensagem for publicada em um tópico e não houver assinantes, a mensagem é simplesmente ignorada e perdida (LEA, 2020, p. 392).

No contexto deste projeto, MQTT foi escolhido para a comunicação entre os dispositivos IoT e o servidor de borda, pois sua eficiência e resiliência são adequadas para nós de sensores que operam com comunicação sem fio e são alimentados por bateria.

2.5.2 AMQP

AMQP (*Advanced Message Queuing Protocol*) é um protocolo MOM robusto e comprovado, utilizado principalmente na comunicação entre os componentes de software dos servidores de borda e nuvem (LEA, 2020, p. 426). Diferentemente de MQTT, AMQP possui mecanismos formais de enfileiramento e balanceamento de carga (LEA, 2020, p. 426), sendo projetado para garantir a entrega confiável de mensagens. Ele permite a criação de filas duráveis, que persistem mesmo que o servidor falhe, e pode marcar mensagens como duráveis para garantir que um nó intermediário possa recuperá-las em caso de falha (STEEN; TANENBAUM, 2023, p. 231, 220).

A arquitetura do AMQP é baseada em *exchanges*, que recebem mensagens dos publicadores e as roteiam para as filas corretas por meio de regras chamadas *bindings* (LEA, 2020, p. 426). Essa estrutura permite padrões de roteamento avançados, como o encaminhamento de uma mesma mensagem para múltiplas filas (*fan-out*) (LEA, 2020, p. 426). No sistema desenvolvido, essa capacidade é explorada para a replicação de dados entre a borda e a nuvem, já que AMQP permite que gerenciadores de filas se conectem uns aos outros, formando uma rede de sobreposição para rotear mensagens de forma confiável (STEEN; TANENBAUM, 2023, p. 231). Além disso, o AMQP foi fundamental na orquestração de tarefas assíncronas com Celery. O uso de um middleware orientado a mensagens é ideal para comunicação assíncrona persistente (STEEN; TANENBAUM, 2023, p. 191, 220). Isso permite que o sistema processe operações de forma distribuída e escalável, contribuindo para a resiliência e a tolerância a falhas da arquitetura.

2.6 Replicação

Para garantir a replicação de dados entre borda e nuvem, especialmente em contextos de conectividade intermitente, foi adotado o recurso de *Shovel*, disponível no *broker* RabbitMQ. Esse mecanismo permite a transferência automática de mensagens entre diferentes *brokers*, com suporte à persistência local, retransmissão em caso de falhas e resiliência a interrupções temporárias de rede.

No sistema proposto, o fluxo de replicação pode ocorrer de três formas distintas:

- **Unidirecional da nuvem para a borda**: utilizado para replicar dados originados na nuvem, como entidades relacionadas à configuração dos componentes do sistema e aspectos de sua administração.
- **Unidirecional da borda para a nuvem**: utilizado para replicar dados produzidos localmente na borda, como leituras de sensores e informações de monitoramento dos dispositivos IoT.
- **Bidirecional**: utilizado em entidades que podem ser modificadas por ambas as instâncias, como os objetos de agendamento de irrigação, que podem ser alterados tanto na nuvem (por meio da interface remota) quanto na borda (durante operação desconectada).

Nos casos de replicação **bidirecional**, torna-se necessária a adoção de um **protocolo de consistência eventual**, capaz de assegurar a conciliação correta dos dados replicados após um período de desconexão. Situações desse tipo ocorrem, por exemplo, quando um mesmo agendamento é alterado simultaneamente na borda (em operação local) e na nuvem (via interface remota). Durante a reconexão, essas alterações devem ser comparadas e reconciliadas para preservar a integridade do sistema.

2.7 Protocolo de Consistência Eventual

Para lidar com a replicação bidirecional em cenários de operação desconectada, foi implementado um **protocolo de consistência eventual**, responsável por garantir a integridade dos dados após a reconexão entre borda e nuvem. O protocolo é fundamentado em mecanismos de versionamento de objetos, controle de conflitos, sincronização assíncrona e em uma política de precedência que reconhece o contexto operacional da borda. Cada objeto replicável mantém um identificador único (UUID) e metadados associados, incluindo os apresentados na Tabela 1.

A maioria desses metadados é utilizada para controle interno e registro em logs da aplicação. No entanto, o atributo `versao_borda` é utilizado para indicar a precedência dos dados de agendamento originados na borda em relação àqueles provenientes da nuvem, assegurando que as alterações mais recentes realizadas pela instância do servidor de borda

Tabela 1 – Metadados de controle de replicação por objeto

Campo	Descrição
`uuid`	Identificador único e global do objeto
`versao_borda`	Número de versão gerado na borda
`versao_nuvem`	Número de versão gerado na borda
`versao_uuid`	UUID da versão atual para garantir idempotência
`atualizado_em_borda`	Timestamp da última modificação feita na borda
`atualizado_em_nuvem`	Timestamp da última modificação feita da nuvem
`atualizado_em`	Timestamp geral da última modificação
`origem_ultima_atualizacao`	Indica se a última atualização veio da borda ou da nuvem

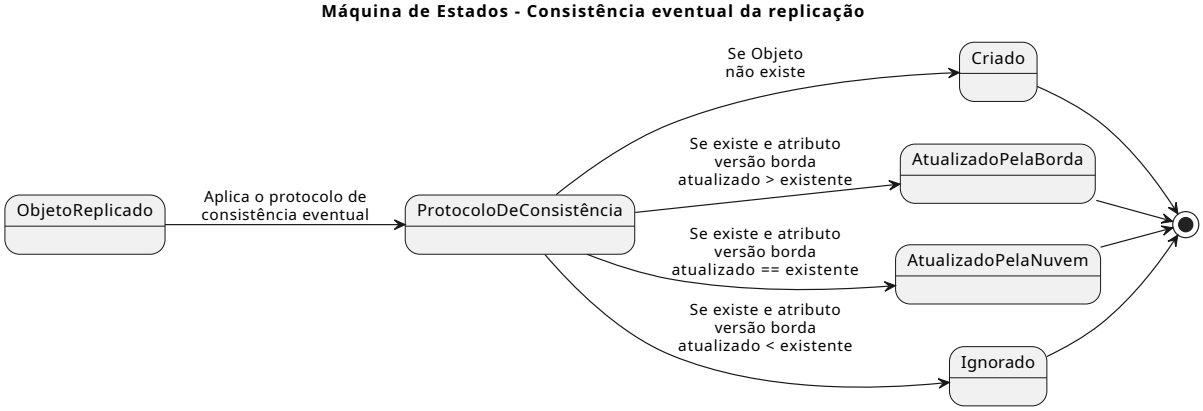

Figura 2 – Máquina de estados de replicação de objetos

sejam preservadas durante o processo de reconciliação. Essa lógica pode ser representada por uma máquina de estados que define quatro possíveis desfechos para a replicação:

- **Criado:** Se o objeto replicado ainda não existe na instância de destino, ele será criado.
- **Atualizado pela borda:** Se o objeto já existe e a `versao_borda` recebida for maior que a armazenada localmente, o objeto será atualizado com os novos dados.
- **Atualizado pela nuvem:** Se as `versao_borda` forem iguais, mas a `versao_nuvem` recebida for superior à versão local, o objeto também será atualizado.
- **Ignorado:** Se as versões recebidas forem iguais ou inferiores às versões locais, nenhuma modificação é aplicada.

Esse mecanismo garante idempotência, evita atualizações desnecessárias e assegura que apenas as modificações mais recentes prevaleçam em situações de replicação assimétrica. A Figura 2 ilustra o fluxo de replicação.

2.8 Segurança em Sistemas IoT

A segurança em sistemas IoT, que constituem a maior superfície de ataque do planeta (LEA, 2020, p. 37, 517, 563), exige uma abordagem holística que envolve múltiplas camadas (LEA, 2020, p. 560). A proteção deve ser considerada desde o início do projeto, abrangendo dispositivos físicos, comunicação, software embarcado e aplicações web (LEA, 2020, p. 517, 560). No sistema proposto, diversas medidas foram adotadas para mitigar riscos, em alinhamento com princípios de segurança consolidados, como a economia de mecanismo, que preza pela simplicidade para reduzir a ocorrência de erros e facilitar a verificação (STEEN; TANENBAUM, 2023, p. 550).

As principais estratégias implementadas foram:

Criptografia TLS nas comunicações MQTT e HTTP(S): TLS é o padrão para comunicação segura em sistemas distribuídos, como em conexões HTTPS (STEEN; TANENBAUM, 2023, p. 551). Toda a comunicação entre os componentes do sistema utiliza criptografia para proteger os dados em trânsito. A adoção de *Transport Layer Security* (TLS) é uma prática fundamental para implementar a confidencialidade e garantir a integridade das informações (STEEN; TANENBAUM, 2023, p. 548). O sistema segue a recomendação de que as soluções IoT só devem ser implantadas com segurança e autenticação habilitadas via TLS (LEA, 2020, p. 562).

Autenticação obrigatória de dispositivos e controle de acesso (ACLs): A arquitetura do sistema é guiada por uma política de segurança que define claramente o que deve ser protegido (STEEN; TANENBAUM, 2023, p. 547). Para implementar essa política, todos os dispositivos devem se autenticar para acessar a rede. O controle de acesso a recursos, como tópicos MQTT, é rigorosamente gerenciado, aplicando o princípio da separação de privilégios para garantir que entidades distintas controlem diferentes aspectos críticos do sistema (STEEN; TANENBAUM, 2023, p. 549). Essa abordagem se alinha ao Controle de Acesso Baseado em Papéis (RBAC), que organiza permissões com base na função do usuário ou dispositivo (STEEN; TANENBAUM, 2023, p. 596).

Uso de UUIDs como identificadores únicos: O sistema utiliza Identificadores Únicos Universais (UUIDs) para a nomeação de entidades, como *gateways* e dispositivos. Essa estratégia, conhecida como *flat naming*, usa identificadores sem significado semântico, mas que são garantidamente únicos (STEEN; TANENBAUM, 2023, p. 326, 329). Nomes planos são ideais para a comunicação entre máquinas, pois simplificam a interoperabilidade e evitam colisões de nomes entre diferentes instâncias.

Isolamento de serviços com Docker: Os serviços do sistema (servidor web, banco de dados, broker de mensagens) são executados em contêineres Docker, que gerenciam a abstração no nível da aplicação (LEA, 2020, p. 348-349). Os contêineres mantêm os processos separados uns dos outros, fornecendo um nível de proteção similar ao de uma máquina virtual, mas de forma mais leve e eficiente em termos de recursos (LEA, 2020, p. 349). Essa abordagem garante que cada processo tenha a visão de que seu contexto é

o único existente, reforçando o isolamento.

Proteções de hardware e firmware: A segurança física e de baixo nível é crucial, pois muitos dispositivos IoT operam em locais remotos e vulneráveis a ataques físicos (LEA, 2020, p. 528). As medidas adotadas incluem.

Proteções físicas (IP65/IP67): Os invólucros dos dispositivos IoT seguem o padrão de Proteção de Ingresso (IP), que qualifica a resistência dos equipamentos contra a infiltração de poeira e água, garantindo sua durabilidade em ambientes não controlados (LEA, 2020, p. 343).

Uso de *eFuses*: Para proteger o firmware dos dispositivos IoT, o sistema utiliza memórias programáveis uma única vez (eFuses) (LEA, 2020, p. 528-529). Essa técnica também é empregada para desativar permanentemente as portas de depuração antes do envio do produto ao campo, dificultando a engenharia reversa (LEA, 2020, p. 562).

Essas práticas, aplicadas em conjunto, formam uma manta de segurança que assegura a confidencialidade, integridade e disponibilidade dos dados e serviços, aumentando a resiliência do sistema contra uma vasta gama de ameaças.

2.9 Metodologia Design Thinking

Design Thinking é uma abordagem de resolução de problemas composta por cinco fases: empatia, definição do problema, ideação, prototipação e teste. Embora tenha sido originalmente aplicada ao design de produtos e serviços, sua flexibilidade permite adaptações para diversos contextos, inclusive no desenvolvimento de sistemas.

Neste trabalho, *Design Thinking* não foi adotado em sua forma canônica, mas serviu como inspiração para orientar o desenvolvimento, organizado em quatro fases que preservam a essência do modelo original, adaptando suas etapas às necessidades do projeto e alinhando-se à característica iterativa e incremental do desenvolvimento de software.

A versão adotada compreende as fases de **Imersão e Definição do Problema**, **Ideação e Prototipação Inicial**, **Desenvolvimento Iterativo** e **Consolidação da Solução**, que se relacionam diretamente com as etapas do modelo original do *Design Thinking*, conforme descrito a seguir:

- Imersão e Definição do Problema, corresponde às fases de *empatia* e *definição*, centrada na compreensão do contexto e das necessidades dos usuários, com base em observações e interações em campo.

- Ideação e Prototipação Inicial, agrupa as fases de *ideação* e *prototipação*, nas quais foram geradas hipóteses arquiteturais e construídos protótipos preliminares para avaliar sua viabilidade técnica.

- Desenvolvimento Iterativo, reflete a continuidade do processo de *prototipação* e aproxima-se da fase de *teste*, por meio de ciclos repetidos de experimentação, refi-

namento técnico e validação parcial dos componentes do sistema.

- Consolidação da Solução, está alinhada à fase de *teste* e à entrega final, caracterizando a definição da arquitetura definitiva com base nos aprendizados acumulados ao longo do processo iterativo.

2.10 Modelagem e Engenharia de Software Experimental

O desenvolvimento foi guiado pelos princípios da Engenharia de Software Experimental, em ciclos iterativos com base em evidências da literatura e testes práticos. A modelagem do sistema utilizou os seguintes artefatos:

- Modelo C4: para representar a arquitetura nos níveis de contexto, contêiner e componentes.
- Diagramas de sequência: para descrever o comportamento dinâmico do sistema.
- Diagramas de máquina de estados: para modelar o ciclo de vida das mensagens e dos objetos sincronizados.

A modelagem foi essencial para garantir rastreabilidade entre requisitos, implementação e testes.

3 REVISÃO SISTEMÁTICA DA LITERATURA

Com os avanços da Internet das Coisas (IoT) e da Agricultura 4.0, tornou-se viável automatizar processos de irrigação por meio de sistemas compostos por sensores, atuadores e plataformas de software. Esses sistemas exigem a integração de componentes adequados de hardware e software, comunicação sem fio compatível com o ambiente agrícola, fontes de alimentação confiáveis e proteção física contra intempéries. Diante desse cenário, foi conduzida uma Revisão Sistemática da Literatura (RSL) com o objetivo de analisar as diferentes abordagens arquiteturais adotadas em sistemas de irrigação inteligente, observando como os desafios técnicos são tratados e quais elementos estruturais são mais utilizados.

A revisão seguiu um protocolo bem definido, com buscas realizadas em três bases de dados. Foram recuperados 8614 estudos, dos quais 25 foram selecionados para análise aprofundada, com base em critérios de inclusão e exclusão. As informações extraídas foram organizadas em cinco categorias: hardware, software, comunicação, alimentação elétrica e invólucro de proteção.

3.1 Definição do Protocolo da Revisão Sistemática de Literatura

Esta Revisão Sistemática da Literatura (RSL) foi conduzida com o objetivo (**O**) de identificar as diversas abordagens arquiteturais utilizadas na estruturação de sistemas de irrigação inteligentes. Para alcançar esse propósito, foram elaboradas as seguintes questões de pesquisa (**Q**):

- **O1**: Analisar a literatura atual com o propósito de identificar a arquitetura de sistemas de irrigação inteligente.
- **Q1**: Qual é o hardware utilizado para o sistema de irrigação?
- **Q2**: Como o software do sistema é estruturado?
- **Q3**: Qual tecnologia de comunicação é utilizada pelo sistema?
- **Q4**: Como o hardware é alimentado eletricamente?
- **Q5**: Como o hardware é protegido das intempéries?

3.1.1 Identificação de Termos e Sinônimos

Para responder às questões de pesquisa levantadas acima, foi produzida uma *string* de busca baseada em sinônimos e termos relacionados a **sistemas de irrigação inteligente**, conforme utilizados na literatura técnica. Essa abordagem visa garantir maior abrangência e precisão na recuperação dos estudos relevantes durante a etapa de busca nas bases de dados, o que inclui:

- *Irrigation Control*
- *Smart Irrigation*
- *Precision Irrigation*
- *Irrigation Automation*
- *Irrigation System Control*
- *Remote Irrigation Control*
- *Irrigation Scheduling*
- *Irrigation Monitoring*
- *Irrigation Control Software*
- *Wireless Irrigation Control*

3.1.2 Strings de Busca

Os termos selecionados foram tratados como sinônimos, sendo combinados com o operador lógico **OR** na confecção da *string* de busca. Essa estratégia teve como objetivo ampliar a abrangência dos resultados e capturar diferentes terminologias utilizadas na literatura técnica.

Após a geração da *string* de busca, ela foi aplicada em três bases de dados eletrônicas amplamente reconhecidas na área de Ciência da Computação e Engenharia:

- *IEEE Xplore (http://ieeexplore.ieee.org)*
- *ACM Digital Library (http://dl.acm.org)*
- *Scopus (http://www.scopus.com/)*

Como cada base de dados possui uma sintaxe própria, a *string* de busca foi devidamente adaptada para atender às particularidades de cada uma delas. Essa adaptação considerou tanto a estrutura sintática específica quanto o uso do operador lógico **OR**, empregado para expressar que os termos utilizados são sinônimos ou semanticamente relacionados. Dessa forma, a mesma lógica de busca foi mantida em todas as bases, respeitando a forma com que cada uma interpreta combinações de termos, garantindo assim a consistência e abrangência dos resultados recuperados.

Tabela 2 – Resultados obtidos na busca inicial

Biblioteca Digital	**Quantidade de estudos encontrados**
IEEE Xplore	810
ACM Digital Library	171
Scopus	7633
Total	**8614**

Fonte: Elaboração própria.

3.1.3 Resultados obtidos

Após a realização da primeira busca em cada uma dessas bases de dados, foram obtidos os resultados[1], conforme apresentados na Tabela 2. Para se filtrar resultados relevantes sobre os dados obtidos na busca sistemática, foram utilizados os seguintes **critérios de inclusão** (CI), com o objetivo de garantir que apenas estudos pertinentes ao escopo desta revisão fossem analisados:

- **CI1**: Artigos publicados nos últimos 9 anos, entre 2014 e 2023, considerando a data de realização da RSL.
- **CI2**: Estudos que detalhem a arquitetura do hardware utilizado
- **CI3**: Artigos publicados em inglês

Após a aplicação dos critérios de inclusão, foram utilizados os **critérios de exclusão** (CE) para refinar a seleção dos artigos considerados relevantes:

- **CE1**: Publicações duplicadas
- **CE2**: Artigos sem acesso ao texto completo
- **CE3**: Artigos sem descrição da arquitetura de um sistema de irrigação inteligente

Após a aplicação dos critérios de exclusão, foram adotados os **critérios de qualidade** (CQ) com o objetivo de classificar e filtrar, de forma qualitativa, os artigos mais relevantes para este estudo:

- **CQ1**: Estudos que apresentem os softwares utilizados
- **CQ2**: Estudos que descrevam a tecnologia de comunicação utilizada
- **CQ3**: Estudos que informem a fonte de alimentação utilizada no hardware
- **CQ4**: Estudos que utilizem alguma forma de proteção do hardware das intempéries

Após a aplicação de todos os critérios, foram selecionados 25 artigos que demonstram alta qualidade e relevância para o estudo proposto, conforme listados na Tabela 3.

[1] Busca realizada em 20/11/2023.

Tabela 3 – Tabela de Artigos Selecionados

ID	Artigo (Título)	Base
1	Renewable Energy Integration Into Cloud & IoT-Based Smart Agriculture (BOUALI et al., 2022)	IEEE Xplore
2	Automated Irrigation System Using a Wireless Sensor Network and GPRS Module (GUTIéRREZ et al., 2014)	IEEE Xplore
3	Prototype Model Design of Automatic Irrigation Controller (POYEN et al., 2021)	IEEE Xplore
4	Cloud-based Internet of Things Approach for Smart Irrigation System: Design and Implementation (THAHER; ISHAQ, 2020)	IEEE Xplore
5	Implementation of IoT in Agriculture: A Scientific Approach for Smart Irrigation (V et al., 2022)	IEEE Xplore
6	IoT Based Power Efficient Agro Field Monitoring and Irrigation Control System: An Empirical Implementation in Precision Agriculture (ISLAM et al., 2018)	IEEE Xplore
7	Design and Implementation of Smart Irrigation System for Groundwater Use at Farm Scale (ZAIER et al., 2015)	IEEE Xplore
8	Gateway, Cloud Server, and Mobile App Dashboard in Evapotranspiration-Based Irrigation Scheduling System (HERELL et al., 2021)	IEEE Xplore
9	Electro-Mechanical Design of Sensor-Hub for Indoor Smart Irrigation (P et al., 2022)	IEEE Xplore
10	Automated Irrigation Scheduling for Different Crop using IoT (PATEL et al., 2022)	IEEE Xplore
11	Automated Water-Gate Controlling System for Paddy Fields (SANJULA et al., 2020)	IEEE Xplore
12	Smart Irrigation System: A Water and Power Management Approach (EDODI; OGIDAN; AMUSAN, 2022)	IEEE Xplore
13	Automated Irrigation Management System using IoT (KANUMALLI et al., 2022)	IEEE Xplore
14	Securing IoT Smart Irrigation Systems with Adapted Blockchain Based Approach (BENAMEUR et al., 2023)	IEEE Xplore
15	Low Cost Center Pivot Irrigation Monitoring Systems Based on IoT and LoRaWAN Technologies (MATILLA et al., 2020)	IEEE Xplore
16	Agro Smart: IoT Autonomous Irrigation System (SANTOS et al., 2022)	IEEE Xplore
17	IoT and Cloud Hinged Smart Irrigation System for Urban and Rural Farmers Employing MQTT Protocol (K.; VIJAYARAGHAVAN, 2020)	IEEE Xplore
18	An Intelligent Irrigation Scheduling System Using Low-Cost Wireless Sensor Network Toward Sustainable and Precision Agriculture (JAMROEN et al., 2020)	IEEE Xplore
19	A Fuzzy Irrigation Control System (SOUZA et al., 2020)	IEEE Xplore
20	Smart Irrigation System using IoT-based Control Valve (MUJOO et al., 2021)	IEEE Xplore
21	Water Saving in Agriculture through the Use of Smart Irrigation System (NAJI; SALMAN, 2021)	ACM Digital Library
22	Low Power IoT Electronics in Precision Irrigation (ROUTIS; ROUSSAKI, 2023)	Scopus
23	Arduino Based Irrigation Monitoring System Using Node Microcontroller Unit and Blynk Application (SHAHAR et al., 2023)	Scopus
24	Smart Irrigation System (ABDIKADIR et al., 2023)	Scopus
25	Experimental Performance of Smart IoT-Enabled Drip Irrigation System Using and Controlled Through Web-Based Applications (JAIN, 2023)	Scopus

Fonte: Elaboração própria.

3.2 Resultados e Discussão da Revisão Sistemática de Literatura

A partir da análise dos artigos selecionados (Tabela 3), foram elaboradas tabelas que sumarizam os elementos que compõem a arquitetura dos sistemas de irrigação inteligente estudados. Nas seções a seguir, essas informações são apresentadas e discutidas.

3.2.1 Arquitetura do Hardware

O objetivo desta seção é responder à questão **Q1**, relacionada à composição da arquitetura de hardware dos sistemas de irrigação inteligente. Com base na análise dos artigos selecionados, os elementos de hardware foram organizados nas seguintes categorias:

- Hardware de processamento - Microprocessadores e Microcontroladores
- Hardware de Sensores
- Hardware de Atuadores

3.2.1.1 Hardware de processamento

Todo sistema de Internet das Coisas (IoT) deve possuir capacidade de processamento e conectividade com a internet (LEA, 2020, p. 10). Para isso, é necessário o uso de hardware com capacidade computacional, geralmente implementado por meio de sistemas baseados em microprocessadores, comumente um *System-on-a-Chip* (SoC), ou por sistemas baseados em microcontroladores (MCU) (LEA, 2020, p. 315). Além da capacidade de processamento, esses dispositivos devem ser capazes de se comunicar, normalmente por meio de tecnologias sem fio (LEA, 2020, p. 89). Essa funcionalidade pode estar integrada ao próprio dispositivo (como Wi-Fi ou Bluetooth) ou ser adicionada por meio de módulos ou placas de expansão.

Durante a análise dos artigos, foi possível observar que parte dos dispositivos possuía comunicação sem fio integrada, enquanto outros utilizavam módulos específicos compatíveis com os padrões discutidos na seção sobre sistemas de comunicação. Na maioria dos estudos analisados, identificou-se uma divisão dos nós de processamento em duas classes principais:

- **Nós de maior capacidade de processamento**: frequentemente com a função de coordenar os nós sensores e atuadores, além de intermediar a comunicação com sistemas hospedados na nuvem. Esses nós são geralmente denominados *gateways* (LEA, 2020, p. 312).
- **Nós de menor capacidade de processamento**: responsáveis por tarefas específicas como a coleta de dados (sensores) ou o acionamento de dispositivos (atuadores).

Tabela 4 – Hardware de Processamento

Total	Hardware de Processamento	Artigos
8	Raspberry Pi	1, 4, 8, 14, 17, 19, 22, 24
5	ESP32	3, 5, 8, 11, 15
9	ESP8266	1, 6, 10, 12, 13, 17, 20, 23, 25
5	Arduino Nano	1, 3, 6, 9, 23
5	Arduino Uno	4, 14, 18, 20, 21
3	Arduino Mega	12, 14, 22
1	Arduino Due	16
1	Arduino Yún	25
1	PIC24FJ64GB004	2

Fonte: Elaboração própria.

Na maior parte dos casos, os *gateways* eram compostos por sistemas baseados em microprocessadores, enquanto os nós sensores e atuadores utilizavam microcontroladores, embora também tenham sido identificadas exceções a essa configuração nos sistemas analisados. As informações sobre o hardware de processamento estão sumarizadas na Tabela 4.

3.2.1.2 Sensores

Os dispositivos IoT utilizam sensores para captar informações sobre o ambiente e o estado do sistema. Diversos tipos de sensores foram utilizados nos diferentes artigos analisados. Os sensores podem ser categorizados, de maneira geral, nas seguintes classes:

- Umidade e Temperatura do Ar
- Temperatura
- Solo
- Ambientais
- Equipamentos

Sensores de Umidade e Temperatura do Ar: Os sensores de umidade e temperatura do ar são, em sua totalidade, compostos por dispositivos da família *DHT*, que realizam a medição simultânea de umidade relativa do ar e temperatura ambiente em um único componente eletrônico. Essa família inclui modelos como o DHT11 e o DHT22, que se diferenciam principalmente pela faixa de operação e pela precisão dos dados coletados.

As características dos sensores de umidade e temperatura do ar utilizados nos estudos estão apresentadas na Tabela 5. Uma observação importante é que alguns dos estudos analisados não especificam qual modelo foi utilizado, indicando apenas a presença

Tabela 5 – Sensores de Umidade e Temperatura do Ar

Total	**Umidade e Temperatura do Ar**	**Artigos**
9	DHT11	1, 5, 6, 9, 10, 11, 13, 17, 24
8	DHT22	3, 4, 14, 16, 18, 19, 22, 25
1	Sensor não especificado	5

Fonte: Elaboração própria.

Tabela 6 – Sensores de Temperatura

Total	**Sensor de Temperatura**	**Artigos**
1	Temperatura DS1822	2
1	Infravermelho MLX90614 GY-906	18
1	Sensor não especificado	5

Fonte: Elaboração própria.

de um sensor de umidade e temperatura do ar. Essa ausência de detalhamento pode impactar na reprodutibilidade e comparação de resultados entre os sistemas analisados.

Sensores de Temperatura: Alguns artigos optaram por utilizar sensores dedicados exclusivamente à medição de temperatura do ar, em vez de sensores combinados como os da família *DHT*. Novamente foi identificado ao menos um estudo (V et al., 2022) que menciona o uso de sensor de temperatura, porém sem especificar o modelo ou a tecnologia empregada. Os sensores dedicados à medição de temperatura do ar utilizados nos estudos estão listados na Tabela 6.

Sensores de Solo: Diversos sensores foram utilizados especificamente para mensurar características do solo, sendo a maioria voltada à medição da umidade. Os sensores variam conforme o modelo e a tecnologia empregada. Sensores de umidade do tipo resistivo, quando não confeccionados com materiais inoxidáveis, tendem a oxidar com o tempo, o que pode comprometer sua durabilidade e precisão.

Por essa razão, diversos autores optaram por sensores de umidade do solo do tipo capacitivo, ou por sensores confeccionados com materiais mais resistentes à oxidação. É importante destacar que alguns artigos incluíram sensores de umidade do solo, mas não especificaram claramente o modelo ou a tecnologia utilizada. Isso ocorreu nos artigos (V et al., 2022; ISLAM et al., 2018; PATEL et al., 2022; SANJULA et al., 2020; KANUMALLI et al., 2022; BENAMEUR et al., 2023; K.; VIJAYARAGHAVAN, 2020; SHAHAR et al., 2023), o que limita uma análise mais precisa sobre a confiabilidade e o desempenho desses componentes.

Além disso, foi identificada a utilização de sensores dedicados à medição de outras propriedades do solo. Em três artigos, foram utilizados sensores de temperatura do tipo DS18B20, que permitem aferir a temperatura diretamente no solo. Adicionalmente, um dos estudos empregou um sensor de pH do solo (modelo SEN0249), ampliando as variáveis

Tabela 7 – Sensores de Solo

Total	**Sensor de Solo**	**Artigos**
3	Umidade do Solo VH400 (capacitivo)	1, 2, 7
2	Umidade do Solo SEN0193 (capacitivo)	16, 18
3	Umidade do Solo CSM (capacitivo)	19, 22, 24
3	Umidade do Solo YL-69 (resistivo)	3, 4, 21
1	Umidade do Solo FC-28 (resistivo)	25
1	Umidade do Solo LM393 (dielétrico)	9
8	Umidade do Solo sensor não especificado	5, 6, 10, 11, 13, 14, 17, 23
3	Temperatura do Solo DS18B20	3, 6, 23
1	pH do Solo SEN0249	3

Fonte: Elaboração própria.

Tabela 8 – Sensores Ambientais

Total	**Sensor Ambiental**	**Artigos**
2	Sensor de Chama Infravermelho	1, 11
2	Sensor LDR Flying-Fish (MH series)	14, 16
2	Sensor de Luz BH1750FVI	3, 18
1	Sensor de Luz UV VEML6070	22
1	Piranômetro BGT-JYZ2	18
2	Sensor de Movimento PIR	1, 16
1	Sensor de Gás MQ2 (Inflamável e Fumaça)	14
1	Sensor de Gás MQ-7 (Monóxido de Carbono)	16
2	Sensor de Nível de Água	11, 14
2	Sensor de Chuva YL-83	11, 23

Fonte: Elaboração própria.

monitoradas no ambiente agrícola. Os sensores utilizados para mensurar características do solo estão listados na Tabela 7.

Sensores de Ambientais: Outra categoria de sensores (diversa) são os sensores nomeados aqui como *sensores ambientais*, que foram agrupados dessa forma pela característica de monitoração do ambiente. Esses sensores incluem dispositivos responsáveis por capturar informações relacionadas à luminosidade, como sensores de chama, luz ultravioleta e intensidade luminosa solar, bem como sensores de movimento. Além disso, alguns artigos empregaram sensores de gases, capazes de detectar fumaça ou presença de monóxido de carbono, úteis para identificar riscos de incêndio. Também foram utilizados sensores de água e chuva, que permitem monitorar o nível de umidade ambiental e a ocorrência de precipitação, contribuindo para decisões automáticas do sistema de irrigação.

A Tabela 8 apresenta os sensores utilizados para o monitoramento de variáveis ambientais.

Sensores de Equipamentos: Esta categoria foi agrupada pois se trata de sensores utilizados para monitorar informações relativas aos equipamentos e ao seu funcio-

Tabela 9 – Sensores de Equipamentos

Total	Sensor de Equipamento	Artigos
4	Sensor de Distância Ultrassônico HC-SR04	1, 3, 12, 23
3	Medidor de Vazão	3, 14, 19
3	Sensor de Tensão e Corrente ACS712	1, 3, 12

Fonte: Elaboração própria.

Tabela 10 – Atuadores

Total	Atuador	Artigos
11	Relé (genérico)	1, 2, 5, 10, 12, 13, 14, 16, 17, 21, 24
1	Relé Sonoff (ESP8266)	1
12	Bomba de Água	2, 3, 5, 6, 12, 13, 14, 18, 19, 23, 24, 25
2	Driver de Motor L298N	11, 20
10	Válvula Solenóide	3, 4, 6, 7, 9, 10, 16, 19, 21, 25

Fonte: Elaboração própria.

namento, como a vazão da água, a tensão e corrente elétrica, além da distância. Este último geralmente é utilizado para mensurar o nível da coluna de água em reservatórios utilizados pelas bombas. Esses sensores desempenham papel fundamental na manutenção da eficiência e segurança do sistema de irrigação, permitindo detectar falhas operacionais e garantir o funcionamento adequado dos dispositivos atuadores. Os sensores utilizados para monitorar os equipamentos e seu funcionamento estão listados na Tabela 9.

Atuadores: O objetivo dos dispositivos IoT normalmente envolve o controle de algum equipamento. Sendo assim, além dos sensores para monitoramento do ambiente e dos nós com capacidade de processamento para interpretar os dados, é necessário o uso de atuadores, responsáveis por executar ações que modifiquem o ambiente.

No caso dos sistemas de irrigação inteligente, os atuadores identificados nos artigos analisados podem ser categorizados em três grupos principais: relés (interruptores elétricos controláveis), *drivers* ou controladores de motor, e dispositivos de controle de fluxo de água. Estes últimos incluem bombas d'água, que atuam ativamente movimentando a água por meio de bombeamento, e válvulas solenóides, que desempenham um papel mais passivo ao abrir ou fechar o fluxo de água em encanamentos. A Tabela 10 apresenta os atuadores utilizados nos sistemas de irrigação inteligente analisados.

3.2.2 Arquitetura do Software

O objetivo desta seção é responder à questão **Q2**, que trata sobre como o software do sistema de irrigação inteligente é estruturado. Após a análise dos artigos selecionados, foi possível classificar os elementos de software em diferentes categorias, conforme suas funções e papéis na arquitetura do sistema:

- Protocolos/Tecnologias de comunicação
- Plataformas de programação de IoT
- Linguagens e ferramentas de programação
- Banco de dados
- Softwares de sistema
- Outros softwares

Diferentemente da classificação anterior voltada ao hardware, esta classificação de elementos de software apresenta um grau notoriamente maior de ambiguidade. Essa ambiguidade decorre do fato de que a categorização tem como principal função orientar o leitor, ao invés de estabelecer uma taxonomia precisa dos softwares utilizados nos sistemas analisados. Ainda assim, é possível perceber que uma ampla variedade de linguagens, frameworks, bibliotecas e sistemas operacionais foram empregados na construção das soluções de irrigação inteligente. Essa diversidade tecnológica evidencia que não há, até o momento, uma padronização consolidada na área.

Apesar disso, observa-se um padrão recorrente em termos de usabilidade: a maioria dos sistemas implementa uma interface de monitoramento e controle acessível via aplicativos móveis nativos ou aplicações web. Uma das poucas exceções identificadas corresponde a um sistema cuja interação ocorre exclusivamente por meio de controles locais implementados em hardware, sem interface remota. A Tabela 11 apresenta os softwares, protocolos e tecnologias utilizados nos sistemas de irrigação inteligente analisados.

3.2.3 Arquitetura do Sistema de Comunicação

O objetivo desta seção é responder à questão **Q3**, que trata da tecnologia de comunicação utilizada pelos sistemas de irrigação inteligente.

Sistemas IoT pressupõem a capacidade de comunicação entre dispositivos e a internet (LEA, 2020, p. 89). No entanto, como discutido anteriormente, a maioria das arquiteturas analisadas neste estudo adota um modelo em que um dispositivo atua como concentrador local, comumente denominado *gateway*, que se encontra fisicamente separado dos nós sensores e atuadores. Dessa forma, para que os dispositivos de campo se comuniquem com o *gateway*, é necessário o uso de um sistema de comunicação local, geralmente sem fio.

Alguns dos dispositivos processadores analisados, como o ESP32 e o Raspberry Pi, já possuem conectividade sem fio nativa (Wi-Fi ou Bluetooth), sendo suficientes para essa função. No entanto, outras tecnologias de comunicação sem fio são frequentemente preferidas no contexto agrícola, pois oferecem vantagens como maior alcance, menor consumo de energia e maior robustez em ambientes abertos e sujeitos a interferências naturais.

Tabela 11 – Arquitetura do Software

Total	**Software**	**Artigos**
5	MQTT	1, 5, 13, 17, 24
4	REST API	6, 14, 15, 19
2	Node-RED	1, 24
3	Blynk	12, 13, 23
4	ThingSpeak	3, 4, 16, 25
2	Firebase	11, 16
3	Python	4, 19, 21
1	C++	4
1	C#	2
2	JavaScript	1, 7
1	Web2py	6
1	Express.js	14
2	Node.js	1, 14
1	SQL Server	2
2	MySQL	6, 19
5	Servidor Web (genérico)	2, 7, 8, 15, 25
1	NGINX	14
1	HTML	7
5	Aplicativo Android	6, 8, 16, 17, 25
3	Raspbian OS	1, 19, 22
1	Docker	14
1	Telegram	15
1	MATLAB	16

Fonte: Elaboração própria.

Para viabilizar essas tecnologias em dispositivos que não possuem conectividade embutida, como os microcontroladores Arduino em diversas de suas versões analisadas, torna-se necessário o uso de módulos de comunicação adicionais. Exemplo disso é o uso de transceptores na faixa de 2,4GHz, como o módulo *nRF24L01*, empregado em dois dos estudos analisados (BENAMEUR et al., 2023; MATILLA et al., 2020). Um dos artigos também descreve o uso de comunicação com fio baseada no protocolo RS485. Dentre as tecnologias sem fio destacadas, os padrões ZigBee e LoRa merecem atenção especial, pois além de oferecerem maior alcance e confiabilidade em campo, também permitem a formação de redes em malha (*mesh networks*), o que é altamente adequado para áreas rurais com cobertura instável.

Além da comunicação local entre os nós e o *gateway*, é necessária uma conexão entre o *gateway* e a internet para permitir o controle remoto e a sincronização dos dados com a nuvem. Nem todos os artigos analisados descrevem detalhadamente essa etapa, mas entre os que o fizeram, observaram-se duas abordagens principais: o uso de Wi-Fi e o uso de redes de telefonia celular (2G, 3G ou 4G). Essa divisão é coerente com a realidade rural, onde o tráfego de dados tende a ser baixo, mas a conectividade pode

Tabela 12 – Arquitetura do Sistema de Comunicação

Total	Tecnologia de Comunicação	Artigos
7	ZigBee / XBee	1, 2, 4, 7, 9, 21, 22
6	LoRa	6, 8, 13, 15, 16, 20
9	Wi-Fi	1, 3, 6, 12, 17, 19, 20, 23, 25
2	Rádio 2.4GHz (genérico)	14, 18
4	GPRS / GSM	2, 3, 10, 15
2	4G LTE	5, 8

Fonte: Elaboração própria.

ser intermitente ou inexistente. Nessas situações, o projeto do sistema deve considerar soluções alternativas para garantir a comunicação com a nuvem, como operação offline ou sincronização posterior. A Tabela 12 apresenta os diferentes sistemas de comunicação utilizados nos artigos analisados.

3.2.4 Arquitetura do Sistema de Alimentação Elétrica

O objetivo desta seção é responder à questão **Q4**, que trata da forma como os sistemas de irrigação inteligente são alimentados eletricamente.

A alimentação elétrica é um fator crítico em qualquer sistema eletrônico, e sua importância é ainda maior quando se trata de sistemas implantados em áreas rurais. Esses locais frequentemente apresentam infraestrutura elétrica precária, sujeita a falhas de abastecimento, oscilações de tensão e longos períodos de instabilidade. Apesar de sua importância, esse aspecto nem sempre recebe a devida atenção no momento da definição da arquitetura do sistema.

Para contornar esse desafio, diversos artigos propuseram soluções que visam garantir a operação contínua dos sistemas, mesmo em cenários adversos. Entre as abordagens mais comuns, destacam-se o uso de painéis fotovoltaicos e a adoção de baterias recarregáveis. A energia solar, por sua disponibilidade e viabilidade técnica no campo, aparece como uma solução recorrente e eficaz.

Foi observada uma grande diversidade de tipos de baterias empregadas, incluindo desde as tradicionais baterias de íon de lítio, amplamente utilizadas em projetos de sistemas embarcados, até baterias alcalinas comuns e baterias recarregáveis de níquel-hidreto metálico (Ni-MH). Também foram identificadas variações significativas na tensão de operação, geralmente entre 5V e 12V, o que reflete as diferentes exigências dos dispositivos utilizados em cada sistema. Essa diversidade de soluções de alimentação elétrica mostra que ainda não há um padrão amplamente consolidado, e que as escolhas feitas pelos autores dependem fortemente do contexto do projeto e dos recursos disponíveis. A Tabela 13 apresenta os diferentes sistemas de alimentação elétrica utilizados nos artigos analisados.

Tabela 13 – Arquitetura do Sistema de Alimentação Elétrica

Total	**Tecnologia de Alimentação Elétrica**	**Artigos**
9	Painel Fotovoltaico	1, 2, 6, 7, 8, 11, 12, 14, 20
4	Controlador de Carga Solar	2, 6, 8, 12
1	Carregador de Bateria	20
6	Fonte de Alimentação (5V a 12V)	3, 6, 9, 14, 22, 25
5	Bateria Li-ion (Íon de Lítio)	1, 7, 8, 9, 14
1	Bateria LiPo (Polímero de Lítio)	20
1	Bateria de Chumbo-ácido	2
1	Bateria Alcalina 9V	4
1	Bateria Ni-MH (Hidreto Metálico de Níquel)	2
2	Conjunto de Pilhas	6, 22
5	Bateria (5V a 12V, não especificada)	4, 5, 6, 10, 12

Fonte: Elaboração própria.

3.2.5 Arquitetura do Sistema de proteção contra as Intempéries

O objetivo desta seção é responder à questão **Q5**, que trata da forma como o hardware dos sistemas de irrigação inteligente é protegido das intempéries.

Todo sistema eletrônico deve possuir algum nível de isolamento contra fatores ambientais, como umidade, poeira, calor excessivo e exposição direta à radiação solar (LEA, 2020, p. 343). Esses fatores podem não apenas comprometer o funcionamento do sistema, resultando em falhas operacionais e prejuízos econômicos, como também representar riscos à segurança, incluindo curtos-circuitos, superaquecimento de baterias e, em casos extremos, acidentes de natureza elétrica (LEA, 2020, p. 83, 343).

Assim como ocorre com a questão da alimentação elétrica, os mecanismos de proteção contra intempéries devem zelar pela integridade dos equipamentos e das pessoas envolvidas na operação (LEA, 2020, p. 83, 344). No entanto, constatou-se que esta foi a dimensão menos abordada e menos detalhada nos artigos analisados. Em muitos casos, sequer há menção direta ao sistema de proteção, sendo possível inferir sua presença apenas por meio de imagens ou descrições indiretas.

Quando mencionados, os métodos de proteção variam entre a utilização de caixas comerciais prontas, com grau de proteção IP65 ou superior (LEA, 2020, p. 343, 345), e soluções personalizadas, como a construção de invólucros por impressão 3D ou adaptações artesanais com painéis de acrílico. Esse ponto de fragilidade nas arquiteturas propostas indica uma lacuna relevante na literatura atual. O estudo e desenvolvimento de soluções mais robustas e padronizadas para proteção ambiental do hardware merecem atenção futura, especialmente em cenários de uso prolongado e exposição intensa, como é o caso de ambientes agrícolas em campo aberto. A Tabela 14 apresenta os diferentes sistemas de proteção contra as intempéries utilizados nos artigos analisados.

Tabela 14 – Arquitetura do Sistema de Proteção contra as Intempéries

Total	**Tecnologia de Proteção**	**Artigos**
2	Caixa de PVC impermeável	2, 3
3	Caixa de junção IP65	7, 19, 22
2	Invólucro em PLA impresso em 3D com painéis de acrílico	8, 9
2	Caixa estanque (não especificada)	1, 18
2	Invólucro personalizado	14, 18

Fonte: Elaboração própria.

3.3 Síntese dos Resultados da Revisão

Na literatura científica, existem diversos estudos propondo sistemas de irrigação inteligentes. No entanto, muitos desses trabalhos carecem de um detalhamento adequado da arquitetura de hardware e software e dos sistemas de comunicação utilizados. Mais raros ainda são os estudos que apresentam uma definição clara da alimentação elétrica e das estratégias de proteção do hardware contra as intempéries. Assim, pesquisas de alta qualidade que abordem de forma integrada e detalhada todos esses aspectos são fundamentais para oferecer um direcionamento sólido a pesquisadores e desenvolvedores da área.

Esta revisão sistemática de literatura teve como objetivo identificar práticas arquiteturais adotadas em sistemas de irrigação inteligentes, por meio da análise de estudos com qualidade comprovada. Para isso, elaborou-se um protocolo replicável, com questões de pesquisa bem definidas, estratégias de busca estruturadas e critérios claros de inclusão, exclusão e avaliação qualitativa. A partir da busca realizada em bases digitais, foram recuperados 8.614 estudos. Após a aplicação rigorosa dos critérios definidos, esse número foi reduzido a 25 artigos, considerados relevantes e de alta qualidade, utilizados para responder às questões propostas.

No que se refere ao hardware, a análise revelou a predominância de plataformas de processamento baseadas em *Raspberry Pi*, *ESP32*, *ESP8266* e microcontroladores *Arduino*. Quanto aos sensores, embora uma grande diversidade tenha sido observada, destacaram-se os sensores de temperatura e umidade do ar, além dos sensores de umidade do solo. Os atuadores mais frequentes incluem bombas d'água, válvulas solenóides e relés.

A arquitetura de software, por sua vez, mostrou-se mais heterogênea e, em geral, menos detalhada do que a de hardware. Notou-se o uso recorrente de protocolos como MQTT e REST API para a comunicação entre os dispositivos e a nuvem, evidenciando uma tendência ao uso de soluções leves e flexíveis.

Em relação à comunicação sem fio, os estudos utilizaram uma variedade de tecnologias, com destaque para ZigBee, LoRa, Wi-Fi e redes móveis (2G a 4G). Tais escolhas refletem a necessidade de adaptar os sistemas às limitações e características do meio rural, como alcance do sinal, consumo energético e disponibilidade de infraestrutura.

No tocante à alimentação elétrica, muitos estudos propuseram o uso de painéis fotovoltaicos associados a diferentes tipos de baterias. Observou-se uma ampla variedade de tecnologias de armazenamento de energia, incluindo baterias de íon de lítio, alcalinas, de chumbo-ácido e de níquel-hidreto metálico, além de fontes com tensões variadas. Essa diversidade reflete as diferentes demandas energéticas dos dispositivos e a necessidade de adaptar a fonte de energia ao contexto de uso rural.

Por fim, constatou-se que o sistema de proteção contra intempéries foi o componente menos abordado nos artigos analisados. Além de ser raramente detalhado, em muitos casos sua presença foi apenas inferida por meio de imagens ou menções indiretas. Este aspecto representa uma fragilidade comum nos sistemas propostos e evidencia uma lacuna na literatura, que demanda investigação mais aprofundada em trabalhos futuros.

Portanto, esta revisão fornece um panorama abrangente e crítico das práticas arquiteturais adotadas no desenvolvimento de sistemas de irrigação inteligente baseados em IoT, contribuindo para o avanço do conhecimento técnico-científico na área e servindo como referência sólida para o desenvolvimento deste projeto.

4 TRABALHOS RELACIONADOS

Este capítulo apresenta uma revisão crítica dos trabalhos relacionados à pesquisa desta dissertação, com foco em sistemas de irrigação inteligente baseados em IoT e arquiteturas distribuídas que combinam nuvem e borda. A análise está organizada em três eixos: (i) arquiteturas centradas em nuvem, (ii) modelos híbridos nuvem–borda e nuvem–névoa–borda, que integram processamento local e remoto; e (iii) plataformas de baixo custo. Em cada eixo, sintetizamos objetivos, abordagens técnicas, evidências empíricas e limitações, destacando as lacunas que motivam esta pesquisa.

4.1 Análise dos Trabalhos Relacionados

4.1.1 Arquiteturas Centradas em Nuvem

Plataformas em nuvem viabilizam a coleta, o armazenamento e a análise de dados em grande escala, oferecendo ferramentas consolidadas para visualização e integração. A *Smart Water Management Platform* (SWAMP) (KAMIENSKI et al., 2019) é um exemplo de arquitetura que adota o FIWARE, um ecossistema aberto financiado pela União Europeia, composto por módulos de software denominados *Generic Enablers*. Esses módulos fornecem funcionalidades essenciais para aplicações em IoT, como gerenciamento de contexto, integração de sensores e atuadores, armazenamento histórico de dados e processamento em larga escala. No SWAMP, o FIWARE desempenha o papel de camada intermediária entre dispositivos de campo e aplicações analíticas, assegurando interoperabilidade, escalabilidade e suporte a decisões em tempo real na irrigação de precisão. Além disso, integra protocolos IoT e componentes de persistência para apoiar a operação em cenários reais, ao mesmo tempo em que evidencia desafios de desempenho e escalabilidade nos módulos de contexto e banco de dados (KAMIENSKI et al., 2019). Embora a centralização em nuvem favoreça aspectos como governança e elasticidade, a forte dependência de conectividade estável representa uma limitação relevante em áreas rurais sujeitas à intermitência de rede, condição típica enfrentada por pequenos produtores.

4.1.2 Modelos Híbridos nuvem–borda e nuvem–névoa–borda

Arquiteturas híbridas buscam conciliar os recursos de processamento e armazenamento em nuvem com a tomada de decisão próxima à fonte dos dados, oferecendo tolerância a falhas e menor latência. Alharbi e Aldossary (ALHARBI; ALDOSSARY, 2021) propõem uma arquitetura nuvem–névoa–borda energeticamente eficiente para agricultura inteligente, demonstrando reduções de até 36% no consumo de energia, 43% nas emissões de carbono e 86% no tráfego de rede por meio da alocação hierárquica de tarefas e do uso de comunicação de baixo consumo. Em um cenário de implantação real, Et-Taibi et al. (ET-TAIBI et al., 2024) descrevem uma solução nuvem–borda aplicada a sistemas de irrigação por gotejamento sem solo, na qual sensores heterogêneos são integrados a

fluxos de processamento distribuídos. Os resultados apontam ganhos na otimização do uso de água e energia.

Voltado a cenários de longo alcance e baixo consumo energético, Zhang et al. (ZHANG et al., 2025) apresentam um sistema baseado em LoRa e computação de borda que incorpora uma variante do algoritmo de Penman–Monteith para cálculo da evapotranspiração, suporte a múltiplos protocolos via gateway e comunicação MQTT/WSS. Os testes de campo evidenciaram alcance de até 4 km com baixas perdas de pacotes, assegurando confiabilidade na transmissão de dados. De forma complementar, Kasera e Acharjee (KASERA; ACHARJEE, 2024) propõem um *framework* de irrigação inteligente apoiado em LoRa e IoT de borda, integrando um esquema de transmissão robusta sobre MQTT, topologias eficientes para comunicação M2M e elementos de aprendizado por reforço para agendamento dinâmico da irrigação.

Abordagens em borda também se destacam de forma isolada ao reduzir latência, filtrar dados próximos à fonte e permitir maior autonomia decisória local. Munir et al. (MUNIR et al., 2021) propõem um sistema que combina computação de borda, ontologias e aprendizado de máquina para classificar a necessidade de água a partir de múltiplas variáveis ambientais, evitando o envio de dados desnecessários ao backend. Em linha semelhante, Ndunagu et al. (NDUNAGU et al., 2022) descrevem uma rede de sensores sem fio integrada a uma plataforma em nuvem, mas com pré-processamento no servidor de borda em janelas de tempo, empregando algoritmos de ML (p.ex., KNN) e alcançando altas acurácias em decisões de irrigação. Esses trabalhos reforçam os ganhos da borda em termos de autonomia e eficiência, embora ainda apresentem limitações na definição de mecanismos robustos de sincronização com a nuvem e de políticas confiáveis de operação desconectada.

Em conjunto, esses estudos demonstram benefícios concretos das arquiteturas híbridas, como eficiência energética, redução de tráfego, cobertura ampliada e maior autonomia na tomada de decisão. Contudo, observa-se que poucos trabalhos detalham protocolos práticos de replicação bidirecional e mecanismos de consistência eventual entre nuvem e borda em situações de desconexão prolongada, um desafio recorrente em áreas rurais.

4.1.3 Plataformas de Baixo Custo

Adoção prática em pequenas propriedades exige custo competitivo, interoperabilidade e facilidade de manutenção. (PUIG; DíAZ; SORIANO, 2022) desenvolvem uma plataforma de baixo custo e código aberto que integra FIWARE (LD), modelos de balanço hídrico e um *datalogger* compatível com múltiplos protocolos (LPWAN, LoRaWAN, NB-IoT, GSM etc.), favorecendo reutilização e integração. Complementarmente, (MORCHID et al., 2024) propõem um sistema com ESP32, MQTT e serviços em nuvem, relatando reduções expressivas no consumo de água e descrevendo procedimentos de calibração de sensores e processo de desenvolvimento estruturado. Embora esses trabalhos avancem

interoperabilidade e custo total de propriedade, a maioria carece de políticas formais de *failover*, reconciliação pós-falha e replicação consistente entre camadas, pontos críticos para ambientes com energia e conectividade instáveis.

4.2 Avaliação dos Trabalhos Relacionados

A revisão dos trabalhos relacionados evidencia que os sistemas de irrigação inteligente baseados em IoT evoluíram significativamente em termos de integração tecnológica, eficiência no uso de recursos e suporte a decisões. As arquiteturas centradas em nuvem demonstraram maturidade quanto à escalabilidade, governança e ferramentas de análise, mas apresentam limitações importantes quando aplicadas a áreas rurais sujeitas à instabilidade de conectividade. Por sua vez, os modelos híbridos nuvem–borda e nuvem–névoa–borda reforçam o potencial de reduzir latência, filtrar dados próximos à fonte e ampliar a autonomia local, alcançando benefícios concretos como maior eficiência energética, menor tráfego de rede e maior confiabilidade em cenários de longa distância. Ainda assim, nota-se que os mecanismos de replicação bidirecional e de consistência eventual em situações de desconexão prolongada permanecem pouco explorados.

As plataformas abertas e de baixo custo surgem como alternativas promissoras para a adoção em pequenas propriedades, conciliando interoperabilidade, reutilização de componentes e custo total de propriedade mais acessível. Contudo, tais soluções ainda carecem de políticas formais de *failover*, reconciliação pós-falha e replicação consistente entre camadas, aspectos críticos em contextos rurais marcados por falhas recorrentes de energia e conectividade.

De modo geral, a literatura revisada demonstra avanços significativos, mas também revela lacunas que motivam esta pesquisa. Entre elas destacam-se: (i) a ausência de protocolos robustos para sincronização confiável entre nuvem e borda; (ii) a necessidade de políticas eficazes de operação em modo desconectado; e (iii) a carência de estratégias práticas para garantir consistência de dados e continuidade do serviço em ambientes sujeitos a restrições de infraestrutura. Esses pontos orientam a contribuição desta dissertação, ao propor uma arquitetura resiliente para irrigação inteligente que enderece tais desafios.

5 METODOLOGIA

A construção da arquitetura proposta para o sistema de irrigação inteligente seguiu uma trajetória incremental, baseada na abordagem do Design Thinking. A evolução do projeto pode ser estruturada em quatro grandes fases, como ilustrado no fluxograma da evolução do trabalho (Figura 3):

- **Fase de Imersão e Definição do Problema**: caracterizada pela realização de um estudo de campo em uma fazenda, com o objetivo de compreender as necessidades dos usuários e levantar os requisitos iniciais, considerando os desafios da irrigação em ambientes rurais e as limitações de conectividade.

- **Fase de Ideação e Prototipação Inicial**: nesta fase, foram formuladas hipóteses sobre a arquitetura do sistema e exploradas alternativas técnicas. A prototipação inicial permitiu testar rapidamente essas premissas e avaliar sua viabilidade prática.

- **Fase de Desenvolvimento Iterativo**: dividida em duas frentes paralelas, a prototipação iterativa que envolveu ciclos de aprimoramento e testes práticos dos protótipos, e a fundamentação teórica que forneceu suporte conceitual por meio de estudos sobre arquiteturas IoT, segurança, modelagem e sistemas distribuídos.

- **Fase de Consolidação da Solução**: etapa na qual foi definida e validada a arquitetura final do sistema, integrando os conhecimentos adquiridos nas fases anteriores. O resultado foi uma solução distribuída, resiliente e escalável, adequada às condições de operação em ambientes rurais com conectividade limitada.

5.1 Fase de Imersão e Definição do Problema

O ponto de partida do projeto foi um estudo de campo realizado em uma fazenda no interior do estado de São Paulo, onde foram identificados os principais desafios relacionados à irrigação em ambientes remotos e à conectividade limitada. A partir desse contato direto com os usuários e suas rotinas, foi possível levantar requisitos iniciais considerando suas necessidades e as restrições do contexto rural.

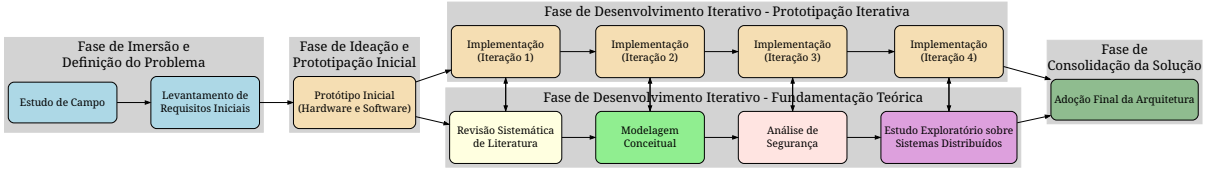

Figura 3 – Fluxograma da Evolução do Trabalho

Tabela 15 – Requisitos identificados

Requisito	**Tipo**
Agendamento da irrigação	Funcional
Notificação de falhas	Funcional
Monitoramento do sistema	Funcional
Registro de eventos	Funcional
Sistema multiusuário	Funcional
Facilidade de uso	Não Funcional
Compatibilidade com dispositivos básicos	Não Funcional
Resiliência a falhas de energia e conectividade	Não Funcional

5.1.1 Estudo de Campo

O estudo de campo foi realizado em 2021 em uma propriedade rural familiar que utiliza motobombas para circular água e nutrientes por canos de PVC elevados, onde as mudas são cultivadas. No entanto, o acúmulo de gases no sistema e as quedas frequentes de energia comprometem o funcionamento, ocasionando risco de perda total da produção em dias ensolarados.

A distância entre a plantação e a sede do sítio agrava o problema, exigindo deslocamento constante de mão de obra já sobrecarregada. O sítio encontra-se em uma área rural montanhosa e isolada, com acesso à internet apenas via satélite, cuja qualidade havia piorado recentemente. Além disso, os responsáveis pela plantação têm pouca familiaridade com tecnologia, dificultando o uso de soluções convencionais.

5.1.2 Levantamento de Requisitos Iniciais

O levantamento dos requisitos funcionais e não funcionais do sistema foi realizado a partir da observação prática durante o estudo de campo, complementada com entrevistas semiestruturadas conduzidas com os produtores da propriedade visitada. A entrevista ocorreu no mesmo momento da visita técnica, permitindo uma coleta de dados simultaneamente técnica e contextual, com base na experiência do usuário final.

A Tabela 15 resume os principais requisitos funcionais e não funcionais identificados.

5.2 Fase de Ideação e Prototipação Inicial

Com os requisitos definidos, o projeto avançou para a formulação de hipóteses sobre a arquitetura do sistema. Alternativas técnicas e conceituais foram exploradas com base na observação empírica e na documentação técnica já produzida. A prototipação inicial teve como objetivo testar rapidamente essas premissas e verificar sua viabilidade.

O desenvolvimento do primeiro protótipo teve como objetivo principal validar os fundamentos técnicos da arquitetura proposta, considerando as restrições identificadas

no estudo de campo, como conectividade limitada, instabilidade elétrica e o baixo nível de familiaridade tecnológica por parte dos usuários finais. A estrutura inicial do sistema foi concebida com base em experiências anteriores com projetos de IoT, servindo como referência para a definição de componentes e fluxos, e foi alinhada aos requisitos funcionais e não funcionais específicos levantados durante a análise do problema.

O sistema foi estruturado em três módulos: o módulo sensor, o servidor de borda (Edge) e o servidor na nuvem (Cloud). O módulo sensor é composto por um microcontrolador ESP32-WROOM-32D, responsável pela leitura periódica de três variáveis ambientais essenciais para a irrigação inteligente: umidade do solo, temperatura do ar e umidade relativa do ar. Os dados coletados são publicados via protocolo MQTT diretamente no broker Mosquitto, instalado no servidor de borda.

O servidor de borda, implementado com o framework Django e banco de dados SQLite, opera localmente e integra um cliente MQTT baseado na biblioteca Paho, responsável por consumir os dados publicados no Mosquitto e armazená-los para análise e tomada de decisão. Esse servidor disponibiliza também uma interface administrativa para os desenvolvedores, acessível por navegador, permitindo operação local, inclusive em cenários de desconexão com a internet.

O servidor de nuvem, hospedado em uma instância IaaS, utiliza igualmente o framework Django com banco SQLite e comunica-se com o servidor de borda por meio de um cliente Paho MQTT conectado ao broker HiveMQ. Ele é responsável por receber os dados da borda e enviar comandos ou atualizações de configuração de volta, mantendo uma comunicação bidirecional via MQTT. Também oferece uma interface administrativa e uma interface web para acesso remoto pelos usuários.

A arquitetura foi concebida para operar de forma resiliente e tolerante a falhas, com suporte a funcionamento offline na borda e sincronização posterior com a nuvem quando a conectividade for restabelecida. Ainda que funcionalidades como agendamento local automatizado e visualização gráfica dos dados históricos estejam em fase inicial de desenvolvimento, o protótipo já demonstra com êxito a viabilidade prática da arquitetura distribuída baseada em borda e nuvem, validando sua aplicação em contextos rurais com infraestrutura limitada. Esse funcionamento está representado no diagrama de contexto do protótipo inicial (Figura 4).

5.3 Fase de Desenvolvimento Iterativo

O desenvolvimento posterior ocorreu de forma iterativa e em duas frentes paralelas: uma dedicada aos estudos teóricos que embasaram decisões arquiteturais e outra voltada para a implementação e aprimoramento dos protótipos.

Os **estudos teóricos** foram conduzidos ao longo de quatro ciclos consecutivos e forneceram a base para o desenho da arquitetura final, sendo compostos por:

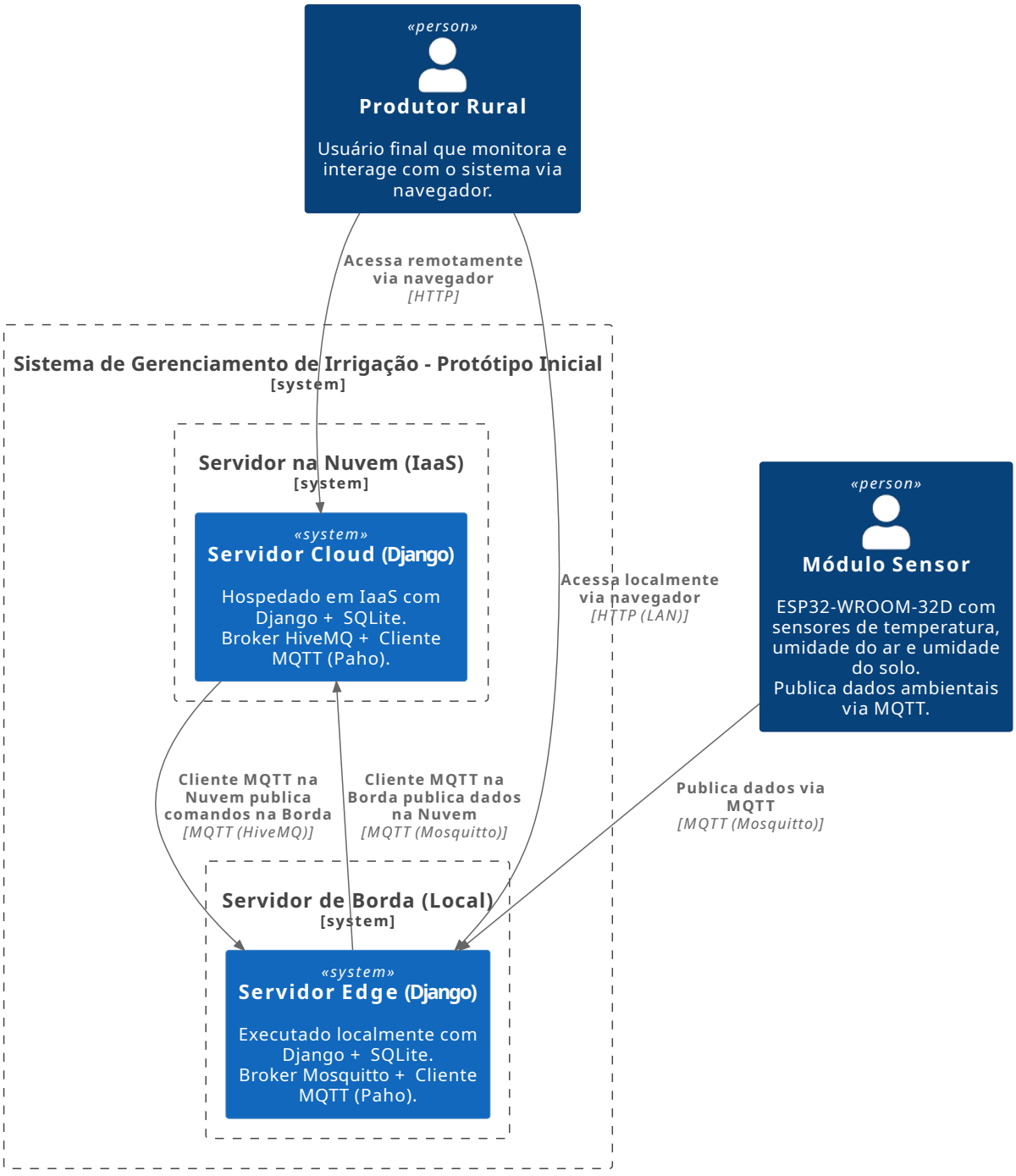

Figura 4 – Diagrama de contexto do protótipo inicial

- Uma revisão sistemática da literatura sobre arquiteturas IoT aplicadas à irrigação inteligente (Seção 5.4.1);
- A modelagem conceitual dos componentes e fluxos de comunicação do sistema (Seção 5.5.1);
- Uma análise de segurança, considerando as camadas de borda, nuvem e dispositivos embarcados (Seção 5.6.1);
- Um estudo exploratório sobre sistemas distribuídos, que fundamentou a escolha de técnicas de comunicação e consistência de dados (Seção 5.7.1).

Concomitantemente aos estudos teóricos, ocorreu a **etapa de implementação**, na qual o protótipo inicial evoluiu com base nas definições arquiteturais estabelecidas. As melhorias foram incorporadas progressivamente a partir de testes práticos. Cada nova versão serviu para validar soluções, identificar limitações e promover o refinamento contínuo do sistema.

5.4 Iteração 1

Na primeira iteração, foi conduzido um estudo teórico que buscou mapear as principais abordagens arquiteturais aplicadas à irrigação inteligente baseada em IoT, por meio de uma revisão sistemática de literatura. Na etapa de implementação, essas definições arquiteturais foram consolidadas e integradas à evolução do protótipo inicial.

5.4.1 Revisão Sistemática de Literatura

A revisão sistemática de literatura (RSL) conduzida no âmbito deste trabalho teve como objetivo identificar, analisar e extrair padrões arquiteturais utilizados na concepção de sistemas de irrigação inteligentes baseados em IoT. A pesquisa foi guiada por um protocolo rigoroso que definiu objetivos, questões de pesquisa, critérios de inclusão, exclusão e qualidade.

As questões norteadoras da RSL buscaram compreender: (i) quais componentes de hardware são utilizados nesses sistemas, (ii) como o software é estruturado, (iii) quais tecnologias de comunicação são adotadas, (iv) de que forma a alimentação elétrica dos dispositivos é provida e (v) como é realizada a proteção física contra intempéries.

Foram realizadas buscas nas bases IEEE Xplore, ACM Digital Library e Scopus, com strings elaboradas a partir de sinônimos técnicos como *Smart Irrigation*, *Precision Irrigation* e *Irrigation Scheduling*. A busca inicial resultou em 8.614 publicações, que, após filtragens por critérios de inclusão (publicações entre 2014 e 2023, com descrição de arquitetura) e exclusão (duplicações, falta de texto completo, irrelevância), resultaram em 25 estudos selecionados.

A análise dos artigos permitiu a categorização dos elementos arquiteturais em cinco grandes dimensões:

- **Hardware:** A arquitetura típica envolve microcontroladores (como ESP32, ESP8266 e Arduíno) e microprocessadores (como Raspberry Pi), organizados em uma estrutura hierárquica entre nós sensores/atuadores e um gateway intermediário. Sensores de umidade do solo, temperatura e umidade do ar foram os mais recorrentes.
- **Software:** Verificou-se grande heterogeneidade na pilha de software, com predominância do uso de protocolos como MQTT e REST, e linguagens como Python, C/C++, Node.js, além do uso de plataformas como ThingSpeak, Firebase e Blynk. A diversidade observada reforça a ausência de um padrão de mercado estabelecido.
- **Comunicação:** As arquiteturas analisadas utilizaram majoritariamente tecnologias sem fio. Wi-Fi, ZigBee, LoRa e redes celulares (GPRS/4G) foram as mais comuns, sendo o Wi-Fi o mais utilizado nos gateways. A presença de redes mesh foi notada em soluções com ZigBee e LoRa, particularmente vantajosas para ambientes agrícolas distribuídos.
- **Alimentação Elétrica:** Devido à limitação de infraestrutura energética nas áreas rurais, muitos sistemas utilizavam painéis solares com baterias recarregáveis (íon de lítio, chumbo-ácido, etc.). A seleção da fonte de alimentação variou de acordo com o consumo do hardware e a necessidade de autonomia.
- **Proteção Física:** A proteção dos dispositivos contra intempéries é uma preocupação recorrente, embora nem sempre detalhadamente descrita. Foram encontrados casos de uso de caixas estanques, carcaças IP65 e até impressão 3D com materiais resistentes à umidade.

Os achados da RSL reforçaram as decisões arquiteturais preliminares adotadas no protótipo inicial deste trabalho, como a separação entre borda e nuvem, o uso de Wi-Fi para comunicação local e a adoção de sensores de umidade do solo e temperatura do ar como base para acionamento. A diversidade de soluções identificadas evidenciou a necessidade de flexibilidade arquitetural, e a importância de garantir resiliência, modularidade e tolerância a falhas em ambientes hostis. Esse estudo permitiu refinar os requisitos não funcionais do sistema proposto, além de fundamentar tecnicamente a seleção de componentes e protocolos, posteriormente validados nos ciclos de prototipação e testes.

5.4.2 Implementação

A partir da validação dos conceitos e das decisões arquiteturais identificadas na revisão sistemática da literatura, a implementação concentrou-se em transpor esses elementos para o contexto da propriedade rural estudada.

A arquitetura implementada seguiu o modelo em camadas recomendado por estudos prévios, organizando o sistema em três níveis distintos: dispositivos IoT (ESP32), servidor de borda (Orange Pi) e servidor na nuvem (IaaS). O ESP32 se mostrou uma escolha adequada devido à sua boa relação entre custo, simplicidade e desempenho. Já o Orange Pi foi selecionado como hardware da borda por oferecer baixo consumo de energia, conectividade estável e capacidade de processamento local. Trata-se de uma plataforma semelhante ao Raspberry Pi, mas com melhor relação custo-benefício e maior disponibilidade no mercado, evitando os problemas de desabastecimento frequentemente associados ao Raspberry Pi.

A comunicação entre os sensores e o servidor de borda foi realizada por meio de uma rede Wi-Fi local, aproveitando a compatibilidade nativa do ESP32 com esse protocolo. Para garantir conectividade estável e dedicada, foi adicionado um ponto de acesso exclusivo ao sistema, instalado junto ao servidor de borda. Essa abordagem foi motivada pela ampla presença do Wi-Fi em soluções agrícolas revisadas, além do baixo custo e da facilidade de implementação nos ambientes rurais analisados.

A conexão com a internet entre o servidor de borda e o servidor na nuvem pode variar conforme a infraestrutura disponível no local de instalação.

Para a comunicação local entre dispositivos IoT e a borda, adotou-se o protocolo MQTT como padrão, conforme a predominância observada na literatura. A leveza do MQTT, sua tolerância a redes instáveis e sua ampla compatibilidade com bibliotecas embarcadas tornaram-no uma escolha estratégica para o ambiente de teste.

O software do sistema seguiu com o uso do Django, da biblioteca Paho MQTT e dos brokers Mosquitto (na borda) e HiveMQ (na nuvem).

No nível físico, uma ênfase significativa foi dada à proteção dos dispositivos instalados em campo. As unidades foram alojadas em invólucros com grau de proteção IP65, vedados contra poeira e umidade, e com resistência a intempéries. A literatura destaca que a ausência de proteção adequada pode comprometer seriamente a durabilidade dos equipamentos, sendo essa medida preventiva essencial.

Reconhecendo a instabilidade energética da região, a infraestrutura foi projetada com fontes de energia redundantes. Os módulos IoT foram projetados para operar com baterias recarregáveis alimentadas por painéis solares, garantindo maior autonomia em campo. Já o servidor de borda e seu ponto de acesso (Access Point) passaram a ser alimentados por uma fonte de alimentação ininterrupta (UPS), assegurando o funcionamento contínuo do sistema mesmo durante interrupções no fornecimento de energia elétrica.

Uma importante observação desta etapa é que diversas escolhas arquiteturais foram validadas a partir dos resultados da revisão sistemática de literatura, sendo incorporadas ao protótipo do sistema. No entanto, do ponto de vista do modelo conceitual de alto nível, a única alteração que se refletiria diretamente no diagrama de contexto seria a definição do hardware da borda, com a adoção do Orange Pi. As demais mudanças, embora

relevantes nos níveis físico e operacional, como a implementação do ponto de acesso dedicado, do sistema de alimentação e dos mecanismos de proteção física, não impactam a lógica estrutural da arquitetura representada no diagrama de contexto. Por esse motivo, optou-se por sua omissão nesta etapa, uma vez que sua elevada similaridade com o diagrama anterior poderia gerar confusão ao leitor, sem agregar informações significativas à compreensão da evolução arquitetural do sistema.

5.5 Iteração 2

A segunda iteração foi marcada pela formalização da modelagem conceitual utilizando UML, pela separação lógica dos módulos IoT em sensores e atuadores e pela definição do padrão de nomeação dos objetos e formato de troca de informações. Além disso, realizou-se a migração dos brokers MQTT, com o objetivo de implementar um mecanismo de replicação automática entre borda e nuvem.

5.5.1 Modelagem Conceitual

A modelagem teve papel central na definição da arquitetura do sistema. Utilizando a linguagem UML e princípios de orientação a objetos, foram elaborados artefatos que representam a estrutura, o comportamento e as interações entre os componentes. O modelo incorporou monitoramento contínuo do estado do sistema, geração de alertas para falhas críticas detectadas pelos sensores e integração entre borda e nuvem para garantir a continuidade do serviço, além de estabelecer padrões de segurança e controles de acesso para proteger os dados dos usuários e assegurar a integridade das informações.

Organizado em três camadas, dispositivos IoT (sensoriamento e atuação), borda e nuvem, o modelo permitiu visualizar os fluxos do sistema de forma distribuída, alinhando dispositivos físicos, lógica de operação e infraestrutura. Um dos principais resultados foi a decisão de separar os dispositivos IoT em dois módulos distintos: sensor e atuador. A modelagem também orientou a adoção de identificadores únicos (`uuid`) como padrão para a identificação dos objetos, garantindo unicidade e rastreabilidade. Definiu-se o uso de campos no formato `JSON` para armazenar os dados provenientes dos dispositivos, o que proporciona maior flexibilidade na estruturação das informações, facilita a interoperabilidade e permite a adaptação do modelo de dados à medida que os requisitos evoluem.

A modelagem também definiu o uso do protocolo MQTT como base para a comunicação assíncrona entre os módulos IoT, o servidor de borda e a nuvem, incorporando mecanismos de replicação dos dados e confirmação das leituras transmitidas. O diagrama de máquina de estados evidenciou a importância de um mecanismo de consistência eventual para a sincronização de leituras entre borda e nuvem, enquanto os diagramas de fluxo de dados ajudaram a definir a comunicação entre os componentes. Foram utilizados diagramas de casos de uso, classes, sequência, estados e implantação para descrever

funcionalidades, responsabilidades e aspectos físicos da solução. A modelagem também sustentou a rastreabilidade entre requisitos e implementação, contribuindo para a validação técnica do sistema.

5.5.2 Implementação

A segunda iteração concentrou-se na validação da comunicação entre os diferentes componentes distribuídos, dispositivos IoT, servidor de borda e servidor na nuvem, além de iniciar a experimentação prática com mecanismos de replicação de dados.

A arquitetura inicial do sistema utilizou os brokers MQTT HiveMQ e Mosquitto. Entretanto, o protocolo MQTT, por definição, não possui mecanismos nativos de replicação automática de dados com garantia de entrega em cenários nos quais os clientes estejam desconectados do broker, como ocorre em casos de falha de rede. Dessa forma, a implementação de mecanismos de replicação e sincronização dependeria de soluções manuais, o que, além de exigir desenvolvimento adicional, introduziria desafios arquiteturais significativos e suscetíveis a diversas vulnerabilidades, especialmente no que se refere à consistência dos dados, tolerância a falhas e integridade das informações. Diante desse cenário, foi realizada uma pesquisa com o objetivo de identificar soluções MQTT que oferecessem, de forma nativa, suporte à replicação de dados entre brokers, atendendo aos requisitos de resiliência, continuidade operacional e robustez, sem a necessidade de desenvolvimento de soluções paralelas para esse propósito.

Durante a pesquisa foram identificados dois brokers MQTT que oferecem suporte nativo às funcionalidades necessárias: o *NanoMQ* e o *EMQX*. Ambos são desenvolvidos pelo mesmo grupo, mas possuem escopos de aplicação distintos. O *NanoMQ* foi selecionado por ser um broker leve, multi-thread e otimizado, adequado para operação em servidores de borda, onde há restrições de recursos. Por outro lado, o *EMQX* é um broker robusto, escalável e baseado na linguagem Erlang, oferecendo suporte massivo a conexões paralelas, alta disponibilidade e tolerância a falhas, sendo ideal para ambientes de nuvem.

Ambos os brokers oferecem a funcionalidade denominada *Data Bridge*, que permite a replicação automática dos tópicos locais para tópicos remotos em outros brokers. Esse mecanismo inclui suporte à utilização da flag `retain`, possibilitando o reenvio automático de mensagens em casos de falhas de comunicação (LEA, 2020, p. 396). Além disso, ambos oferecem persistência local por meio de bancos de dados embarcados, o que garante que as mensagens sejam armazenadas de forma segura até que possam ser entregues e confirmadas no destino. Essa estratégia atende diretamente aos requisitos de resiliência e continuidade operacional do sistema.

Conforme definido na etapa de modelagem, os dispositivos IoT foram separados em dois módulos: leitor e atuador. Os objetos do sistema Django passaram a utilizar *UUID* como identificador padrão. Além disso, os campos destinados ao armazenamento dos dados dos módulos IoT foram ajustados para o tipo adequado ao armazenamento de

estruturas em *JSON*.

O cliente MQTT embarcado nos módulos IoT manteve a comunicação com o broker na borda, que agora passou a ser o *NanoMQ*. No entanto, a comunicação entre borda e nuvem foi transferida para o mecanismo *Data Bridge* entre os brokers. Com essa mudança, os clientes MQTT locais, implementados com *Paho*, passaram a se comunicar exclusivamente com o broker e com o sistema Django presentes em sua respectiva instância, deixando de realizar a sincronização direta entre borda e nuvem.

Como o script de instalação do *NanoMQ* não disponibilizava uma versão compatível com o Orange Pi, que utiliza o Arch Linux como sistema operacional, foi adotado o uso de contêineres Docker para executar a instância do *NanoMQ* no servidor de borda.

As modificações arquiteturais do sistema na segunda iteração do processo de prototipação são representadas no diagrama de contexto do protótipo da Iteração 2 (Figura 5).

5.6 Iteração 3

A terceira iteração do desenvolvimento foi orientada pela análise de segurança do sistema, abrangendo tanto os componentes de hardware quanto de software. Foram avaliados aspectos relacionados à proteção dos servidores, dos dispositivos IoT, das interfaces de comunicação, da alimentação elétrica e da segurança física dos equipamentos.

Dentre os mecanismos implementados, destacam-se a adoção de um processo de deploy baseado em contêineres, que garante isolamento, portabilidade e consistência no ambiente de execução; a substituição do servidor de desenvolvimento por uma arquitetura mais robusta e adequada a ambientes de produção, utilizando servidores WSGI e proxy reverso; e a configuração de mecanismos de comunicação segura, assegurando a confidencialidade, integridade e autenticidade dos dados transmitidos.

5.6.1 Análise de Segurança

A segurança foi tratada de forma transversal na arquitetura do sistema de irrigação inteligente, com atenção específica à proteção dos componentes distribuídos e das comunicações. O estudo concentrou-se em três camadas principais: nuvem, borda e dispositivos IoT, além das interfaces de comunicação, da alimentação elétrica e da proteção física do hardware.

5.6.1.1 Servidor de Nuvem

Na nuvem, foram utilizados contêineres Docker para garantir o isolamento dos serviços e a consistência no ambiente de execução. Foram configurados volumes específicos para armazenamento de logs, permitindo a auditoria e o monitoramento contínuo, e volumes de dados para garantir a persistência das informações em caso de interrupções. Houve

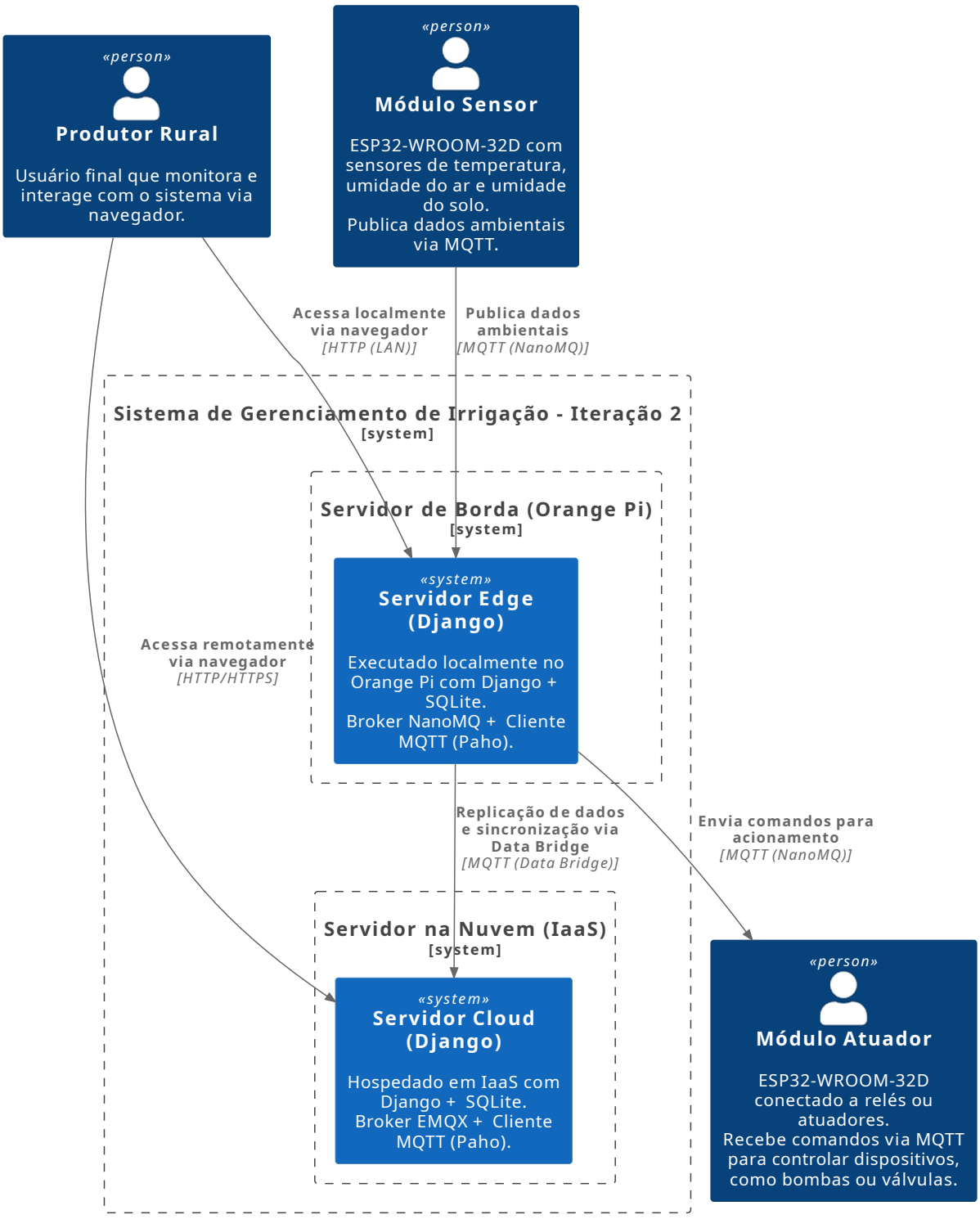

Figura 5 – Diagrama de contexto do protótipo da Iteração 2

a devida atenção à configuração dos parâmetros de cada serviço e à correta definição do orquestrador, assegurando a estabilidade, a portabilidade e a integridade do ambiente.

O servidor de desenvolvimento nativo do Django foi substituído por um servidor WSGI (Web Server Gateway Interface), que é o padrão para aplicações Python em ambientes de produção. Essa substituição foi essencial para garantir a escalabilidade do sistema, uma vez que o servidor de desenvolvimento não é projetado para lidar com um grande volume de requisições simultâneas. Associado ao servidor WSGI, foi implementado um proxy reverso, que além de encaminhar as requisições de forma eficiente, também atua como servidor de arquivos estáticos, contribuindo para a otimização do desempenho e da gestão dos recursos.

O Django oferece, por padrão, mecanismos de proteção contra ataques XSS (*Cross-Site Scripting*) e CSRF (*Cross-Site Request Forgery*), por meio do seu sistema de *templates*, que realiza *escaping* automático, e do middleware responsável pela verificação de tokens de sessão. No entanto, a efetividade dessas proteções depende da correta implementação por parte do desenvolvedor, especialmente quanto à inclusão explícita dos tokens de CSRF nos formulários e ao uso criterioso dos filtros de *template*, evitando a desativação inadvertida dos mecanismos de escaping.

Além disso, diversos parâmetros críticos de segurança devem ser rigorosamente observados na configuração do Django. Dentre os mais relevantes, destacam-se:

A variável `SECRET_KEY`, utilizada internamente pelo framework para a geração de tokens de sessão, de CSRF e de outras assinaturas criptográficas. A exposição dessa chave compromete gravemente a segurança da aplicação, permitindo a falsificação de sessões, a quebra de tokens e outros tipos de ataques baseados na manipulação de dados assinados.

Outro parâmetro fundamental é `DEBUG`, que deve estar, obrigatoriamente, configurado como `False` em ambientes de produção. Quando ativado, o modo de depuração expõe informações sensíveis, como variáveis de ambiente, detalhes da infraestrutura, rastreamentos de erro (`stack traces`) e outros dados internos que podem ser explorados por agentes mal-intencionados.

Em relação ao controle de acesso da aplicação web, o Django possui um sistema robusto de gestão de usuários, que implementa mecanismos de autenticação e autorização, permitindo a definição de permissões personalizadas e o uso de grupos (*roles*) para atribuição de perfis de acesso. Esse modelo oferece alta granularidade no controle de operações, sendo fundamental para atender aos requisitos de um sistema multiusuário e *multi-tenant*, como é o caso da aplicação desenvolvida.

Uma decisão arquitetural anterior que contribui diretamente para a segurança do sistema é a adoção de UUIDs como padrão no sistema de nomeação de objetos. Essa estratégia impede que identificadores presentes em URIs e nos endereços dos tópicos dos *brokers* de mensagens sejam facilmente dedutíveis por atacantes, uma vez que os UUIDs não seguem sequências previsíveis ou padrões legíveis, reduzindo significativamente a su-

perfície de exposição a ataques baseados em enumeração ou exploração de padrões na estrutura dos identificadores. Esses aspectos foram devidamente considerados no desenvolvimento do sistema web, assegurando a aplicação consistente das práticas de segurança recomendadas pelo *framework*.

Para garantir a confidencialidade e a integridade dos dados em trânsito, foi implementado suporte a HTTPS, por meio da utilização de certificados digitais válidos, assegurando a criptografia nas comunicações HTTP.

Da mesma forma, o `broker MQTT` utilizado (EMQX) foi configurado com suporte a TLS, proporcionando uma camada adicional de segurança na troca de mensagens. Outras configurações de segurança do `broker` MQTT que devem ser observadas incluem a autenticação obrigatória dos clientes, garantindo que apenas dispositivos autorizados possam se conectar, e o uso de políticas de controle de acesso (ACL) para definir quais tópicos cada dispositivo pode publicar ou assinar, prevenindo acessos e publicações indevidas. Devem ser aplicadas limitações no número de conexões simultâneas, na taxa de envio de mensagens e no tamanho dos pacotes, como medida de proteção contra ataques de negação de serviço (DoS).

5.6.1.2 Servidor de Borda

O servidor de borda foi implementado em um dispositivo Orange Pi, executando um sistema operacional baseado em Arch Linux (Orange Pi OS). Embora a distribuição Linux, utilizada no servidor de borda, seja considerada uma distribuição segura, alguns aspectos de segurança devem ser rigorosamente observados. Entre eles, destacam-se a correta configuração do sistema, a adoção de boas práticas na gestão de senhas e credenciais, e a implementação de regras adequadas no `firewall`, de forma a restringir acessos não autorizados e proteger os serviços expostos na rede.

A pilha de softwares adotada é semelhante à utilizada no ambiente de nuvem, incorporando as mesmas preocupações de segurança apresentadas anteriormente. Entretanto, apresenta peculiaridades específicas relacionadas à sua implementação, que serão discutidas na **Subseção 5.6.2**.

Um ponto de atenção fundamental durante a implementação foi a escolha do dispositivo de armazenamento. Inicialmente, o sistema utilizava um cartão SD, que, embora prático, apresenta alta suscetibilidade a falhas e corrupção de dados, especialmente em operações com escrita intensiva.

Diante dessa limitação, houve a substituição do hardware, migrando do modelo Orange Pi Zero 2W para o Orange Pi 3B. Além de ser um equipamento mais robusto em termos de processamento e estabilidade, a principal motivação para essa troca foi a presença de slots compatíveis com eMMC e SSD no padrão M.2(LEA, 2020, p. 330). Esses dispositivos de armazenamento oferecem maior confiabilidade, robustez e desempenho, tanto em operações de leitura quanto de escrita, quando comparados aos cartões SD

tradicionais(LEA, 2020, p. 330). Essa mudança foi essencial para garantir a integridade dos dados e a estabilidade operacional do servidor de borda(LEA, 2020, p. 313, 329).

5.6.1.3 Dispositivos IoT

Os dispositivos IoT inicialmente utilizavam o modelo ESP32-WROOM-32D. No entanto, com a descontinuação desse modelo, optou-se pela substituição pelo ESP32-S3. Este microcontrolador oferece desempenho superior, mantendo a arquitetura dual-core, característica adequada para aplicações IoT, assim como seu antecessor.

Além do aumento de desempenho, o ESP32-S3 incorpora novos recursos, como suporte ao conjunto de instruções SIMD, que permite aceleração para tarefas de inteligência artificial embarcada e visão computacional, viabilizando futuras expansões do sistema. Adicionalmente, o modelo conta com um acelerador criptográfico mais eficiente e atualizado, aprimorando os mecanismos de segurança e mantendo a compatibilidade com os requisitos operacionais do projeto.

Durante a pesquisa, foram levantadas medidas de proteção contra a extração de dados (*dump*), contra tentativas de adulteração (*tampering*) tanto do *firmware* quanto do hardware dos dispositivos IoT, além da análise da viabilidade de atualizações OTA (*Over-The-Air*) seguras. No contexto do `firmware`, foram analisadas práticas para mitigar riscos associados ao *firmware tampering*, que consistem na substituição, modificação ou injeção de código malicioso no *firmware* embarcado. Nesse sentido, foram considerados mecanismos como verificação de integridade, assinatura digital do *firmware*, *boot* seguro (*Secure Boot*) e criptografia da memória flash (*Flash Encryption*).

Adicionalmente, foram analisadas as funcionalidades de segurança baseadas nos eFuses do ESP32, que permitem a gravação irreversível de chaves criptográficas diretamente no hardware. Uma vez configurados, esses fusíveis digitais impedem a modificação do *firmware* ou a leitura do conteúdo da memória flash, a menos que a autenticação correta seja realizada(LEA, 2020, p. 562). Esse mecanismo garante que apenas *firmwares* devidamente assinados sejam carregados no dispositivo, além de proteger contra ataques de engenharia reversa e tentativas de extração física dos dados.

No aspecto do hardware, também foram consideradas ameaças como hardware *tampering*, incluindo acesso não autorizado a interfaces de *debug*, interceptação de barramentos e ataques por injeção de falhas(LEA, 2020, p. 533). Para mitigar esses riscos, devem ser adotadas práticas como o encapsulamento físico dos componentes, bloqueio das portas de depuração e aplicação de resinas epóxi sobre circuitos sensíveis.

Por fim, também foram avaliados os benefícios da migração da plataforma Arduino para o framework ESP-IDF, associado ao sistema operacional em tempo real FreeRTOS, fornecido pela Espressif. Essa abordagem permite não apenas um controle mais granular sobre os recursos de hardware e segurança, como também o uso nativo dos mecanismos avançados de proteção oferecidos pela plataforma ESP32.

No entanto, devido às limitações do escopo deste projeto, esses tópicos foram direcionados para desenvolvimentos futuros. Ainda assim, é importante destacar sua relevância, especialmente no que se refere à segurança dos dispositivos e à escalabilidade da solução.

As vulnerabilidades presentes nos componentes eletrônicos dos módulos sensor e atuador também foram consideradas na análise de segurança, uma vez que esses dispositivos representam pontos críticos na integridade do sistema. Esses módulos são compostos pelos seguintes componentes:

- **Módulo Sensor:** sensor de temperatura e umidade do ar, sensor capacitivo de umidade do solo.

- **Módulo Atuador:** sensor de temperatura e umidade do ar, sensor de fluxo, sensor de corrente elétrica e atuador do tipo relé.

Dentre as vulnerabilidades associadas aos sensores, destaca-se a necessidade de proteção física contra danos, fator crítico em ambientes rurais. Isso evidencia a importância de implementar proteções mecânicas e sinalização adequada, de forma que os trabalhadores não danifiquem acidentalmente os dispositivos durante suas atividades rotineiras. Além disso, o ambiente impõe desafios como corrosão, poeira, umidade e intempéries, que foram diretamente considerados na seleção dos componentes. Um exemplo disso foi a escolha do sensor de umidade do solo do tipo capacitivo, mais resistente à corrosão em comparação com sensores do tipo resistivo, que são mais suscetíveis à degradação ao longo do tempo.

Outros vetores de ataque envolvem tentativas de manipulação das leituras, alteração de calibração em sensores que possuem esse recurso e, em alguns casos, modificação do *firmware* em sensores programáveis. A troca física de sensores, intencional ou acidental, também é um risco a ser considerado, especialmente quando os dispositivos não possuem mecanismos robustos de verificação de identidade ou autenticação física.

Outro fator que deve ser considerado na segurança do projeto é a correta escolha do relé, que deve ser compatível com a tensão, corrente e o tipo de carga a ser acionada. A escolha incorreta desse dispositivo pode provocar falhas no acionamento, comprometer a operação do sistema e, em situações mais críticas, gerar riscos de acidentes elétricos. O atuador utilizado é o relé de estado sólido (SSR) Fotek SSR-40DA (40A), compatível com cargas monofásicas em ampla faixa de tensão AC. Possui isolação óptica e baixa corrente de acionamento, sendo compatível diretamente com as saídas de 3,3 V do ESP32. Embora seja possível o acionamento direto, recomenda-se o uso de circuitos de interface com opto-acopladores, como o TLP250, que proporcionam isolação galvânica, protegendo o ESP32 contra ruídos, surtos e interferências, além de aumentar a robustez do sistema, especialmente em cargas indutivas, como motobombas.

O SSR permite controlar sistemas elétricos monofásicos de média potência. Contudo, motobombas podem variar quanto ao número de fases, corrente, potência e modos de acionamento, como o uso de contatores de estado sólido (CSS). Assim, o dimensionamento elétrico adequado é fundamental para garantir segurança e confiabilidade, sendo esse aspecto fora do escopo deste projeto.

Devido às limitações de acesso a equipamentos, este projeto adota uma válvula solenoide como dispositivo de controle hídrico. Por demandar baixa corrente, não há necessidade de dissipador térmico no SSR.

5.6.1.4 Interfaces de comunicação

Diversos protocolos de comunicação foram analisados durante esta etapa. Embora o Wi-Fi tenha sido escolhido como tecnologia de comunicação para a primeira versão do sistema, outros protocolos foram inicialmente considerados, tanto de curto alcance, como Bluetooth e ESP-NOW, quanto de longo alcance, como Zigbee e LoRa. Entretanto, por questões de escopo e priorização, a implementação desses protocolos foi postergada para desenvolvimentos futuros.

A adoção do Wi-Fi exigiu a consideração de medidas específicas de segurança, incluindo a definição de senhas robustas, a escolha de protocolos de autenticação seguros (como WPA2 ou WPA3) e a correta configuração do ponto de acesso, de forma a minimizar riscos de acesso não autorizado. Além dos aspectos relacionados à segurança lógica, também foram avaliados fatores operacionais, como o alcance do sinal, a degradação em função de barreiras físicas e a perda de visada direta entre o ponto de acesso e os dispositivos IoT. Riscos associados a interferências eletromagnéticas, sejam acidentais, originadas de motores e equipamentos locais, ou intencionais, como ataques de jamming, também foram considerados na análise, uma vez que podem comprometer a disponibilidade e a confiabilidade da comunicação no ambiente rural.

5.6.1.5 Segurança Energética e Continuidade Operacional

A operação contínua do sistema foi assegurada por meio de soluções resilientes a quedas de energia, como o uso de uma fonte de alimentação ininterrupta (UPS) para o servidor de borda e o ponto de acesso, e de baterias alimentadas por energia solar nos dispositivos IoT, conforme discutido anteriormente(LEA, 2020, p. 10, 72). Além disso, o estudo também analisou a possibilidade de adotar redundância energética nos dispositivos IoT, utilizando alimentação elétrica convencional como fonte primária, com o sistema solar e as baterias funcionando como fonte secundária de backup.

Aspectos relacionados à segurança das baterias foram igualmente considerados, uma vez que, em caso de danos físicos ou falhas, podem apresentar riscos de incêndio. Esse risco é particularmente relevante no uso de baterias de íon de lítio (Li-Ion), escolhidas

por suas características físicas, como baixo peso e alta densidade energética, que as tornam adequadas para sistemas embarcados(LEA, 2020, p. 82).

O sistema de alimentação baseado em baterias inclui um módulo de gerenciamento capaz de controlar tanto o carregamento quanto a descarga, funcionando também como um sistema UPS. Esse módulo assegura o fornecimento contínuo de energia ao microcontrolador (ESP32) durante quedas da rede elétrica (em sistemas com redundância) ou em situações onde a geração solar é insuficiente.

Os componentes eletrônicos que compõem o sistema de alimentação dos dispositivos IoT são:

- **18650 Battery Shield**: módulo para carregamento, proteção e fornecimento de energia ininterrupta.
- **Bateria 18650 Li-Ion**: célula cilíndrica de alta densidade energética.
- **Regulador de Tensão Step-Up (MT3608)**: circuito elevador de tensão que garante o funcionamento estável do ESP32. Foi adotado após testes que evidenciaram desligamentos ocasionais, causados pela queda natural de tensão da célula durante a descarga.
- **Painel Solar 10W 5V**: responsável pela geração de energia para operação autônoma.

Optou-se inicialmente por hardware disponível no mercado, e está prevista, como desenvolvimento futuro, uma análise da autonomia energética, visando validar se o sistema suporta quedas prolongadas de energia e dimensionar adequadamente sua capacidade.

5.6.1.6 Proteção Física dos Dispositivos

A proteção física dos dispositivos deve considerar os desafios impostos pelo ambiente rural. Foram adotadas medidas para garantir tanto a integridade dos equipamentos quanto a continuidade da operação em condições adversas(LEA, 2020, p. 343, 528). Entre as soluções de proteção levantadas, destacam-se:

- **Caixas herméticas com grau de proteção IP67/IP68**, que asseguram vedação contra poeira, umidade e chuva(LEA, 2020, p. 343).
- **Sistemas de dissipação térmica** visando evitar o superaquecimento dos dispositivos durante longos períodos de exposição ao sol(LEA, 2020, p. 10).
- **Revestimentos anticorrosão** na estrutura física de suporte, conectores, fiação e sensores, prevenindo danos causados por umidade, maresia ou produtos químicos presentes no ambiente agrícola.

- **Barreiras físicas e sinalização adequada** para proteção contra impactos acidentais, especialmente em áreas de circulação de veículos agrícolas ou durante atividades de manejo com ferramentas.

- **Encapsulamento de circuitos sensíveis** e proteção adicional nas interfaces de comunicação e energia, reduzindo riscos tanto de falhas operacionais quanto de manipulações não autorizadas.

Por fim, outros aspectos relevantes de segurança foram analisados, porém direcionados para desenvolvimentos futuros. Embora o sistema já disponha de mecanismos de geração de *logs* e de acompanhamento básico da saúde dos serviços, prevê-se, em etapas posteriores, a adoção de soluções mais robustas de observabilidade e monitoramento. Da mesma forma, foram avaliadas práticas avançadas de segurança de rede, como a segregação de tráfego por meio de VLANs e a expansão das regras de *firewall* para o ponto de acesso, complementando as já aplicadas no servidor de borda.

5.6.2 Implementação

Na terceira iteração do processo de implementação, o foco principal foi a incorporação, ao protótipo, dos mecanismos de segurança identificados na análise anterior.

5.6.2.1 Servidor de Nuvem

No ambiente de nuvem, o *framework* web Django passou a utilizar o banco de dados PostgreSQL, mais adequado a ambientes de produção e capaz de oferecer maior robustez, escalabilidade e integridade transacional.

O servidor de desenvolvimento nativo do Django foi substituído por uma arquitetura composta pelo servidor WSGI Gunicorn, operando em conjunto com o Nginx configurado como proxy reverso. Essa configuração segue as boas práticas recomendadas oficialmente pelo próprio framework, contribuindo para maior desempenho, segurança e escalabilidade da aplicação. As recomendações de segurança do Django, discutidas anteriormente, foram adotadas como premissa nas diretrizes de desenvolvimento.

A comunicação MQTT permaneceu implementada por meio da biblioteca Paho, devidamente configurada com mecanismos de autenticação, mantendo sua função de cliente MQTT. Por sua vez, o *broker* MQTT (EMQX) foi configurado com suporte a TLS, garantindo a confidencialidade, integridade e autenticidade dos dados transmitidos.

O uso de contêineres Docker, que anteriormente estava restrito ao broker na borda, foi estendido para todos os componentes do servidor de nuvem. Esse modelo de virtualização leve assegura isolamento dos serviços, portabilidade e consistência no ambiente de execução. Para a orquestração desses contêineres adotou-se o Docker Compose, permitindo gerenciamento simplificado, automação de *deploy* e manutenção do ambiente.

5.6.2.2 Servidor de Borda

A pilha de softwares do servidor de borda segue a mesma estrutura conceitual adotada na nuvem, com adaptações específicas descritas a seguir.

O Django manteve o uso do banco de dados SQLite, por ser mais adequado a cenários de operação local, caracterizados por menor volume de transações e baixa demanda por concorrência. Assim como na nuvem, adotou-se a arquitetura baseada no servidor WSGI Gunicorn, operando em conjunto com o Nginx configurado como *proxy* reverso.

O broker MQTT utilizado foi o NanoMQ, devidamente configurado com suporte a TLS, garantindo a confidencialidade, a integridade e a autenticidade dos dados transmitidos. O cliente MQTT permaneceu implementado com a biblioteca Paho.

Adicionalmente, o uso de contêineres, inicialmente restrito ao NanoMQ, foi expandido para todos os serviços do servidor de borda, adotando também o Docker Compose para orquestração. Essa abordagem assegura isolamento, portabilidade e consistência no ambiente de execução.

5.6.2.3 Dispositivos IoT

O firmware dos dispositivos IoT foi adaptado para incorporar os mecanismos de autenticação junto ao broker MQTT, refletindo também as atualizações realizadas na arquitetura dos sensores e atuadores.

5.6.2.4 Proteção Física dos Dispositivos

A proteção física dos dispositivos IoT foi projetada considerando os desafios do ambiente rural, priorizando soluções de baixo custo, robustas e de fácil instalação. A estrutura é composta por um suporte tubular metálico, que serve como base para a fixação das caixas herméticas, dos painéis solares e para a organização da fiação. As caixas foram selecionadas com grau de proteção adequado para ambientes externos, sendo previstas também plataformas superiores para sombreamento, a fim de reduzir o aquecimento dos componentes(LEA, 2020, p. 343, 528). Sobre essas plataformas são fixados os painéis solares, com suportes ajustáveis para otimizar a inclinação de acordo com a posição solar.

A estrutura requer tratamento contra corrosão, como galvanização, pintura anticorrosiva ou aplicação de *primer*, a fim de assegurar maior durabilidade e resistência às intempéries.

Ainda está em desenvolvimento a definição da base estrutural, considerando duas abordagens principais: a fixação direta ao solo por cravação do suporte tubular ou o uso de uma base pesada, como blocos de concreto, que proporcionam estabilidade sem necessidade de escavação.

Também estão sendo avaliadas as melhores soluções para proteção da fiação, que deverá ser acomodada em eletrodutos ou conduítes resistentes a intempéries e impactos,

bem como para a proteção da sonda de umidade do solo, que demanda isolamento contra danos físicos, movimentação do solo e degradação por umidade ou corrosão.

Adicionalmente, a estrutura prevê a possibilidade de instalação de antenas externas, com o objetivo de ampliar a cobertura e melhorar a qualidade do sinal no campo.

Outro ponto que permanece em desenvolvimento é a avaliação da necessidade de sistemas passivos ou ativos de refrigeração para as caixas herméticas, considerando os riscos associados ao superaquecimento(LEA, 2020, p. 10). Nesse contexto, também está em análise a viabilidade da instrumentação com sensores de temperatura interna para monitoramento das condições operacionais e a realização de testes específicos de segurança nas baterias, especialmente quanto aos riscos de incêndio, sobreaquecimento e falhas catastróficas em ambientes severos.

As modificações arquiteturais do sistema na terceira iteração do processo de prototipação estão representadas no diagrama de contexto do protótipo da Iteração 3 (Figura 6).

5.7 Iteração 4

A revisão sistemática da literatura, a modelagem conceitual e a análise de segurança buscaram embasamento na literatura especializada em IoT, de modo a orientar a construção de um sistema fundamentado em evidências, em conformidade com os princípios da Engenharia de Software Experimental.

Entretanto, o desenvolvimento deste projeto evidenciou uma série de desafios arquiteturais que iam além das soluções pontuais de comunicação entre dispositivos e servidores. Questões como sincronização e consistência dos dados em caso de falhas de conectividade, replicação seletiva de dados entre borda e nuvem, autenticação entre componentes descentralizados e coordenação de tarefas assíncronas passaram a demandar fundamentos mais amplos, próprios do campo dos sistemas distribuídos.

Essa constatação motivou a realização de um estudo exploratório focado nos fundamentos teóricos dos sistemas distribuídos. Dentre as obras analisadas, adotou-se como principal referência a quarta edição da obra de Van Steen e Tanenbaum (STEEN; TANENBAUM, 2023), por se tratar de uma fonte atualizada, abrangente e amplamente reconhecida na área.

Segundo os autores, “a maioria dos sistemas computacionais modernos é, na prática, um sistema distribuído” (STEEN; TANENBAUM, 2023, p. 1), o que reforça a importância de compreendê-los não como exceção, mas como a base sobre a qual soluções IoT contemporâneas estão inevitavelmente construídas. Essa abordagem permite compreender a arquitetura proposta como parte de um sistema distribuído mais amplo, no qual dispositivos, servidores locais e infraestrutura em nuvem atuam de forma especializada e interdependente.

Essa definição ampla de sistemas distribuídos é coerente não apenas com autores contemporâneos, como Van Steen e Tanenbaum (STEEN; TANENBAUM, 2023), mas

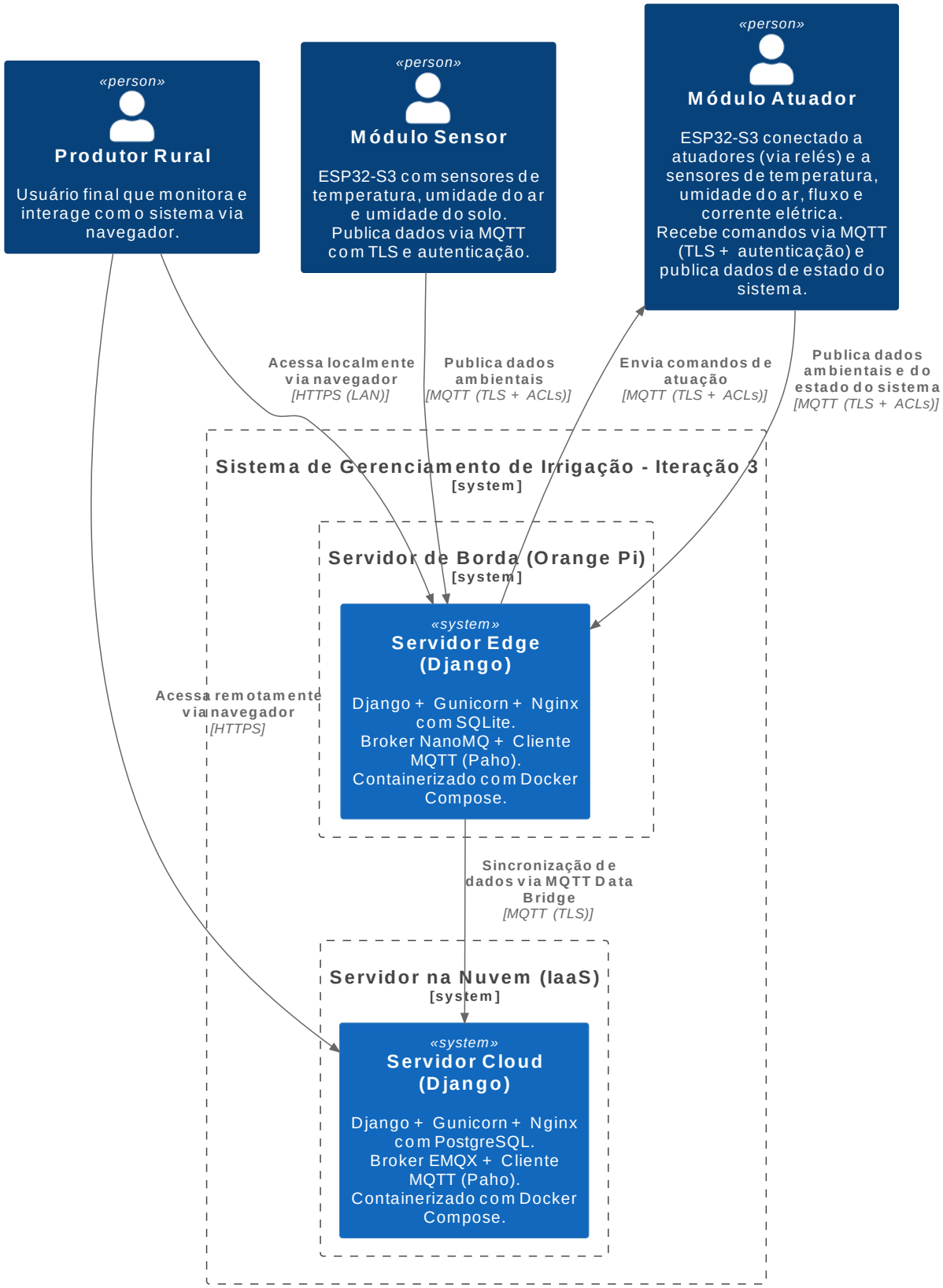

Figura 6 – Diagrama de contexto do protótipo da Iteração 3

também é validada por obras anteriores, como a quinta edição de Coulouris et al. (COULOURIS et al., 2013, p. 1), que definem sistemas distribuídos como aqueles em que "os componentes localizados em computadores interligados em rede se comunicam e coordenam suas ações apenas passando mensagens".

Diferentemente da obra de Van Steen e Tanenbaum (STEEN; TANENBAUM, 2023), Coulouris et al. (COULOURIS et al., 2013) não utiliza explicitamente o termo "Internet das Coisas". Contudo, o conceito está presente de forma recorrente, como no exemplo do *gerenciamento ambiental*, no qual sensores em rede monitoram variáveis ambientais e alimentam sistemas capazes de gerar alertas precoces e apoiar respostas a desastres naturais (COULOURIS et al., 2013, p. 4).

Além disso, os autores dedicam uma seção significativa à *computação móvel e ubíqua* (Capítulo 19), destacando que os avanços na miniaturização de dispositivos e na interconectividade sem fio conduzem à integração de pequenos dispositivos, desde sensores embarcados em eletrodomésticos até smartphones, como componentes ativos de sistemas distribuídos (COULOURIS et al., 2013, p. 10–11).

Dessa forma, a própria evolução das aplicações descritas por Coulouris et al. (COULOURIS et al., 2013) demonstra que a IoT se insere, de maneira natural, no escopo dos sistemas distribuídos, reforçando a atualidade e a relevância dos fundamentos teóricos empregados neste trabalho.

No que se refere à terminologia adotada para descrever a arquitetura composta por *nuvem*, *borda* e dispositivos IoT (*things*) é respaldada por Van Steen e Tanenbaum (STEEN; TANENBAUM, 2023, p. 52), que descrevem essa organização hierárquica como uma evolução dos sistemas distribuídos, impulsionada pelo crescimento da *Internet das Coisas* e pela integração com serviços em nuvem. Segundo os autores, essa arquitetura surge da necessidade de conectar dispositivos locais a serviços distribuídos em larga escala, por meio de redes de longa distância.

Essa organização responde diretamente aos desafios clássicos dos sistemas distribuídos, discutidos em profundidade na mesma obra (STEEN; TANENBAUM, 2023, p. 53), os quais decorrem de suposições equivocadas frequentemente adotadas no desenvolvimento de sistemas, como presumir que a rede é confiável, segura, homogênea, de latência zero e com largura de banda infinita. Assim, o modelo *cloud-edge-things* não apenas reflete uma tendência arquitetural contemporânea, mas também constitui uma estratégia prática para enfrentar as limitações inerentes à comunicação distribuída, à escalabilidade e à heterogeneidade dos ambientes computacionais atuais.

Os próprios autores reconhecem que essa terminologia ainda é objeto de debate, coexistindo com termos como *computação em névoa* (*fog computing*) e diferentes interpretações sobre os papéis e os limites da *edge computing* (STEEN; TANENBAUM, 2023, p. 65). Como ocorre frequentemente na área de sistemas distribuídos, eles destacam que leva alguns anos até que os debates sobre nomenclatura e escopo se estabilizem (STEEN;

TANENBAUM, 2023, p. 100). Apesar disso, adotam uma visão ampla e pragmática da arquitetura *cloud-edge-things*, considerando-a adequada para descrever a organização de sistemas que interligam dispositivos IoT, infraestrutura de borda e serviços em nuvem, alinhada aos desafios atuais de latência, disponibilidade, escalabilidade e processamento distribuído (STEEN; TANENBAUM, 2023, p. 100–104).

5.7.1 Estudo Exploratório sobre Sistemas Distribuídos

O estudo exploratório sobre sistemas distribuídos permitiu validar e aprofundar a compreensão de diversos aspectos arquiteturais do sistema, incluindo aqueles previamente identificados nas seções anteriores. Entretanto, o principal desafio identificado no projeto foi garantir a consistência dos dados durante o processo de replicação em cenários de falha de conexão.

Para essa finalidade, adotou-se inicialmente o *NanoMQ* como broker MQTT na borda, utilizando o recurso de *data bridge* para encaminhar automaticamente os dados ao broker na nuvem (EMQX). No entanto, testes práticos demonstraram que essa solução era sensível a interrupções na conectividade, falhas momentâneas resultavam na perda de mensagens e exigiam o reenvio manual dos dados.

Embora o MQTT seja um protocolo leve e eficiente para comunicação em tempo real, ele não oferece garantias robustas de entrega de mensagens assíncronas em cenários em que os clientes estão desconectados do broker, como em redes instáveis ou sujeitas a quedas prolongadas de conexão. Seus mecanismos de QoS garantem a entrega apenas enquanto há uma sessão ativa entre cliente e broker(LEA, 2020, p. 390 - 391).

Na ausência de conexão, a retenção e o reenvio de mensagens dependem de funcionalidades adicionais, esperava-se que o *data bridge* dos brokers utilizados, aliado a mecanismos locais de persistência e cache, suprisse essas limitações(LEA, 2020, p. 391, 392, 396). Na prática, essa abordagem se mostrou ineficaz, comprometendo a confiabilidade do sistema.

Van Steen e Tanenbaum descrevem sistemas que implementam mecanismos capazes de lidar com este cenário, utilizando modelos baseados em comunicação assíncrona e persistente. De acordo com os autores, a comunicação assíncrona caracteriza-se pela capacidade de um emissor continuar sua execução imediatamente após enviar uma mensagem, uma vez que essa mensagem é armazenada temporariamente pelo middleware assim que é submetida (STEEN; TANENBAUM, 2023, p. 191). Na comunicação persistente, o middleware mantém a mensagem armazenada pelo tempo necessário até que ela seja entregue ao receptor, mesmo que remetente e destinatário não estejam ativos simultaneamente (STEEN; TANENBAUM, 2023, p. 191). Esse modelo é viabilizado por sistemas de filas de mensagens, nos quais as aplicações se comunicam inserindo mensagens em filas específicas, que são encaminhadas por servidores intermediários até o destino, independentemente de este estar ativo no momento do envio (STEEN; TANENBAUM, 2023,

p. 220).

Esses conceitos apresentam uma correspondência direta com os princípios de desacoplamento espacial e temporal, conforme definido por Coulouris et al. O desacoplamento espacial ocorre quando os emissores e receptores não precisam conhecer suas identidades ou localizações mutuamente, sendo suficiente interagir por meio de um intermediário, como uma fila de mensagens. Já o desacoplamento temporal refere-se à capacidade de comunicação mesmo quando remetente e destinatário não estão ativos simultaneamente, sendo possível devido à persistência das mensagens no middleware até que possam ser entregues (COULOURIS et al., 2013, p. 230–232).

O protocolo Advanced Message Queuing Protocol (AMQP) surgiu como uma proposta para padronizar esse tipo de comunicação, descrevendo detalhadamente como as mensagens são transferidas entre os nós. Contudo, o protocolo não estabelece diretrizes sobre como os brokers devem ser organizados, como as políticas de roteamento devem ser implementadas ou como ocorre o armazenamento e a replicação das mensagens (STEEN; TANENBAUM, 2023, p. 231).

O RabbitMQ, por sua vez, é uma implementação que se baseia no AMQP. Esse broker complementa as limitações do protocolo ao introduzir elementos como as exchanges, que são responsáveis por receber mensagens dos produtores e direcioná-las para uma ou mais filas, sem armazená-las (STEEN; TANENBAUM, 2023, p. 231). Cabe ao broker, portanto, implementar funcionalidades críticas não contempladas pelo protocolo, tais como armazenamento de mensagens, lógica de roteamento, autenticação, autorização e formação de clusters, garantindo uma comunicação assíncrona, persistente e resiliente, adequada às demandas de replicação de dados em sistemas distribuídos.

5.7.2 Implementação

Na implementação da Iteração 4, a comunicação entre a borda e a nuvem foi modificada, substituindo os brokers MQTT por um sistema baseado em filas de mensagens AMQP, o que garante desacoplamento temporal e espacial entre os componentes. Diferentemente do MQTT, no qual publicadores e assinantes precisam estar conectados simultaneamente, o protocolo AMQP permite que os clientes não estejam conectados no momento da publicação ou consumo, sendo naturalmente resiliente a interrupções de conexão(LEA, 2020, p. 388, 426, 428).

Para essa implementação, foi adotado o RabbitMQ, que oferece suporte nativo ao protocolo AMQP, proporcionando uma infraestrutura robusta para filas de mensagens distribuídas. A comunicação entre os *brokers* da nuvem e da borda foi viabilizada por meio de dois *plugins* nativos do RabbitMQ:

- **rabbitmq_shovel**: responsável por encaminhar mensagens entre as instâncias do *broker* localizadas na borda e na nuvem, permitindo replicação de dados de forma

contínua e transparente.

- **rabbitmq_shovel_management**: *plugin* auxiliar que fornece métricas e informações detalhadas sobre o estado das conexões do *Shovel*, sendo fundamental para o monitoramento da conectividade entre os ambientes distribuídos e para a geração de diagnósticos em caso de falhas.

Em complemento, foi necessário incorporar o Celery, um sistema de filas assíncronas para Python, que, além de se integrar ao RabbitMQ via AMQP, também permite que os processos executados pelos seus workers sejam tratados de forma assíncrona, melhorando significativamente a escalabilidade e o desempenho do sistema.

O Celery, por sua vez, requer um backend de resultados para armazenar o estado das tarefas, garantir persistência em caso de falhas e possibilitar o rastreamento de execução. Para essa função, foi escolhido o Redis, um banco de dados chave-valor leve, eficiente e amplamente utilizado como backend de resultados em arquiteturas assíncronas.

Algumas tarefas do sistema precisam ser executadas periodicamente. Para isso, foi utilizado o **Celery Beat**, um agendador que dispara rotinas em intervalos definidos. Ele foi incorporado ao projeto em um *container* próprio, garantindo organização e isolamento dos demais serviços.

Apesar da mudança na comunicação entre a borda e a nuvem, o broker NanoMQ foi mantido no servidor de borda para intermediar a comunicação via MQTT com os dispositivos IoT, pois esse protocolo continua sendo o mais adequado para esse tipo de cenário, caracterizado por redes de baixa largura de banda, alta latência e dispositivos com recursos limitados. Além disso, o *NanoMQ* possui um plugin chamado `http_server`, que disponibiliza uma *API HTTP* com informações sobre os clientes atualmente conectados ao *broker*. Essa API é utilizada pelo módulo de monitoramento do sistema desenvolvido.

Adicionalmente, o cliente MQTT, implementado com a biblioteca Paho-MQTT, foi encapsulado em um container Docker próprio. Essa decisão favorece o desacoplamento entre a comunicação via MQTT e a aplicação Django, permitindo que o envio e a recepção de mensagens sejam tratados de forma assíncrona e resiliente, sem comprometer a disponibilidade ou o desempenho da aplicação principal.

Estas modificações estão representadas no diagrama de contexto do protótipo da Iteração 4 (Figura 7).

5.8 Fase de Consolidação da Solução

Ao final do processo, foi definida e implementada a arquitetura final do sistema, com base nos estudos teóricos, nos testes práticos e nas decisões técnicas tomadas ao longo do desenvolvimento. O resultado é uma solução distribuída, estruturada em camadas de borda e nuvem, projetada para operar de forma resiliente em ambientes com

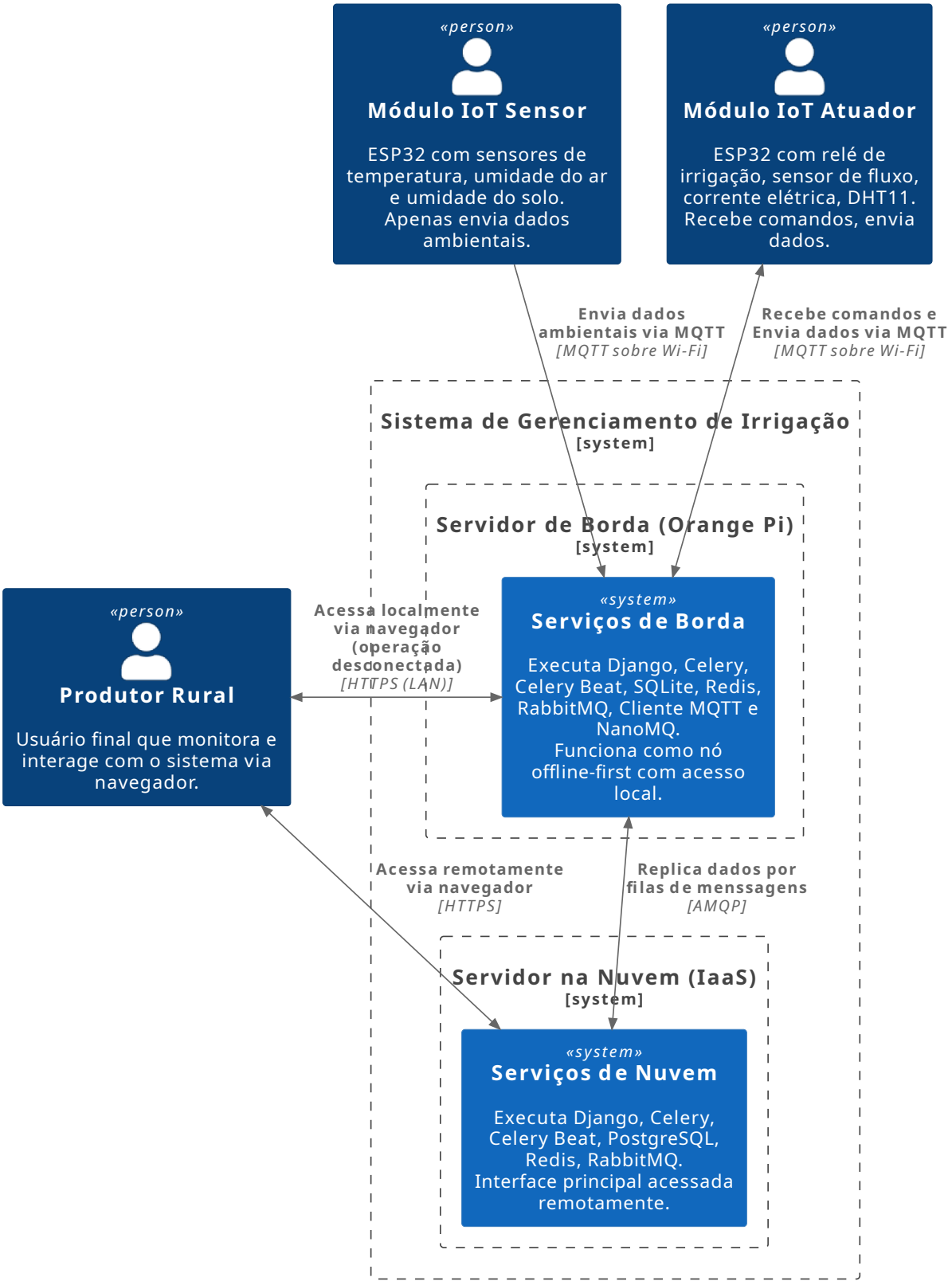

Figura 7 – Diagrama de contexto do protótipo da Iteração 4

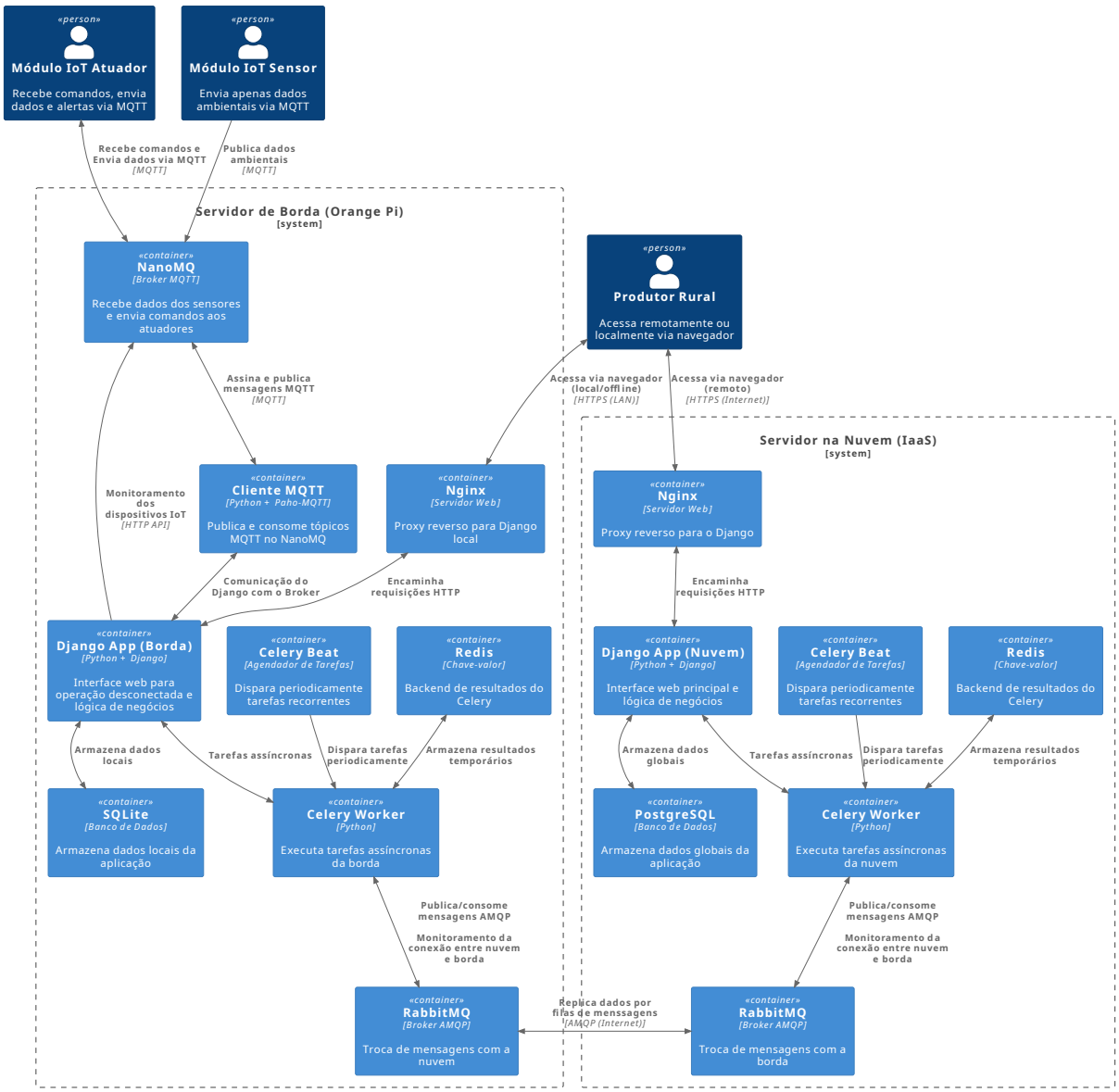

Figura 8 – Diagrama de contêineres da solução proposta

conexão limitada e falhas de energia, mantendo escalabilidade, segurança e facilidade de manutenção.

A arquitetura do sistema está representada no diagrama de contêineres da solução proposta (Figura 8).

A consolidação também envolveu a definição do mecanismo de replicação de dados entre as camadas de borda e nuvem. Esse mecanismo foi implementado de forma assíncrona, utilizando filas específicas no broker RabbitMQ para isolar os fluxos de comunicação em cada direção.

Quando eventos de interesse ocorrem na borda, como a criação ou atualização de registros de leitura ou agendamento, os dados são serializados e enviados a uma fila nomeada, como `to_cloud`. Uma tarefa Celery consome essa fila e realiza a persistência

dos dados na camada de nuvem. A replicação no sentido oposto (nuvem para borda) segue o mesmo princípio, utilizando a fila `to_edge`.

Para garantir a integridade dos dados e prevenir estados inconsistentes durante condições de rede instável, foi implementado um controle de versão baseado em campos `versao_borda` e `versao_nuvem` em cada modelo replicável. No momento da replicação, o sistema compara as versões locais e remotas, aplicando regras de precedência para decidir se uma atualização deve ser aplicada, ignorada ou sobreposta.

Além disso, a lógica de replicação está desacoplada dos modelos principais, sendo organizada em módulos reutilizáveis que facilitam a inclusão de novos modelos no processo. Essa abordagem modular permite que a replicação seja tratada como uma infraestrutura transversal, e não como lógica acoplada aos fluxos de aplicação, aumentando a manutenibilidade do sistema.

6 PROVA DE CONCEITO

Uma prova de conceito do projeto foi realizada por meio do desenvolvimento de uma aplicação responsável por integrar e gerenciar o fluxo de dados entre as camadas de nuvem, borda e dispositivos IoT, conforme descrito nas fases anteriores. O sistema implementa, de forma parcial ou completa, todos os requisitos funcionais identificados, possibilitando o agendamento de irrigação, o monitoramento do sistema, a detecção de falhas de conectividade entre nuvem e borda, bem como a verificação da conectividade dos dispositivos IoT. Além disso, realiza o registro de eventos de telemetria e dos dados obtidos pelos sensores conectados.

Em relação aos requisitos não funcionais, o sistema conta com uma interface web responsiva, compatível com computadores e smartphones, desenvolvida com foco na redução da carga cognitiva do operador. O sistema foi projetado para ser resiliente a falhas de energia e conectividade, persistindo dados localmente na borda e realizando a replicação automática após a reconexão.

A aplicação web desenvolvida com o framework Django foi estruturada de forma modular, com a criação de aplicativos internos (apps) que organizam a lógica de negócio e facilitam a manutenção e expansão do sistema. Cada app concentra funcionalidades específicas, permitindo uma separação clara de responsabilidades dentro do projeto.

Os principais apps implementados são:

- **Core**: responsável pela gestão das entidades centrais do sistema, reunindo tanto os modelos de usuários e clientes quanto os elementos técnicos da infraestrutura
- **Telemetria**: gerencia o recebimento e armazenamento de dados provenientes dos sensores instalados nos dispositivos IoT;
- **Irrigação**: controla os agendamentos e a lógica de acionamento dos atuadores responsáveis pela irrigação;
- **Monitoramento**: responsável por armazenar os dados de verificação da conectividade entre a nuvem e a borda, bem como informações sobre os clientes MQTT conectados ao broker na borda. Além disso, calcula métricas de saúde dessas conexões com base nas coletas mais recentes.

Embora os mesmos aplicativos Django estejam presentes tanto na instância de borda quanto na de nuvem, suas funcionalidades foram adaptadas para atender aos distintos contextos operacionais de cada camada. Essa diferenciação funcional é evidenciada nos diagramas C4 de nível 3, apresentados nas Figuras 9, que detalha os componentes da nuvem, e 10, que representa os componentes da borda.

Em muitos casos, isso envolve a replicação de dados por meio de filas de mensagens AMQP do RabbitMQ, cuja intermediação no Django é realizada por tarefas assíncronas

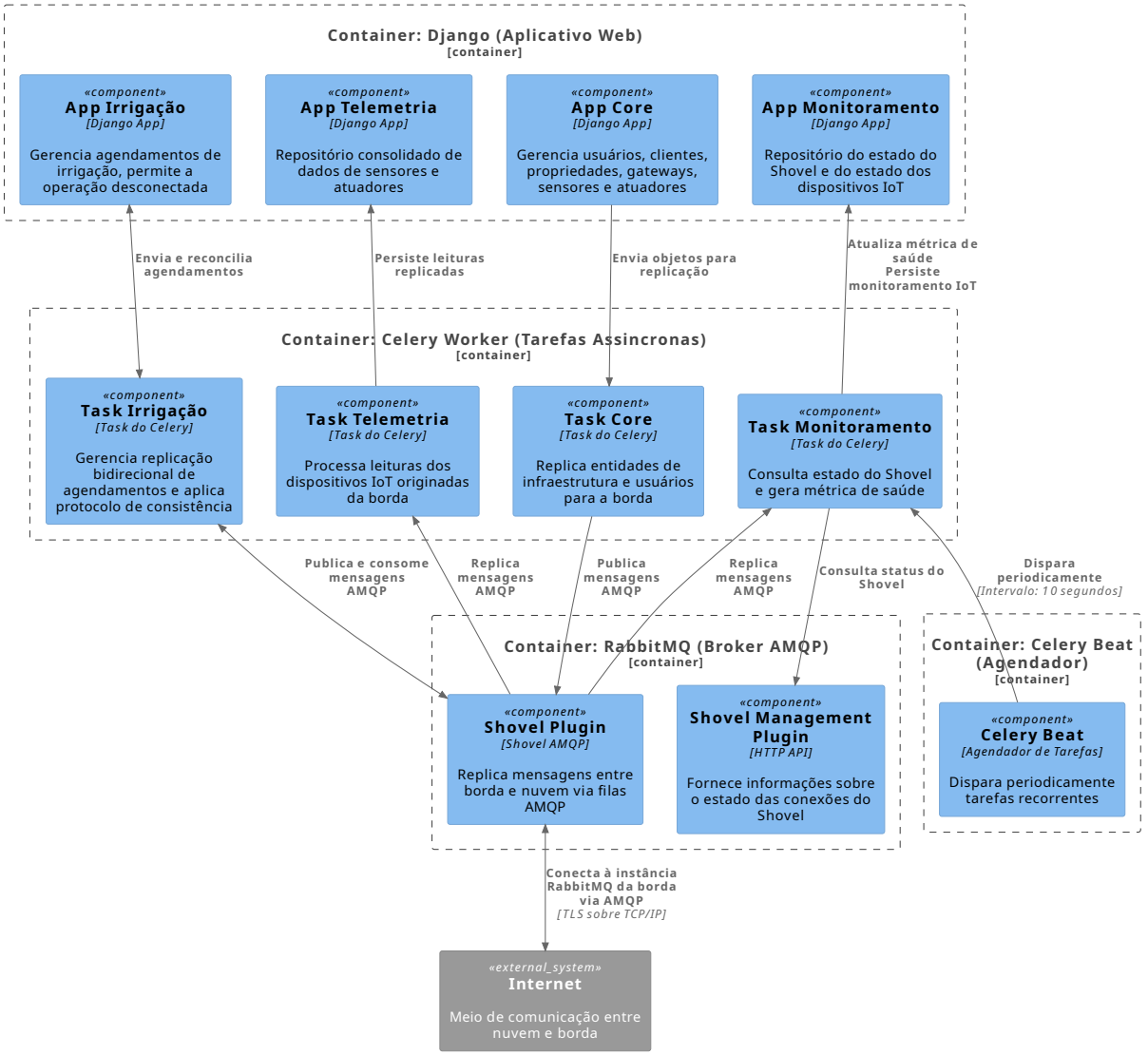

Figura 9 – Diagrama C4 Nível 3 - Componentes internos do servidor em nuvem

gerenciadas com o *Celery*. Esse processo de replicação é detalhado nas seções a seguir, que apresentam os fluxos específicos de cada aplicativo do sistema, acompanhados por diagramas de sequência UML ilustrando a comunicação entre os componentes.

6.1 App Core

O app *core* reúne as entidades centrais do sistema, responsáveis por estruturar tanto o modelo organizacional quanto os elementos técnicos da infraestrutura.

Modelos como `User` e `Cliente` representam os diferentes tipos de usuários e seus vínculos com a instância do sistema, permitindo o controle de acesso baseado em papéis (como proprietário, gerente e funcionário) e o isolamento lógico por cliente. Já os modelos `Propriedade`, `Gateway`, `Atuador` e `Sensor` representam os elementos técnicos que compõem a arquitetura distribuída do sistema, conectando os dispositivos IoT ao software

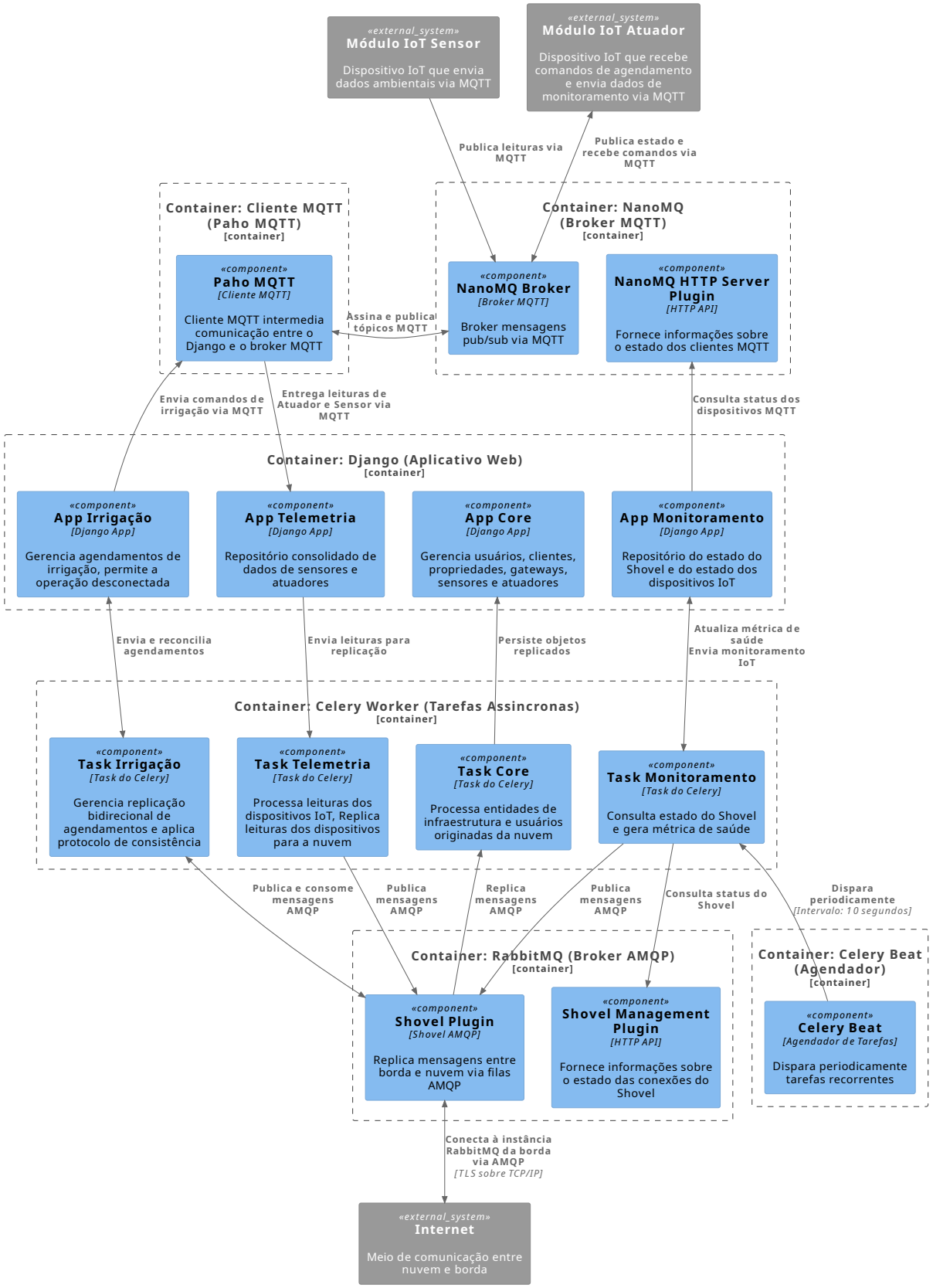

Figura 10 – Diagrama C4 Nível 3 - Componentes internos do servidor de borda

por meio de uma estrutura hierárquica.

A decisão de agrupar esses modelos em um único app se justifica pela interdependência entre eles: usuários estão associados a clientes, que por sua vez gerenciam propriedades equipadas com gateways, sensores e atuadores. Assim, o app *core* serve como base comum e estrutural para os demais módulos do sistema, consolidando as informações essenciais para o seu funcionamento.

A lógica de replicação do *core* segue o sentido da nuvem para a borda. Como as alterações nesses modelos são permitidas apenas na instância da nuvem, essa parte da aplicação não é diretamente manipulada pelo usuário final. Essa estratégia simplifica o gerenciamento da arquitetura, dispensando a replicação bidirecional e a necessidade de resolução de conflitos, ao mesmo tempo em que garante a consistência dos dados essenciais em ambas as camadas do sistema.

O diagrama de sequência representa o fluxo de dados (Figura 11).

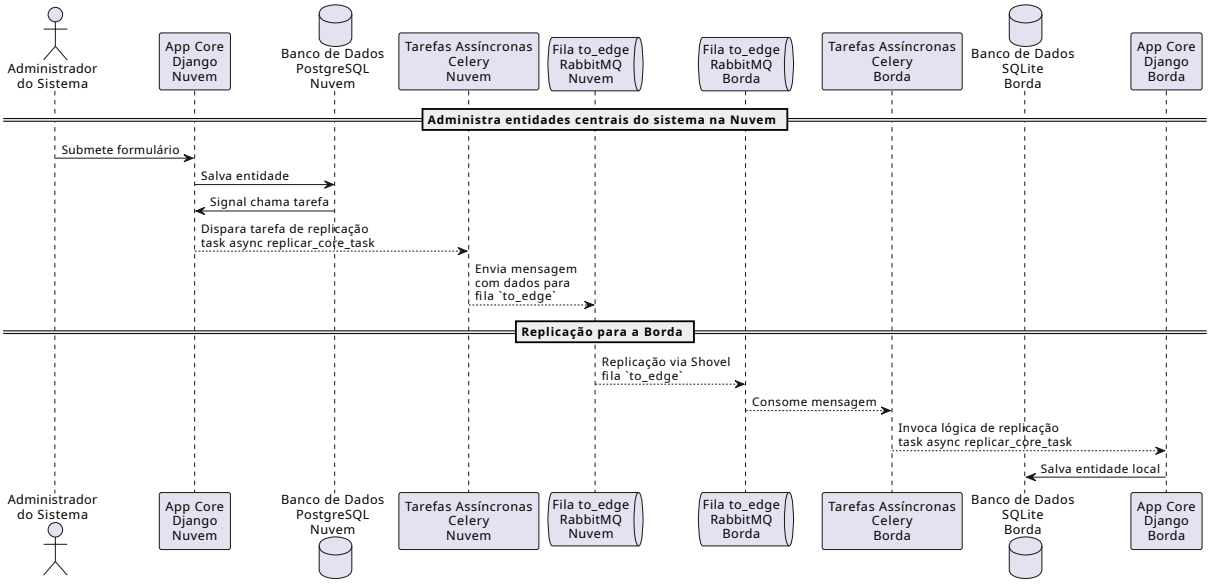

Figura 11 – Diagrama de sequência - Replicação de entidades centrais do sistema (core)

6.2 App Telemetria

O app *telemetria* é responsável por gerenciar o fluxo de dados coletados pelos sensores e atuadores instalados em campo. Esse fluxo é unidirecional: os dispositivos IoT enviam as leituras via protocolo MQTT para o servidor de borda, onde os dados são recebidos, processados e armazenados localmente.

Após essa persistência inicial, as informações são replicadas de forma assíncrona para a instância na nuvem utilizando o protocolo AMQP, garantindo sua centralização, segurança e disponibilidade para análises posteriores.

Como se tratam de dados gerados automaticamente pelos sensores, esses registros não são manipulados diretamente pelos usuários finais ou administradores do sistema.

Dessa forma, o app *telemetria* atua na borda como ponto de entrada dos dados brutos provenientes do ambiente físico, enquanto na nuvem cumpre a função de repositório consolidado.

O diagrama de sequência representa o fluxo de dados (Figura 12).

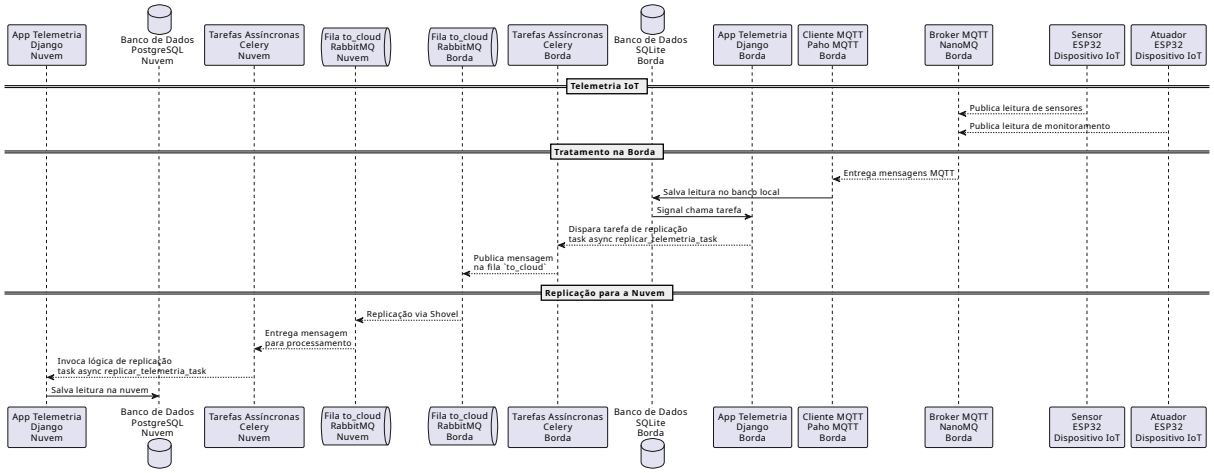

Figura 12 – Diagrama de sequência - Fluxo de replicação das leituras de telemetria

6.3 App Irrigação

O app *irrigação* é responsável por controlar os agendamentos e a lógica de acionamento dos atuadores envolvidos no processo de irrigação. Como esses agendamentos são definidos diretamente pelo usuário final, o sistema precisa oferecer suporte à operação desconectada, garantindo o funcionamento mesmo em situações de instabilidade ou perda de conexão com a internet, conforme previsto neste projeto.

Por essa razão, a administração desse módulo pelo usuário final está disponível tanto na instância da nuvem quanto na instância local do servidor de borda. Em condições normais, os agendamentos são gerenciados por meio da interface disponibilizada na nuvem, acessada via internet, chamada aqui de ‘operação conectada‘.

O diagrama de sequência representa o fluxo de dados em uma operação conectada (Figura 13).

No entanto, em situações de desconexão, o sistema permite o acesso local à instância replicada na borda, garantindo a continuidade operacional. Essa situação é caracterizada como ‘operação desconectada‘ (COULOURIS et al., 2013, p. 770).

O diagrama de sequência representa o fluxo de dados em uma operação desconectada (Figura 14).

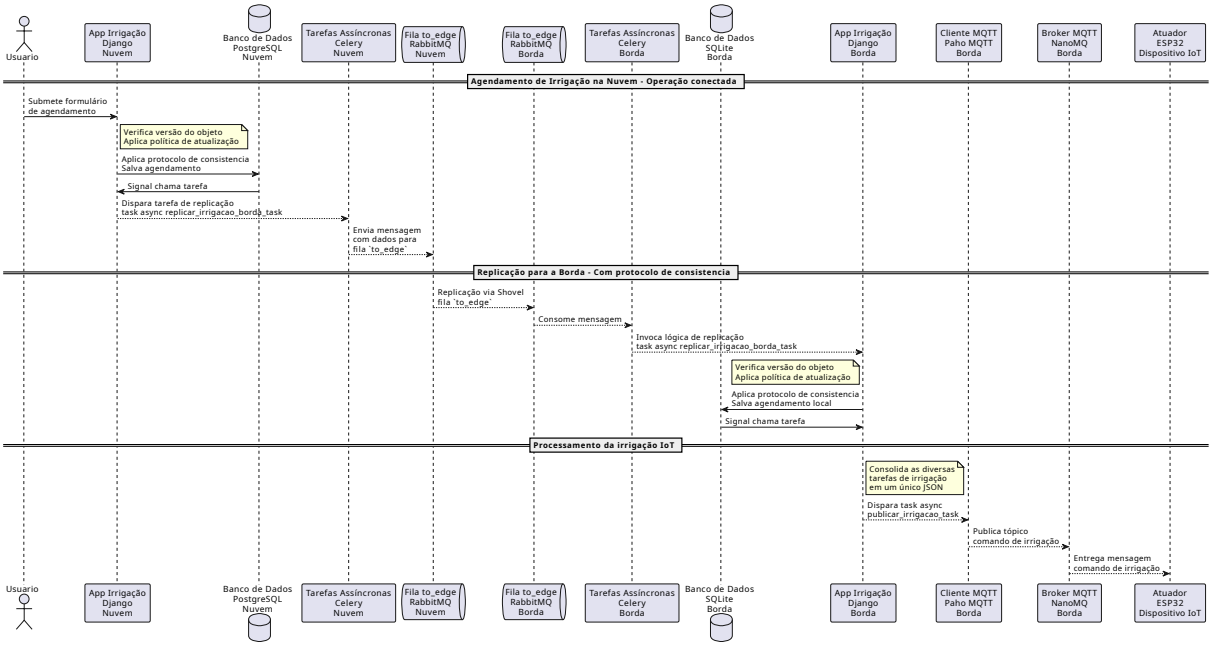

Figura 13 – Diagrama de sequência - Agendamento de irrigação com operação conectada

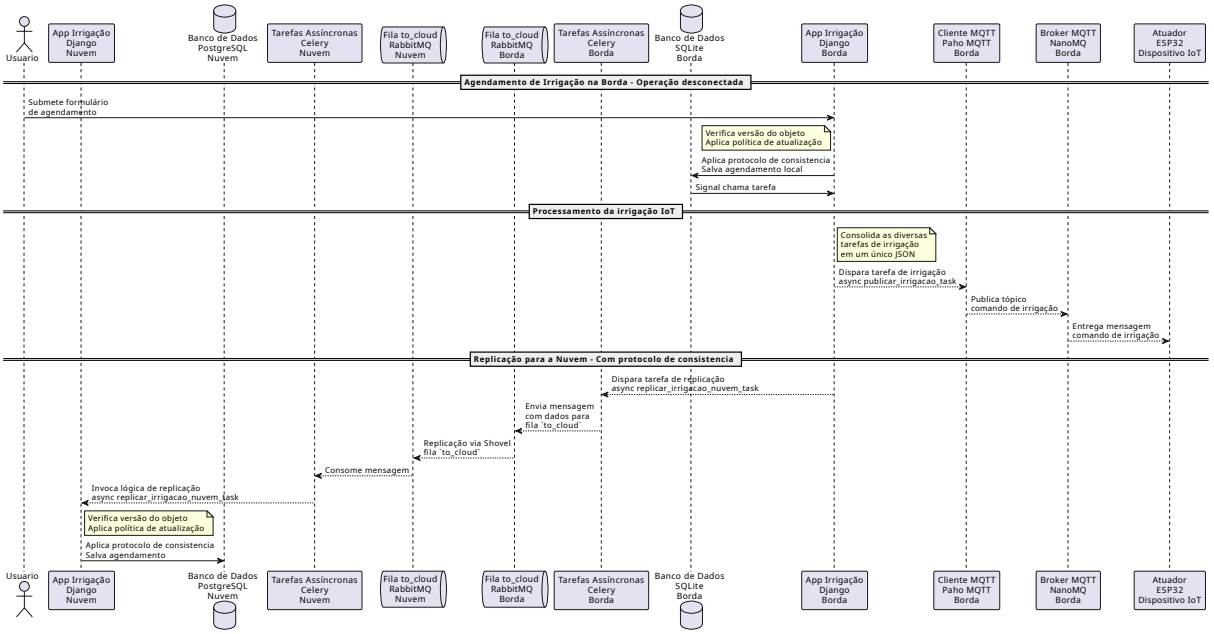

Figura 14 – Diagrama de sequência - Agendamento de irrigação com operação desconectada

6.3.1 Aplicação do protocolo de consistência eventual

Para viabilizar essa funcionalidade, o app *irrigação* implementa um fluxo de replicação bidirecional entre borda e nuvem. Como os mesmos dados podem ser modificados em ambas as instâncias durante um período de desconexão, o sistema adota um protocolo de consistência simplificado para lidar com conflitos de atualização.

A principal situação de conflito ocorre quando um mesmo agendamento é alterado simultaneamente na borda e na nuvem durante a desconexão. Após a reconexão, ambos os lados tentariam replicar suas versões do mesmo objeto, gerando uma divergência. Para resolver esse impasse, o protocolo define que, em caso de conflito, prevalece a versão da borda, sob a premissa de que os operadores locais possuem maior conhecimento da situação operacional.

Três casos que ilustram essa situação são apresentados a seguir.

6.3.1.1 Criação de novo objeto

Criação de novo objeto durante a desconexão, em que um agendamento é criado em uma das instâncias enquanto a comunicação está indisponível. Após a reconexão, o objeto é replicado normalmente para a outra instância. Como ele ainda não existe no destino, não há conflitos de versionamento, e a operação de criação é concluída com sucesso (Figura 15).

6.3.1.2 Atualização de objetos distintos

Durante o período de desconexão, as instâncias da borda e da nuvem realizam modificações em objetos diferentes, ou seja, não coincidentes. Após a reconexão, cada instância replica seu respectivo objeto para a outra, sem que ocorra conflito de versionamento. Como as alterações envolvem registros distintos, o protocolo de consistência permite a propagação simultânea de ambas as atualizações, garantindo a integridade dos dados em ambas as instâncias (Figura 16).

6.3.1.3 Atualização paralela do mesmo objeto

Durante a desconexão, tanto a borda quanto a nuvem realizam alterações no mesmo objeto. Ao restabelecer a comunicação, surge um conflito de versionamento, uma vez que ambas as instâncias tentam replicar versões divergentes do mesmo registro. Para resolver esse impasse, o protocolo de consistência aplica uma política de precedência, na qual a versão da borda prevalece sobre a da nuvem. Essa decisão é baseada na premissa de que a borda está mais próxima do contexto operacional e, portanto, possui maior autoridade sobre os dados. Assim, a versão local substitui a remota, garantindo a consistência final do sistema (Figura 17).

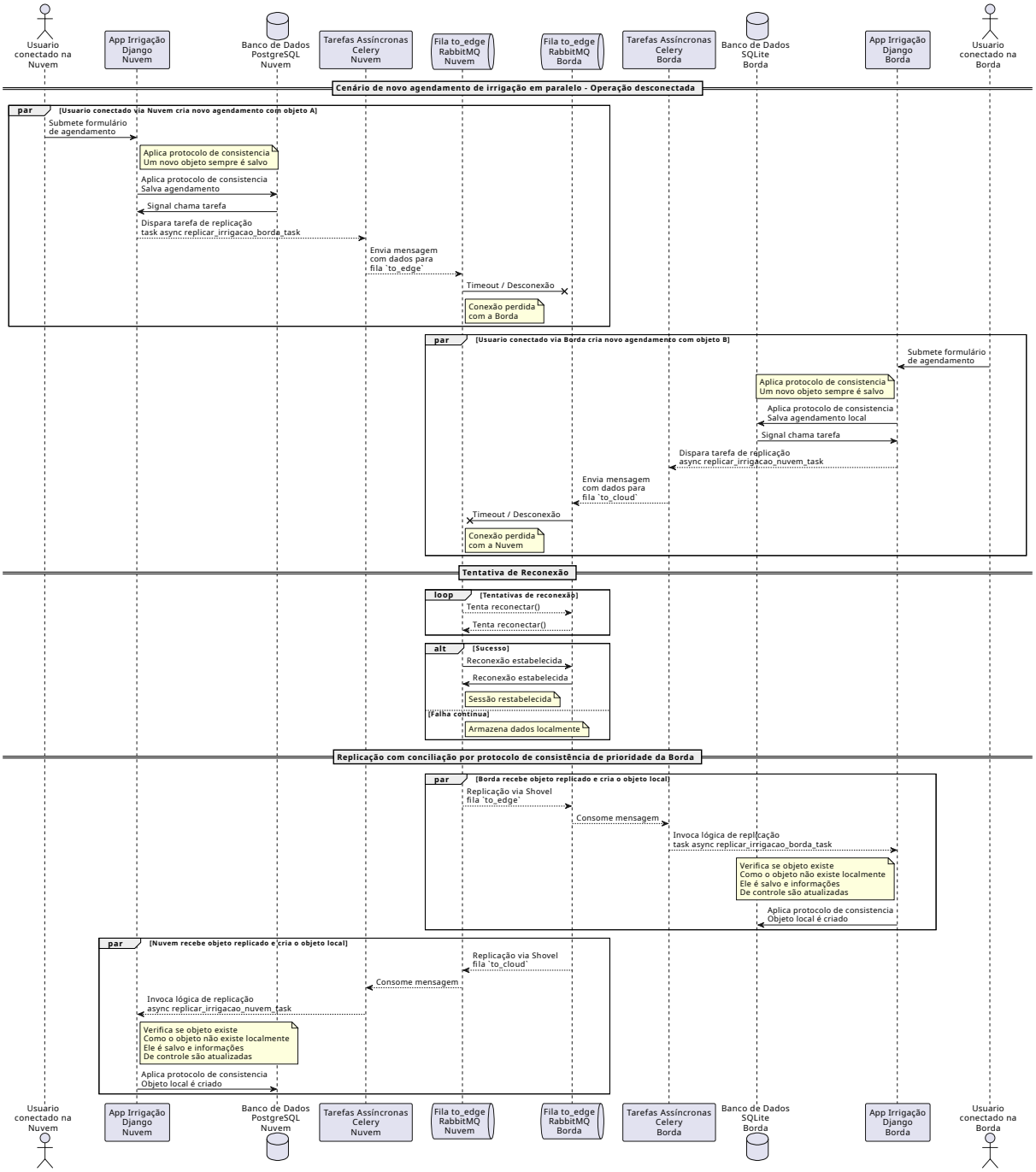

Figura 15 – Diagrama de sequência - Protocolo de consistência aplicado a um novo objeto

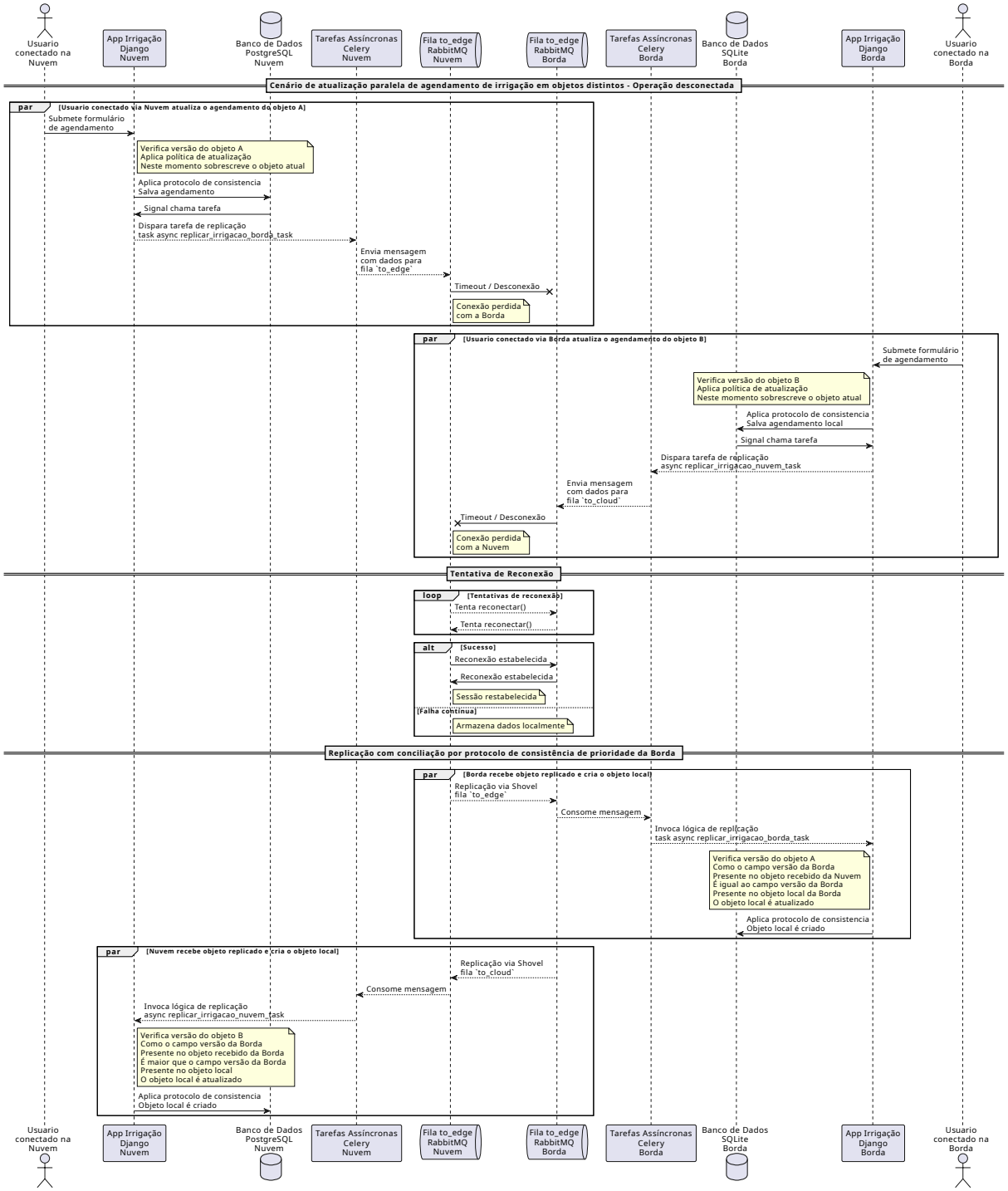

Figura 16 – Diagrama de sequência - Protocolo de consistência aplicado a atualizações de objetos distintos

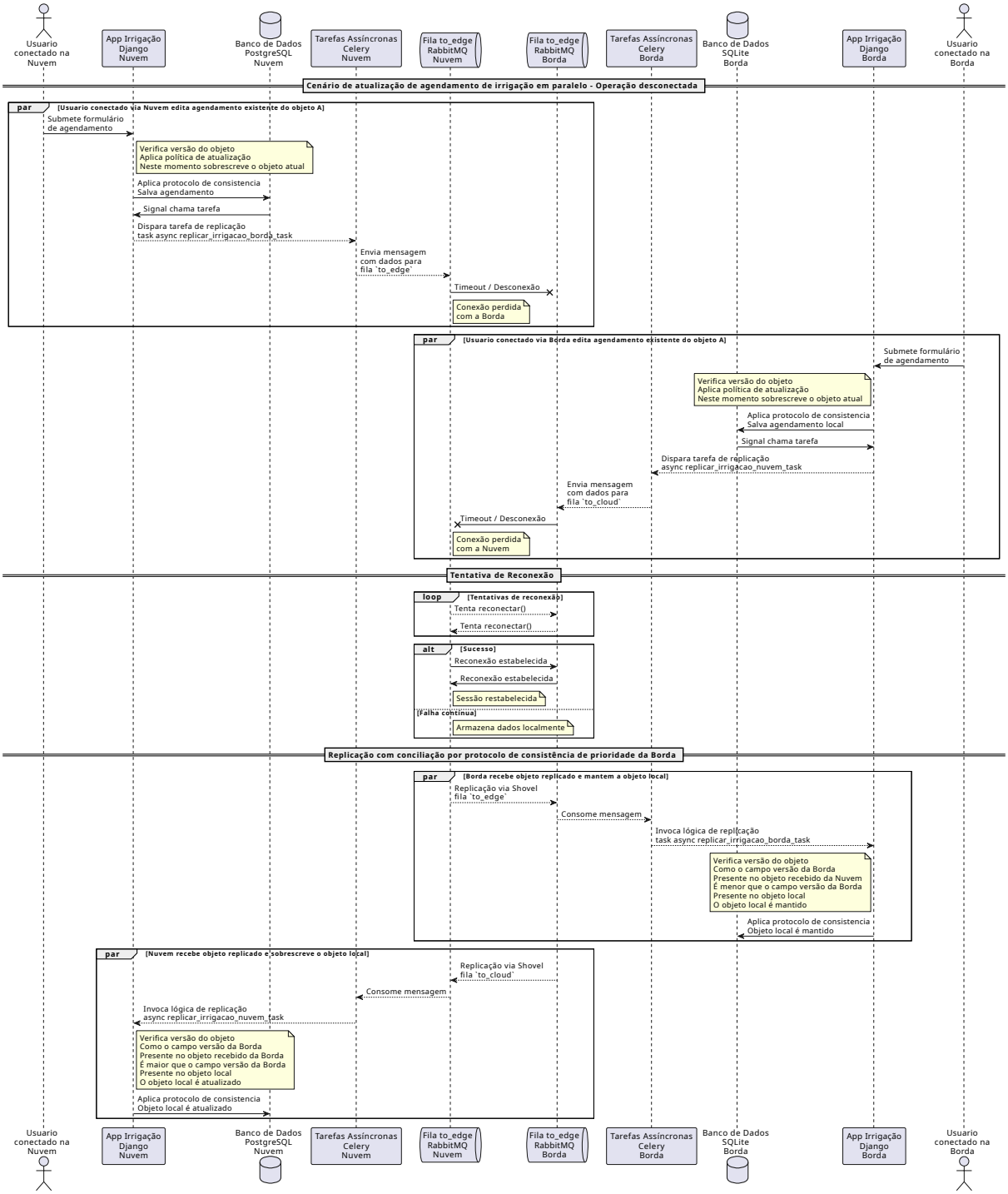

Figura 17 – Diagrama de sequência - Protocolo de consistência aplicado a atualizações paralelas no mesmo objeto

Apesar de ser uma abordagem simples, essa política garante a consistência funcional do sistema e reflete a prioridade da atuação local em cenários de instabilidade.

6.3.1.4 Consolidação dos agendamentos

Além das operações de replicação, o app *irrigação*, na instância de borda, é responsável por consolidar todos os agendamentos registrados em um comando único contendo a rotina diária. Esse comando é gerado a partir da união de múltiplos registros de agendamento e tem como objetivo otimizar a comunicação com o dispositivo atuador. Após a consolidação, o comando é transmitido via protocolo MQTT, permitindo a execução automatizada do plano de irrigação.

6.4 App Monitoramento

O app *monitoramento* é responsável por informar o usuário final sobre o estado da conexão entre a instância de borda e a nuvem. A replicação dos dados entre essas camadas é realizada por meio do plugin *Shovel*, disponibilizado pelo RabbitMQ. Esse plugin permite a criação de filas AMQP integradas entre diferentes brokers e oferece suporte ao monitoramento da conexão entre as instâncias.

O próprio RabbitMQ fornece, por meio de um plugin de gerenciamento, informações sobre o estado da conexão do *Shovel*. A granularidade dessa atualização é definida por uma frequência fixa, configurada no próprio RabbitMQ. Devido à natureza periódica desses dados, sua coleta é realizada por meio de uma tarefa agendada com o *Celery Beat*, o agendador de tarefas recorrentes do Celery. Optou-se por adotar o intervalo de coleta recomendado pela documentação oficial do RabbitMQ, fixado em 10 segundos.

As informações coletadas são armazenadas no app *monitoramento* e utilizadas para gerar uma métrica de saúde da conexão. Essa métrica é baseada nas dez últimas coletas e representa o percentual de sucesso na comunicação entre nuvem e borda. Com base nesse índice, o sistema classifica o estado da conexão da seguinte forma:

- `Desconectado`: quando o índice é igual a 0;
- `Ruim`: quando o índice está entre 1
- `Moderado`: entre 40
- `Bom`: a partir de 70

Essa métrica indica a estabilidade da comunicação entre os servidores e é atualizada periodicamente no mesmo intervalo de coleta dos dados do *Shovel*.

O diagrama de sequência a seguir representa o processo de monitoramento da conexão entre a borda e a nuvem. Esse fluxo é executado em ambas as instâncias, nuvem

e borda, uma vez que a lógica de coleta e avaliação da conexão é idêntica em ambos os contextos operacionais (Figura 18).

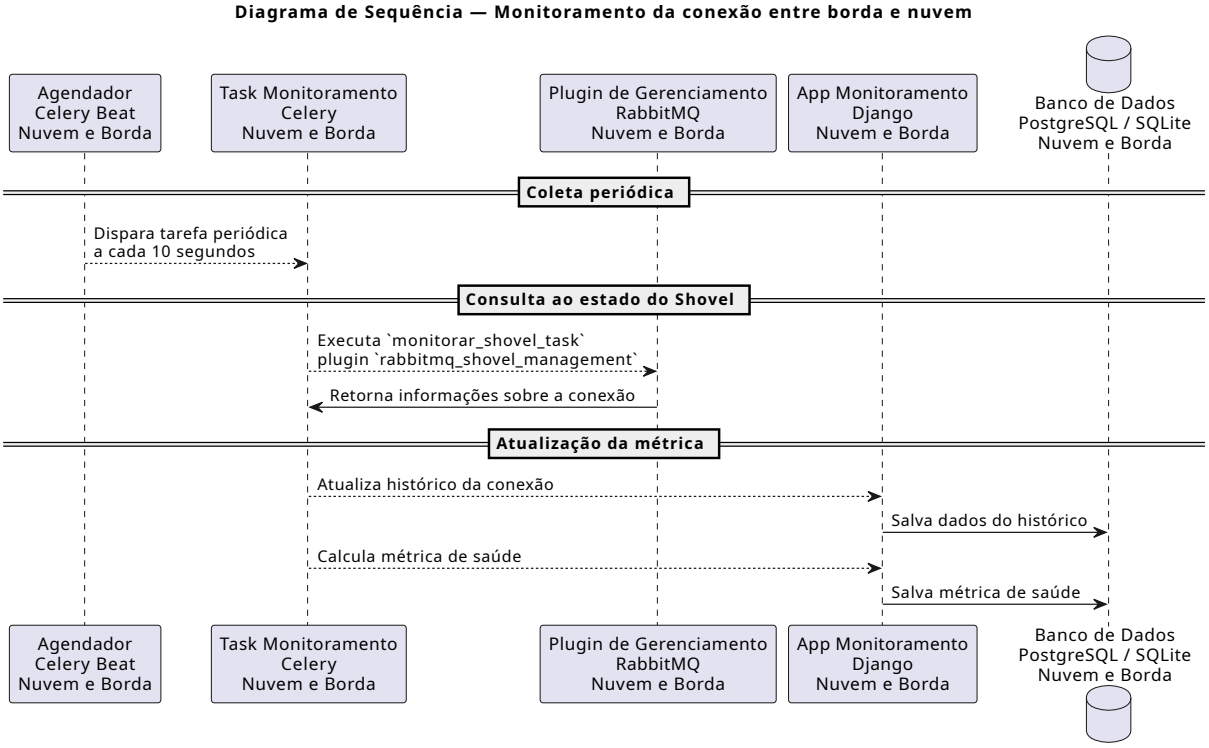

Figura 18 – Diagrama de sequência - Monitoramento da conexão entre borda e nuvem

Além do monitoramento das conexões entre borda e nuvem, o app *monitoramento* também é responsável por acompanhar os clientes MQTT conectados ao broker *NanoMQ*, por meio do plugin *http_server*. Essa funcionalidade é semelhante ao monitoramento do *Shovel*, com a diferença de que sua execução ocorre exclusivamente na instância de borda, uma vez que é onde o *NanoMQ* está instalado, assim como os dispositivos IoT. As informações obtidas localmente são posteriormente replicadas para a nuvem, garantindo visibilidade unificada. Assim como no caso do *Shovel*, o sistema calcula uma métrica de saúde com base na presença e estabilidade da conexão dos clientes MQTT, permitindo avaliar o estado geral da rede de dispositivos.

Com base nessas informações, o sistema pode alternar automaticamente o acesso à aplicação para a instância local na borda, garantindo o funcionamento para os clientes conectados à rede local durante períodos de instabilidade ou desconexão da Internet, por meio do mecanismo de operação desconectada.

Como se trata de um componente de diagnóstico local, o app *monitoramento* não participa da replicação de dados entre a borda e a nuvem. Assim como no app *telemetria*, os registros desse módulo não são manipulados diretamente pelos usuários finais ou administradores do sistema.

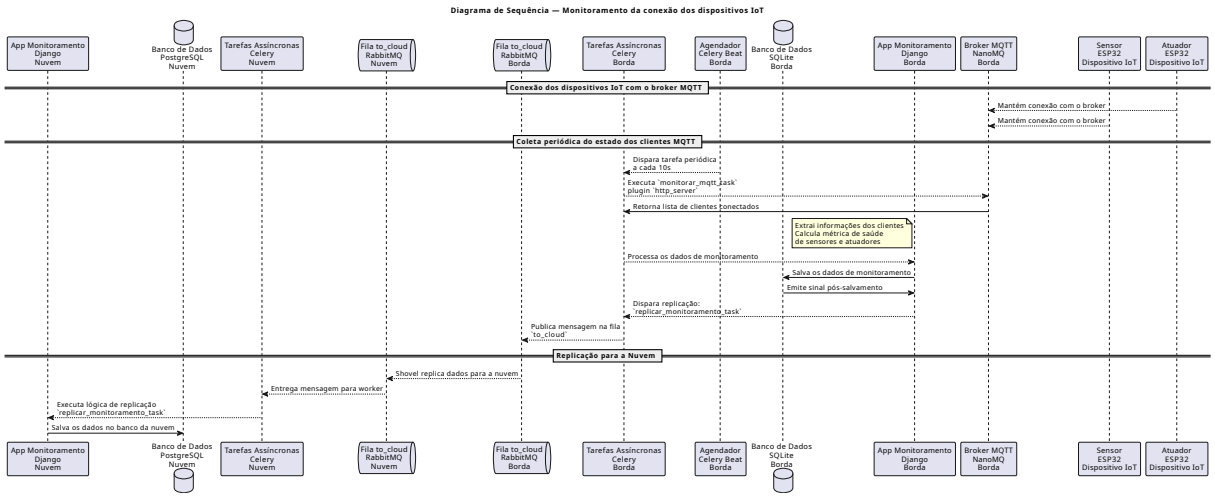

Figura 19 – Diagrama de sequência - Monitoramento das conexões MQTT na borda

O diagrama de sequência a seguir representa o processo de monitoramento das conexões MQTT dos dispositivos IoT ao broker *NanoMQ*. Esse fluxo é executado exclusivamente na instância de borda, pois é onde se encontram tanto o *NanoMQ* quanto os dispositivos conectados. A coleta das informações de conectividade é realizada periodicamente por meio da API do plugin *http_server*, e os dados obtidos são utilizados para calcular uma métrica de saúde dos clientes MQTT. As informações são posteriormente replicadas para a nuvem, permitindo a visualização consolidada do estado da rede de dispositivos (Figura 19).

7 CONCLUSÃO

Este trabalho abordou a complexidade inerente ao desenvolvimento de sistemas distribuídos, especificamente no contexto da Internet das Coisas (IoT), que integra componentes heterogêneos como nuvem, borda e dispositivos embarcados com diversas restrições. O objetivo central foi projetar e desenvolver uma arquitetura resiliente para sistemas de irrigação inteligente, capaz de operar de forma contínua em ambientes rurais, onde falhas de energia e conectividade são desafios recorrentes.

A principal contribuição desta dissertação foi a proposição e validação de uma arquitetura híbrida Nuvem-Borda. Nesta arquitetura, a camada de borda assume um papel fundamental ao prover autonomia computacional local, garantindo a continuidade das operações de irrigação mesmo durante períodos de operação desconectada com a nuvem. Para gerenciar a sincronização de dados em um ambiente propenso a falhas parciais, foram implementados mecanismos de replicação assíncrona bidirecional e um protocolo de consistência eventual que, em caso de conflitos, prioriza as atualizações realizadas na borda, assegurando a integridade do sistema após a reconexão.

A viabilidade da arquitetura foi demonstrada por meio de uma prova de conceito que validou a capacidade do sistema em manter as funcionalidades essenciais durante interrupções, atendendo aos requisitos levantados em campo. O sistema desenvolvido não apenas atende a uma demanda específica de pequenos produtores rurais, frequentemente desassistidos por soluções comerciais de larga escala, mas também oferece um modelo arquitetural escalável e adaptável a outros contextos com desafios de infraestrutura similares.

Apesar dos resultados positivos, foram identificadas diversas oportunidades para trabalhos futuros. Destacam-se a implementação de mecanismos de tolerância a falhas mais avançados, a integração de algoritmos agronômicos para otimizar o uso da água, a melhoria da eficiência energética dos dispositivos e a exploração de protocolos de comunicação de longo alcance, como LoRa e Zigbee. Adicionalmente, a evolução da interface a partir de testes com usuários finais é um passo crucial para a consolidação e adoção prática da solução.

Em suma, esta dissertação apresenta uma contribuição relevante para a construção de sistemas IoT resilientes, estabelecendo uma base sólida para futuras pesquisas e desenvolvimentos na área da agricultura inteligente e de sistemas distribuídos aplicados a ambientes com infraestrutura restrita.

REFERÊNCIAS

ABDIKADIR, N. M. et al. Smart irrigation system. **SSRG International Journal of Electrical and Electronics Engineering**, v. 10, n. 8, p. 224 – 234, 2023. Cited by: 0; All Open Access, Hybrid Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85170286020&doi=10.14445%2f23488379%2fIJEEE-V10I8P122&partnerID=40&md5=c8e04252c2b834a48fde178acb300efe>. Citado na página 36.

ALHARBI, H. A.; ALDOSSARY, M. Energy-efficient edge-fog-cloud architecture for iot-based smart agriculture environment. **IEEE Access**, v. 9, p. 110480–110492, 2021. Citado na página 49.

BENAMEUR, R. et al. Securing iot smart irrigation systems with adapted blockchain based approach. In: **2023 International Conference on Earth Observation and Geo-Spatial Information (ICEOGI)**. [S.l.: s.n.], 2023. p. 1–6. Citado 3 vezes nas páginas 36, 39 e 43.

BOUALI, E.-T. et al. Renewable energy integration into cloud & iot-based smart agriculture. **IEEE Access**, v. 10, p. 1175–1191, 2022. Citado na página 36.

COULOURIS, G. et al. **Sistemas distribuídos: conceitos e projeto**. 5. ed. Porto Alegre: Bookman, 2013. ISBN 978-85-8260-054-2. Citado 4 vezes nas páginas 24, 74, 76 e 85.

EDODI, M. U.; OGIDAN, O. K.; AMUSAN, A. Smart irrigation system: A water and power management approach. In: **2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)**. [S.l.: s.n.], 2022. p. 1–7. Citado na página 36.

ET-TAIBI, B. et al. Cloud and edge based smart agriculture: A real-world deployment. **2024 International Conference on CircuiCloud and Edge based Smart Agriculture: A Real-World Deploymentt, Systems and Communication (ICCSC)**, p. 1–6, 2024. Citado na página 49.

GUTIéRREZ, J. et al. Automated irrigation system using a wireless sensor network and gprs module. **IEEE Transactions on Instrumentation and Measurement**, v. 63, n. 1, p. 166–176, 2014. Citado na página 36.

HERELL, M. D. et al. Gateway, cloud server, and mobile app dashboard in evapotranspiration-based irrigation scheduling system. In: **2021 International Symposium on Electronics and Smart Devices (ISESD)**. [S.l.: s.n.], 2021. p. 1–6. Citado na página 36.

ISLAM, A. et al. Iot based power efficient agro field monitoring and irrigation control system : An empirical implementation in precision agriculture. In: **2018 International Conference on Innovations in Science, Engineering and Technology (ICISET)**. [S.l.: s.n.], 2018. p. 372–377. Citado 2 vezes nas páginas 36 e 39.

JAIN, R. K. Experimental performance of smart iot-enabled drip irrigation system using and controlled through web-based applications. **Smart Agricultural Technology**, v. 4, 2023. Cited by: 6; All Open Access, Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85150181858&doi=10.1016%

2fj.atech.2023.100215&partnerID=40&md5=17a474076bd21cfa7fb192909d5ae259>. Citado na página 36.

JAMROEN, C. et al. An intelligent irrigation scheduling system using low-cost wireless sensor network toward sustainable and precision agriculture. **IEEE Access**, v. 8, p. 172756–172769, 2020. Citado na página 36.

K., L. R.; VIJAYARAGHAVAN, V. Iot and cloud hinged smart irrigation system for urban and rural farmers employing mqtt protocol. In: **2020 5th International Conference on Devices, Circuits and Systems (ICDCS)**. [S.l.: s.n.], 2020. p. 71–75. Citado 2 vezes nas páginas 36 e 39.

KAMIENSKI, C. et al. Smart water management platform: Iot-based precision irrigation for agriculture †. **Sensors (Basel, Switzerland)**, v. 19, 2019. Citado na página 49.

KANUMALLI, S. S. et al. Automated irrigation management system using iot. In: **2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)**. [S.l.: s.n.], 2022. p. 476–482. Citado 2 vezes nas páginas 36 e 39.

KASERA, R. K.; ACHARJEE, T. A comprehensive iot edge based smart irrigation system for tomato cultivation. **Internet Things**, v. 28, p. 101356, 2024. Citado na página 50.

LEA, P. **IoT and Edge Computing for Architects: Implementing edge and IoT solutions from devices to cloud**. 2. ed. Birmingham, UK: Packt Publishing, 2020. ISBN 9781839214806. Citado 21 vezes nas páginas 19, 20, 21, 23, 24, 26, 27, 30, 31, 37, 42, 45, 61, 65, 66, 68, 69, 71, 72, 75 e 76.

MATILLA, D. M. et al. Low cost center pivot irrigation monitoring systems based on iot and lorawan technologies. In: **2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)**. [S.l.: s.n.], 2020. p. 262–267. Citado 2 vezes nas páginas 36 e 43.

MORCHID, A. et al. High-technology agriculture system to enhance food security: A concept of smart irrigation system using internet of things and cloud computing. **Journal of the Saudi Society of Agricultural Sciences**, 2024. Citado na página 50.

MUJOO, S. et al. Smart irrigation system using iot based control valve. In: **2021 Asian Conference on Innovation in Technology (ASIANCON)**. [S.l.: s.n.], 2021. p. 1–5. Citado na página 36.

MUNIR, M. et al. Intelligent and smart irrigation system using edge computing and iot. **Complex.**, v. 2021, p. 6691571:1–6691571:16, 2021. Citado na página 50.

NAJI, A. Z.; SALMAN, A. M. Water saving in agriculture through the use of smart irrigation system. In: **2021 4th International Conference on Data Storage and Data Engineering**. New York, NY, USA: Association for Computing Machinery, 2021. (DSDE '21), p. 153–160. ISBN 9781450389303. Disponível em: <https://doi.org/10.1145/3456146.3456170>. Citado na página 36.

NDUNAGU, J. et al. Development of a wireless sensor network and iot-based smart irrigation system. **Applied and Environmental Soil Science**, 2022. Citado na página 50.

P, J. et al. Electro-mechanical design of sensor-hub for indoor smart irrigation: System prototype. In: **2022 IEEE Sensors**. [S.l.: s.n.], 2022. p. 1–4. Citado na página 36.

PATEL, S. V. et al. Automated irrigation scheduling for different crop using iot. In: **2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)**. [S.l.: s.n.], 2022. p. 100–105. Citado 2 vezes nas páginas 36 e 39.

POYEN, F. B. et al. Prototype model design of automatic irrigation controller. **IEEE Transactions on Instrumentation and Measurement**, v. 70, p. 1–17, 2021. Citado na página 36.

PUIG, F.; DíAZ, J. A. R.; SORIANO, M. Development of a low-cost open-source platform for smart irrigation systems. **Agronomy**, 2022. Citado na página 50.

ROUTIS, G.; ROUSSAKI, I. Low power iot electronics in precision irrigation. **Smart Agricultural Technology**, v. 5, 2023. Cited by: 0; All Open Access, Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85170108988&doi=10.1016%2fj.atech.2023.100310&partnerID=40&md5=d72a8d6f9fa819489d65f2642c5fd1b0>. Citado na página 36.

SANJULA, W. et al. Automated water-gate controlling system for paddy fields. In: **2020 2nd International Conference on Advancements in Computing (ICAC)**. [S.l.: s.n.], 2020. v. 1, p. 61–66. Citado 2 vezes nas páginas 36 e 39.

SANTOS, G. et al. Agro smart : Iot autonomous irrigation system. In: **2022 17th Iberian Conference on Information Systems and Technologies (CISTI)**. [S.l.: s.n.], 2022. p. 1–6. Citado na página 36.

SHAHAR, S. H. et al. Arduino based irrigation monitoring system using node microcontroller unit and blynk application. **Indonesian Journal of Electrical Engineering and Computer Science**, v. 31, n. 3, p. 1334 – 1341, 2023. Cited by: 0; All Open Access, Gold Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85168956203&doi=10.11591%2fijeecs.v31.i3.pp1334-1341&partnerID=40&md5=3e366f534af3d88c0956c1141e385976>. Citado 2 vezes nas páginas 36 e 39.

SOUZA, G. et al. A fuzzy irrigation control system. In: **2020 IEEE Global Humanitarian Technology Conference (GHTC)**. [S.l.: s.n.], 2020. p. 1–6. Citado na página 36.

STEEN, M. V.; TANENBAUM, A. S. **Distributed Systems**. 4. ed. [S.l.]: Pearson, 2023. Version 4.03, January 2025. ISBN 978-90-815406-4-3. Citado 10 vezes nas páginas 23, 24, 25, 26, 27, 30, 72, 74, 75 e 76.

THAHER, T.; ISHAQ, I. Cloud-based internet of things approach for smart irrigation system: Design and implementation. In: **2020 International Conference on Promising Electronic Technologies (ICPET)**. [S.l.: s.n.], 2020. p. 32–37. Citado na página 36.

V, V. et al. Implementation of iot in agriculture: A scientific approach for smart irrigation. In: **2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)**. [S.l.: s.n.], 2022. p. 1–6. Citado 2 vezes nas páginas 36 e 39.

ZAIER, R. et al. Design and implementation of smart irrigation system for groundwater use at farm scale. In: **2015 7th International Conference on Modelling, Identification and Control (ICMIC)**. [S.l.: s.n.], 2015. p. 1–6. Citado na página 36.

ZHANG, Y. et al. Research and development of an iot smart irrigation system for farmland based on lora and edge computing. **Agronomy**, 2025. Citado na página 50.