UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

João Pedro de Souza Jardim da Costa

Uma Arquitetura de *Edge-Fog-Cloud* para Conciliar Interoperabilidade e Redução de Dados Focada em *E-Health*

Juiz de Fora
2023

João Pedro de Souza Jardim da Costa

Uma Arquitetura de *Edge-Fog-Cloud* para Conciliar Interoperabilidade e Redução de Dados Focada em *E-Health*

Dissertação apresentada ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciências da Computação. Área de concentração: Engenharia de Software e Banco de Dados

Orientador: Prof. Dr. Mário Antônio Ribeiro Dantas

Coorientador: Prof. Dr. José Maria Nazar David

Juiz de Fora

2023

Ficha catalográfica elaborada com os dados fornecidos pelo autor

Costa, João Pedro de S. J..
Uma Arquitetura de *Edge-Fog-Cloud* para Conciliar Interoperabilidade e Redução de Dados Focada em *E-Health* / João Pedro de Souza Jardim da Costa. – 2023.
64 f. : il.

Orientador: Mário Antônio Ribeiro Dantas
Coorientador: José Maria Nazar David
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciências da Computação, 2023.

1. Arquiteturas *Edge-Fog-Cloud*. 2. *e-Health*. 3. Interoperabilidade. 4. Ontologia 5. Redução de Dados I. Dantas, Mario, orient. II Nazar, José Maria, coorient. III. Título.

João Pedro de Souza Jardim da Costa

Uma arquitetura de Borda-Névoa-Nuvem para Conciliar Interoperabilidade e Redução de Dados Focada em E-Health

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação. Área de concentração: Ciência da Computação.

Aprovada em 13 de dezembro de 2023.

BANCA EXAMINADORA

Prof. Dr. Mario Antonio Ribeiro Dantas - Orientador

Universidade Federal de Juiz de Fora

Prof. Dr. José Maria Nazar David - Coorientador

Universidade Federal de Juiz de Fora

Prof. Dr. Victor Ströele de Andrade Menezes

Universidade Federal de Juiz de Fora

Profª. Dra. Aletéia Patrícia Favacho de Araújo Von Paumgartten

Universidade de Brasília

Juiz de Fora, 11/12/2023.

Documento assinado eletronicamente por **Victor Stroele de Andrade Menezes**, **Professor(a)**, em 15/12/2023, às 10:21, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Jose Maria Nazar David**, **Professor(a)**, em 15/12/2023, às 22:19, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Aleteia Araujo**, **Usuário Externo**, em 18/12/2023, às 16:11, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **mario antonio ribeiro dantas**, **Usuário Externo**, em 21/12/2023, às 10:22, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **1619893** e o código CRC **8DA73A40**.

Aos meus pais, pelo apoio e sustento.
À minha irmã, professores e amigos.

AGRADECIMENTOS

Agradeço aos meus pais, André e Solange, por me apoiarem durante o percurso de minha vida e o amor incondicional. Agradeço à minha irmã, Maria Clara, pelo amor, carinho e apoio emocional. A existência da minha família é o alicerce que sustenta a minha vontade de me tornar uma pessoa melhor e bem sucedida.

Agradeço ao meu orientador, Mário Dantas, e ao co-orientador, José Maria Nazar, por compartilharem seu tempo e experiência comigo. Neste vasto oceano, chamado conhecimento, eu ainda estaria à deriva sem a rota que foi traçada por sua sabedoria.

Agradeço aos meus colegas e amigos pelo trabalho em conjunto, que elevou a nossa capacidade e qualidade de nossos projetos.

Este trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, CNPq e FAPEMIG.

RESUMO

Com a necessidade de diminuir custos e melhorar o atendimento, o setor da saúde passou a empregar o conceito chamado *e-Health*. *E-Health* se refere ao uso de Tecnologias da Informação, com auxílio da Internet, para apoiar os processos da saúde e prover serviços. Entre essas tecnologias, há a interoperabilidade, a redução de dados e as arquiteturas *Edge-Fog-Cloud*. A interoperabilidade proporciona às aplicações a capacidade de compartilhar dados entre si, aumentando o escopo de informação às quais elas têm acesso. Arquiteturas de *Edge-Fog-Cloud* disponibilizam diversos serviços e produtos que facilitam a implantação, o gerenciamento e o funcionamento de aplicações e estruturas. Apesar de essas tecnologias proverem diversos benefícios, seu uso não está isento de ter seus próprios desafios. Um desses desafios é o aumento nos custos de armazenamento e transmissão de dados. Técnicas de redução de dados são uma das soluções para diminuir esses custos. Elas utilizam diversos procedimentos para diminuir o tamanho dos dados gerados pelas aplicações. No entanto, o uso de interoperabilidade em conjunto com técnicas de redução de dados acarreta em problemas que dificultam a sua conciliação. Esta dissertação de mestrado abordou os desafios de empregar interoperabilidade em paralelo com redução de dados em *Edge-Fog-Cloud computing*, com foco em *e-Health*. Este projeto apresenta uma arquitetura de *Edge-Fog-Cloud* que oferece funcionalidades como serviço, com o objetivo de garantir a interoperabilidade e a redução de dados. A proposta utiliza ontologias para verificar a interoperabilidade sintática e reduzir dados. Para avaliar a proposta, serão usadas simulações de contexto. Nos resultados das simulações, a proposta conseguiu aumentar o fluxo de dados na rede e reduzir o tamanho dos dados, sem interferir no compartilhamento de dados entre as aplicações.

Palavras-chave: arquiteturas *Edge-Fog-Cloud*; *e-Health*; interoperabilidade; ontologia; redução de dados.

ABSTRACT

With the need to reduce costs and improve care, the healthcare sector started to employ the concept called e-Health. e-Health refers to the use of Information Technologies, with the help of the Internet, to support healthcare processes and provide services. These technologies include interoperability, data reduction and Edge-Fog-Cloud architectures. Interoperability gives applications the ability to share data, increasing the scope of information they have access to. Edge-Fog-Cloud architectures provide several services and products that facilitate the deployment, management and operation of applications and structures. Although these technologies provide several benefits, their use is not without its challenges. One of these challenges is the increase in data storage and transmission costs. Data reduction techniques are one of the solutions to reduce these costs. They use several procedures to reduce the size of data generated by applications. However, the use of interoperability in conjunction with data reduction techniques leads to problems that make their reconciliation difficult. This master's thesis addressed the challenges of employing interoperability in parallel with data reduction in Edge-Fog-Cloud computing, with a focus on e-Health. This project presents an Edge-Fog-Cloud architecture that offers functionality as a service, to ensure interoperability and data reduction. The proposal uses ontologies to verify syntactic interoperability and reduce data. To evaluate the proposal, context simulations will be used. In the simulation results, the proposal managed to increase data flow on the network and reduce data size, without interfering with data sharing between applications.

Keywords: Edge-Fog-Cloud architectures; e-Health; interoperability; ontology; data reduction.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CPIISS	Catálogo de Padrões de Interoperabilidade] de Informações de Sistemas de Saúde
EEG	eletroencefalogramas
IaaS	Infrastructure as a Service
IoT	Internet of Things
IoE	Internet of Everything
HL7	Health Level Seven International
OpenEHR	Open Eletronic Health Records
OWL	Web Ontology Language
PaaS	Platform as a Service
EHR	Electronic Health Record
SaaS	Software as a Service
SMS	Sistema de Monitoramento de Saúde
SUS	Sistema Único de Saúde
TCC	Trabalho de Conclusão de Curso
TI	Tecnologias de Informação
W3C	World Wide Web Consortium

SUMÁRIO

1 INTRODUÇÃO

Com o aumento dos custos dos serviços de saúde e escassez de profissionais, novos métodos para oferecer atendimento são estudados. Uma das soluções encontradas foi o uso de Tecnologias de Informação (TI) como suporte para serviços de saúde. Eysenbach et al. (2001) definem Serviços de Saúde Eletrônica, ou *e-Health*, como a intersecção entre informática médica, saúde pública e negócios. Esses serviços englobam atividades e informações de saúde que são entregues através ou aprimorados pela internet e por tecnologias relacionadas. Ademais, o setor da Saúde necessita lidar com o crescimento de demanda desses serviços.

AbuKhousa, Mohamed e Al-Jaroodi (2012) exemplificam a Computação na *Cloud* como uma das tecnologias que poderiam auxiliar *e-Health* a lidar com esse crescimento. O raciocínio dos autores parte do fato de que a Computação na *Cloud* fornece uma infraestrutura forte, e facilita o fornecimento de serviços através da Internet. Outra possibilidade, discutida por Scarpato et al. (2017), é o uso de dispositivos de *Internet of Things* (IoT). A IoT fornece uma enorme quantidade de informações para serem reunidas, armazenadas e analisadas. Dispositivos de IoT são capazes de coletar e compartilhar grandes quantidades de dados. Eles são capazes de se comunicar diretamente com outros dispositivos por meio do ambiente da *Cloud*. A interoperabilidade é uma das outras formas das aplicações de saúde acessarem grandes quantidades de dados.

Shull (2019) argumenta sobre a importância da interoperabilidade para *e-Health*. Diferente das tecnologias mencionadas acima, a interoperabilidade é uma qualidade que a aplicação de saúde pode possuir, não uma função a ser realizada. Aplicações que são interoperáveis podem compartilhar dados e informações entre si. Interoperabilidade aumenta o escopo dos dados com o qual a aplicação trabalha e, consequentemente, melhora a precisão dos resultados de seus processamentos.

Todavia, gerenciar e transportar grandes quantidades de dados são desafios para a arquitetura de Computação na *Cloud*. Isso é devido ao fato dessas atividades gerarem altos custos de manutenção. A utilização de técnicas de redução de dados é uma das formas que essas arquiteturas encontraram para lidar com esses desafios. Utilizando essas técnicas, é possível diminuir o tamanho dos dados a serem transportados e ou armazenados. O fato de modificarem os dados originais, impossibilita a interoperabilidade entre as aplicações, anulando os benefícios resultantes do compartilhamento de dados.

Estudos anteriores que discutiram sobre a interoperabilidade e o uso de técnicas de redução em aplicações de saúde, na pesquisa realizada, não conciliaram ambos esses temas. Enquanto Rubí e Gondim (2020) e Rinty, Prodhan e Rahman (2022) buscaram soluções para desafios relacionados à interoperabilidade, Kahdim e Manaa (2022) e Idrees e Idrees (2022) discutem sobre métodos de redução de dados para aplicações de saúde.

Este estudo propôs desenvolver o protótipo de uma arquitetura *Edge-Fog-Cloud* para aplicações de saúde. Assim sendo, foram utilizadas neste trabalho técnicas de redução de dados para diminuir a quantidade de dados que necessitam trafegar pela rede, sem que a capacidade dos aplicativos de compartilhar dados fosse afetada.

1.1 OBJETIVO GERAL

Este trabalho teve como objetivo geral desenvolver uma abordagem, apoiada em um arquitetura distribuída contemporânea, utilizando-se de técnicas de redução de dados, para diminuir os custos de armazenamento e transporte de dados pela rede, sem dificultar a interoperabilidade das aplicações de *e-Health*.

1.2 OBJETIVOS ESPECÍFICOS

Para satisfazer o objetivo geral definido, os seguintes objetivos específicos foram definidos:

(a) Desenvolver uma abordagem que consuma o objetivo geral especificado, qual seja, utilizar técnicas de redução de dados, para reduzir os custos com gerenciamento e transporte de dados pela rede, sem afetar a interoperabilidade das aplicações que utilizam a arquitetura *Edge-Fog-Cloud* na qual a abordagem for aplicada;

(b) Testar e analisar o tráfego de dados quando utilizada a redução de dados proposta. Comparar os resultados do tráfego de dados com auxílio de técnicas de redução com os resultados sem o auxílio. Este objetivo teve o intuito de confirmar se a abordagem era capaz de reduzir os dados, sem impedir o compartilhamento de dados.

1.3 METODOLOGIA

Para construir a base de conhecimento, necessária para o desenvolvimento deste documento, foram realizadas buscas nas bases de dados IEEE Xplore[1], *Google Scholar* e *Research Gate*, dentre outras; haja vista encontrar trabalhos relacionados com a área em questão. Também foram feitas pesquisas sobre provedores de estruturas como serviço, à procura de ambientes para testes. Do mesmo modo, foi mandatório pesquisar por bases de dados que fornecessem conjuntos de dados que atendessem aos requisitos dos testes.

Para realizar as buscas, foram utilizadas palavras-chave como: "*Edge Computing*", "*Fog Computing*", "*Cloud Computing*", "*Data Reduction*", "IoT", "*Survey*", "*Data Compression*", "*interoperability*", "*Ontology*", "*e-Health*" e combinações dessas palavras. Outros instrumentos utilizados foram o de buscar os documentos citados por aqueles que foram encontrados nas buscas, os documentos que os próprios citados citaram e os documentos que citam os documentos buscados.

[1] Base de dados do *Institute of Electrical and Electronics Engineers*

Com os resultados das buscas e os documentos encontrados através das citações, cerca de 22 documentos foram estudados e 5 (cinco) provedores de estruturas como serviço foram analisados. 4(quatro) trabalhos relacionados à proposta da dissertação foram abordados. Até onde foi pesquisado, não se encontrou uma base de dados que estivesse de acordo com o projeto, por tanto, optou-se por utilizar simuladores de contexto para prover os dados necessários para testar a proposta.

1.4 ESTRUTURA DO TRABALHO

Para os objetivos desta proposta com sucesso, esta dissertação foi separada em cinco capítulos: 1. Introdução, 2. Referencial Bibliográfico, 3. Solução Proposta, 4. Ambiente de Testes e Resultados e 5. Conclusões e Trabalhos Futuros. O Capítulo 2 apresenta os conceitos e trabalhos utilizados para compor a base de conhecimento, tal qual discute sobre os trabalhos relacionados. O Capítulo 3, entra em detalhes sobre a arquitetura proposta e o processo de seu desenvolvimento. O Capítulo 4 discute sobre o ambiente de testes escolhido, como ele foi construído, quais testes foram feitos e os resultados desses testes. O Capítulo 5 contém a conclusão do trabalho e aborda os desenvolvimentos futuros almejados para o projeto.

1.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo introduziu o trabalho com um resumo do documento completo, descreveu a estrutura, os objetivos do trabalho e as etapas realizadas para construir a pesquisa proposta.

O próximo capítulo, aprofunda sobre os conceitos necessários para entender a proposta deste trabalho e identifica documentos científicos que desenvolveram soluções relacionadas à nossa proposta.

2 REFERENCIAL BIBLIOGRÁFICO

Este capítulo apresenta conceitos de Computação na *Cloud*, Computação na *Fog*, Computação na *Edge*, técnicas de redução de dados, ontologias e trabalhos relacionados à proposta desta dissertação.

2.1 COMPUTAÇÃO NA CLOUD

Computação na *Cloud*, ou *Cloud Computing*, é um paradigma que possui o objetivo de promover que empresas ou pessoas possam contratar plataformas, softwares e infra estruturas computacionais sobre demanda. Um aparelho com navegador web e acesso à Internet é o mínimo necessário para acessar e gerenciar esses recursos contratados. O serviço de contratação de software, conhecido como *Software as a Service* (SaaS), possibilita que usuários utilizem aplicações sem que elas estejam instaladas em seu aparelho; alguns exemplos desse serviço são o Facebook, Youtube e Dropbox. *Platform as a Service* (PaaS) se refere à contratação de ambientes de desenvolvimento ou plataformas. Usuários podem contratar esse produto para desenvolver e executar seus softwares dentro dele. Nessa modalidade, o provedor disponibiliza uma composição pré definida de sistema operacional e aplicações, para permitir que o contratante gerencie suas aplicações. Uma das composições mais comuns é a LAMP (Linux, Apache, MySQL e PHP). O terceiro serviço, *Infrastructure as a Service* (IaaS), provê recursos computacionais como redes, sistemas operacionais e hardware sob-demanda. O cliente que deseja contratar essa modalidade pode utilizá-la para montar sua própria plataforma(SRIVASTAVA; KHAN, 2018).

Apesar de Alam (2020) citar *Cloud Computing* como um dos melhores paradigmas computacionais da atualidade, ele também apresenta desafios para os seus usuários. Srivastava e Khan (2018) listam desafios críticos para *Cloud*, quando lida com fontes de dados de grande volume. Sensores e dispositivos inteligentes são algumas dessas fontes. Os desafios listados são: segurança, armazenamento e performance computacional, confiabilidade, armazenamento de *Big Data* e dispositivos inteligentes auxiliados por usuário.

segurança: Dados são enviados à *Cloud* para armazenamento e processamento. A falta de informação relacionada ao estado físico dos dados é tanta, que os usuários nem ao menos sabem em qual máquina, ou até mesmo qual o local físico da máquina, em que seus dados estão;

armazenamento e performance computacional: Serviços de IoT exigem alta performance dos servidores computacionais, essa exigência pode ser difícil de atender em todas as situações pois esses dispositivos normalmente estão em movimento;

confiabilidade: Dispositivos de IoT também hospedam serviços críticos, como segurança pública, aviação e instrumentos cirúrgicos. A confiabilidade do serviço prestado pelo servidor de *Cloud* reflete diretamente nos serviços que a utilizam;

armazenamento de *big data*: O crescimento exponencial de dispositivos de IoT aumenta a quantidade de dados gerados. Prover um acesso rápido e seguro a esses dados é um desafio para Computação na *Cloud*;

dispositivos inteligentes auxiliados por usuários: Com a contribuição dos usuários para o funcionamento dos dispositivos, é necessário levar em consideração os fatores sociais do contexto dos usuários.

2.2 COMPUTAÇÃO NA FOG

Vista como uma solução promissora para o problema do grande volume de dados, produzidos por dispositivos de IoT. Computação na *Fog* é um paradigma computacional que traz os recursos computacionais para mais perto da *Edge*. Esse paradigma faz a ponte entre os dispositivos da *Edge* e os servidores de *Cloud*. A característica responsável pela maioria das diferenças entre Computação na *Fog* e Computação na *Cloud* é o tamanho dos recursos computacionais aplicados nesses paradigmas. Enquanto a Computação na *Cloud* faz uso de grandes centros de dados, a Computação na *Fog* utiliza pequenos servidores e aparelhos relacionados a gerenciamento de rede. O custo para estabelecer um servidor de *Fog* é menor que o custo para um servidor de *Cloud*. Sua instalação física pode ser muito mais próxima à borda da rede. Servidores de *Fog* são projetados para possuírem um poder de processamento menor. Devido estarem mais perto da borda, sua latência é muito menor quando comparada a servidores de *Cloud* tradicionais, possibilitando que aplicações sensíveis à latência possam ser atendidas enquanto a Qualidade de Serviço (QdS) é garantida. Com diferenças e comprometimentos claros, a Computação na *Fog* não foi idealizada para substituir a Computação na *Cloud*, mas complementá-la(YOUSEFPOUR et al., 2019).

Assim como Computação na *Cloud*, Computação na *Fog* também possui seus desafios. Esse fato é resultante da *Fog* funcionar em um ambiente distribuído. Os desafios da *Fog* são relacionados a rede, segurança, dispositivos e integração entre *Fog* e IoT. Assim, os principais desafios deste ambiente são: estrutura descentralizada, recursos de rede, heterogeneidade de dispositivos, computação em diferentes níveis, recursos computacionais distribuídos, uso de recursos, falha de dispositivos de *Fog*, tratamento de solicitação de provisão, complexidade, segurança, computação orientada a serviços, gerenciamento de recursos e integração entre nós de *Fog* e *Cloud*(SABIREEN; NEELANARAYANAN, 2021).

estrutura descentralizada: A descentralização resulta em redundância na estrutura. Com a repetição de código entre os dispositivos, é necessário focar na redução

de redundância na arquitetura;

recursos de rede: Recursos de rede são distribuídos de forma aleatória entre os dispositivos na arquitetura de *Fog*. Soluções para gerenciar a distribuição dos recursos de rede diminuiria a complexidade de conectividade;

heterogeneidade de dispositivos: Como a *Fog* lida com diversos tipos e marcas de dispositivos de ponta, as aplicações que utilizam a *Fog* necessitam considerar esse aspecto;

computação em diferentes níveis: A *Fog* é uma ponte entre a *Edge* e a *Cloud*, em outras palavras, ela lida com diferentes níveis computacionais. De forma simplificada, o nível abaixo da *Fog* é a *Edge*, o nível acima é a *Cloud*. Esse desafio consiste em saber quais processamentos devem ser realizados por ela e quais devem ser passados para a *Cloud*;

recursos computacionais distribuídos: Portabilidade é uma das restrições à *Fog*. Nós na *Edge* podem ser portáteis. A *Fog* deve ser acessível de qualquer local e deve fornecer recursos computacionais para qualquer lugar. Ela deve ser capaz de realizar essas tarefas sem que haja um impacto perceptível na qualidade do serviço.

uso de recursos: O uso de recursos na *Fog* é extremamente volátil e heterogêneo, por consequência da diversidade de natureza dos dispositivos dela. Todos esses dispositivos são passíveis de executar a aplicação por conta própria.

falha de dispositivos da névoa: Erro de usuário, falha de hardware e software, fonte de energia e conectividade são algumas das causas dos dispositivos da *Fog* apresentarem falhas. Cabe ao provedor da *Fog* e desenvolvedores de aplicativos responderem a essas falhas, sem que a qualidade do serviço seja impactada;

tratamento de solicitação de provisão: *Fog* lida com milhões de dispositivos de IoT, que em certos momentos são responsivos e em outros são não-responsivos. O nível de obtenção e natureza desses serviços são muito diferentes na *Fog*;

complexidade: A heterogeneidade de sensores e dispositivos de IoT, disponíveis para a *Fog*, gera uma diversidade de estruturas de hardware, configurações de software e demandas pessoais. Ela causa um aumento na complexidade da escolha de mecanismos ideais para cada serviço a ser atendido. Ademais, certas aplicações exigem protocolos e dispositivos específicos para seu funcionamento;

segurança: A diversidade dos dispositivos utilizados pela *Fog* resulta na diversidade de vulnerabilidades com a qual a *Fog* necessita lidar. O ambiente no qual esses dispositivos estão dispostos não é altamente seguro, consequentemente, é fácil realizar um ataque digital ou físico;

computação orientada a serviços: A *Fog* necessita lidar com múltiplos pequenos serviços que estão espalhados pela *Cloud* e *Edge*. Realizar pequenos serviços por meio de nós da *Fog* tem seus próprios problemas. Organizar adequadamente a estrutura da *Fog* para executar os serviços é a grande questão na área de Computação na *Fog*;

gerenciamento de recursos: O gerenciamento de recursos da *Fog* necessita ser ajustável e adaptável para lidar com questões como falhas de curto tempo e falta de recursos. Recursos na *Fog* são virtualizados para lidar com eventuais falhas de nós, mas a própria virtualização de recursos possui seus próprios desafios;

integração entre nós de névoa e nuvem: A administração dos dispositivos heterogêneos na *Edge* deve ser efetuada em conjunto com o manuseio da arquitetura em *Cloud*. Isso é consequência da necessidade de realizar processamento e armazenamento em um ambiente distribuído. Desta forma, há a necessidade de alocar dinamicamente a integração de ponta a ponta dos dispositivos de *Fog* e os servidores em *Cloud*.

2.3 COMPUTAÇÃO NA EDGE

Enquanto as arquiteturas computacionais discutidas nas Seções 2.1 e 2.2 se importavam com IoT, a Computação na *Edge* surgiu em resposta à *Internet of Everything* (IoE). A insuficiência da Computação na *Cloud* no suporte às necessidades atuais foi o outro estopim para o surgimento da Computação na *Edge*. Sua ênfase é no processamento na borda da rede. Seu objetivo é atender às necessidades críticas do setor de TI, negócios em tempo real, otimização de dados, inteligência em aplicativos, em segurança e em privacidade, e atender aos requisitos de latência baixa e largura de banda alta na rede(CAO et al., 2020).

Tal como a Computação na *Fog*, que complementa a Computação na *Cloud*, a Computação na *Edge* também veio para preencher as lacunas que foram deixadas por esses outros paradigmas, ao invés de tentar substituí-los. Por aproximar os recursos computacionais o máximo possível ao usuário, Computação na *Edge* consegue realizar processamento e armazenamento na *Edge* sem a necessidade de enviar os dados para um servidor de *Cloud*. Essa característica da Computação na *Edge* trás vantagens na velocidade de resposta, diminui a exposição dos dados a possíveis perdas, possibilidades de vazamentos e diminui os gastos de energia, financeiros e de largura de banda. Tais gastos seriam muito maiores se apenas a Computação na *Cloud* tradicional estivesse disponível(CAO et al., 2020).

Proximidade aos usuários finais, suporte à mobilidade e densidade de implantação geográfica de servidores são as características que distinguem a Computação na *Edge* de outras arquiteturas. Essas características possibilitam que a Computação na *Edge* forneça serviços com latência muito baixa e largura de banda alta. Esse contexto e características,

distintos dos das outras arquiteturas, introduzem a Computação na *Edge* a uma outra gama de desafios, como: protocolo padrão, implantação efetiva, usuários móveis e transparência, heterogeneidade e escalabilidade, disponibilidade e segurança, interoperabilidade entre *Fog* e *Cloud*, gerenciamento de dados, gerenciamento de recursos distribuídos, confiabilidade e portabilidade e integração à rede e portabilidade de aplicações(AL-ANSI et al., 2021).

protocolo padrão: Computação na *Edge* é uma tecnologia moderna que necessita da colaboração entre a indústria e pesquisadores para sua padronização, através de uma plataforma mutuamente aceita;

implantação efetiva: A implantação efetiva de um servidor na *Edge* é o que possibilita a baixa latência e alta largura de banda. Otimizar o uso desse espectro é um desafio quando há dependência de componentes complexos do sistema;

usuários móveis e transparência: Prover um serviço ininterrupto e com migração transparente, para usuários móveis em um ambiente heterogêneo, é um desafio para Computação na *Edge*;

heterogeneidade e escalabilidade: Computação na *Edge* lida com vários dispositivos que utilizam diferentes tipos de tecnologias de acesso, como 5g, 4g e Wi-Fi. A qualidade de performance deve ser garantida para todos os dispositivos. Isso também envolve prover escalabilidade para diferentes plataformas com variados números de usuários;

disponibilidade e segurança: A disponibilidade de recursos frequentemente depende de aspectos físicos da estrutura e, da mesma forma, medidas no meio físico também devem ser tomadas para garantir a segurança dos dados;

interoperabilidade entre névoa e nuvem: É necessário considerar as diferenças em comunicação entre os componentes de neblina e os serviços de *Cloud* e as conexões entre componentes de neblina e componentes de *Edge*, ou as conexões entre os próprios componentes de *Fog*;

gerenciamento de dados: É necessário que a Computação na *Edge* possua diversos recursos de gerenciamento de dados, sendo eles: normalização dos dados, filtragem e consulta de dados, integração com análises da *Edge* e compilação de dados abstratos e ou metadados;

gerenciamento de recursos distribuídos: Gerenciamento de recursos distribuídos é importante para Computação na *Edge*. A quantidade limitada de recursos, aumento do número de aplicações e aumento massivo do tráfego de dados relacionados a dispositivos móveis são a causa. Apesar do método centralizado ser capaz de oferecer

um desempenho competitivo, ele sofre de baixa complexidade computacional e grandes despesas com relatórios;

confiabilidade e portabilidade: A mobilidade do usuário pode causar entregas frequentes. Isso resulta em problemas de inatividade dos serviços e afeta o desempenho geral da rede. Outra possibilidade é o usuário se movimentar para fora da cobertura do servidor, impossibilitando que receba o resultado de suas operações. Fornecer serviços confiáveis em ambientes móveis é realmente desafiador devido a variações de tempo e mobilidade do usuário;

integração à rede e portabilidade de aplicações: A capacidade de ser implantado em várias localidades é uma das características dos servidores destinados a Computação na *Edge*. Sendo assim, outro desafio da Computação na *Edge* é a integração do servidor de *Edge* à estrutura de rede já existente na região. Essa operação deve ser realizada sem que os padrões básicos de rede e de hardware periférico sejam afetados.

2.4 COMPARATIVO ENTRE EDGE, FOG E CLOUD

Seguindo os conceitos apresentados nas Seções 2.1, 2.2 e 2.3, conclui-se que *Edge*, *Fog* e *Cloud* possuem objetivos e características diferentes. A tabela 2.1 dispõe um comparativo dessas características.

Tabela 2.1 – Comparativo entre as características de estruturas da *Edge*, *Fog* e *Cloud*.

Características	***Edge***	***Fog***	***Cloud***
Armazenamento	Baixo	Alto	Muito alto
Estrutura	Descentralizada	Descentralizada	Centralizada
Consumo de Energia	Baixo	Alto	Muito alto
Latência	Muito baixa	Baixa	Alta
Segurança	Alta	Baixa	Baixa
Processamento	Baixo	Alto	Muito alta

A partir da Tabela 2.1, é possível distinguir o contexto nos quais cada estrutura se sobressai. Aplicações que necessitam extrair informações de um grande conjunto de dados exigem um alto poder de armazenamento e processamento. Levando em consideração essas necessidades e as informações da Tabela 2.1, a melhor escolha para essa aplicação seria contratar um servidor de *Cloud*. Aplicações que lidam com uma grande quantidade de usuários, espalhados geograficamente, necessita que sua aplicação tenha baixa latência e poder de processamento alto. Essas exigências são uma reflexão da necessidade da aplicação suprir a demanda de seu serviço. Uma estrutura composta de servidores de *Fog* seria capaz de atender a essas exigências. Por fim, estruturas compostas por servidores de

Edge podem atender a aplicações críticas, que necessitam de uma resposta rápida às suas solicitações que somente seria possível com uma latência muito baixa.

Não obstante, ainda existem as estruturas híbridas, que são compostas de combinações das estruturas de *Edge*, *Fog* e *Cloud*. A proposta deste projeto utilizou de duas estruturas híbridas, uma composta por servidores de *Fog* e *Cloud* e outra composta por servidores de *Edge* , *Fog* e *Cloud*.

2.5 TÉCNICAS DE REDUÇÃO DE DADOS

Como apontado nas Seções 2.1, 2.2 e 2.3, Computação na *Cloud*, *Fog* e *Edge* necessitam gerenciar grandes quantidades de dados. Uma das formas de lidar com esse grande volume de dados é o uso de técnicas de redução de dados. Redução de dados é uma das quatro técnicas de pré-processamento de dados. O objetivo desse tipo de pré-processamento é reduzir o tamanho de um conjunto de dados. A capacidade da Computação na *Edge* utilizar técnicas de redução de dados de forma eficiente é um dos motivos de sua emergência. Isso acontece pois a latência da largura de banda da rede pode ser reduzida em um *gateway*, diminuindo o tráfego de dados e prevenindo gargalos na rede. As técnicas que pesquisadores mais utilizam como método para reduzir dados na *Edge* são: processamento de sinal, redução de dimensão, clusterização de dados, imputação de dados, padrão de dados, predição de dados, agregação de dados, compressão de dados, filtragem de dados, encaminhamento seletivo, amostragem de dados, redução por partes, correlação de dados e fusão de dados. Filtragem de dados foi a técnica mais utilizada, seguida pela compressão de dados, enquanto processamento de sinal, amostragem de dados, redução por partes e amputação de dados ficaram empatadas como as menos populares(PIOLI et al., 2022).

2.6 INTEROPERABILIDADE

Interoperabilidade é a habilidade de diferentes tecnologias de comunicação, de informação e aplicações de software se comunicarem. O intuito é a troca de dados de forma eficiente e precisa e utilizar as informações que foram recebidas. Em um cenário com duas aplicações, desenvolvidas e mantidas por organizações diferentes, ambas só seriam interoperáveis se fossem capazes de trocar dados entre si e de interpretar as informações contidas nesses dados. Não apenas as aplicações, para que a interoperabilidade seja alcançada, os usuários também precisam entender as informações que foram compartilhadas. Geralmente, a interoperabilidade pode ser classificada em sete diferentes níveis(IROJU et al., 2013).

O primeiro nível é classificado como o "sem interoperabilidade". Ele é caracterizado por sistemas independentes sem nenhuma capacidade de compartilhar dados. A interoperabilidade técnica é o segundo nível. Nela é feito o uso de protocolos de comunicação para a troca de mensagem entre sistemas. A interoperabilidade sintática exige que dois ou mais sistemas consigam utilizar protocolos de interoperabilidade para comutar dados e servi-

ços. Chama-se de interoperabilidade semântica quando dois ou mais sistemas conseguem automaticamente interpretar, de forma significativa e precisa, as informações trocadas entre eles. Na interoperabilidade pragmática, os sistemas são conscientes dos métodos e procedimentos que cada um emprega. Para alcançar a interoperabilidade dinâmica, os sistemas necessitam compreender as mudanças de estado que ocorrem nas suposições e restrições que fazem ao longo do tempo, além de se aproveitarem dessas mudanças. No último nível de interoperabilidade, chamado interoperabilidade conceitual, é necessário alinhar as suposições e restrições da abstração significativa da realidade(IROJU et al., 2013).

No Brasil, a Portaria 2.073, de 31 de agosto de 2011[1] regulamenta os padrões utilizados na interoperabilidade entre sistemas de saúde governamentais, privados e do setor de saúde suplementar. Os sistemas governamentais são Sistema Único de Saúde (SUS), sistemas de nível Municipal, Distrital, Estadual e Federal. O artigo 4º, da portaria em questão, apresenta o Catálogo de Padrões de Interoperabilidade de Informações de Sistemas de Saúde (CPIISS)[2]. Esse catalogo dispõe os padrões a serem utilizados no âmbito do território brasileiro para fins de interoperabilidade entre os sistemas de informações de saúde.

2.7 ONTOLOGIAS

Em Tecnologias de Informação, ontologias descrevem os conceitos de um domínio e as relações que existem entre esses conceitos. A *Web Ontology Language*(OWL) é uma das linguagens ontológicas existentes, que também é uma das mais utilizadas. Desenvolvida pela *World Wide Web Consortium*(W3C)[3], a OWL possibilita descrever conceitos, mas também adiciona novas funcionalidades, que linguagens anteriores não possuíam(HORRIDGE et al., 2009).

Welty, McGuinness e Smith (2004) descrevem, em seu guia, recomendado pela W3C, as funcionalidades e as características que a OWL possui. O guia define classes, propriedades, instâncias de classes e relacionamentos entre essas instâncias como os elementos básicos da linguagem OWL. Classes simbolizam a categoria, conceito ou tipo ao qual um indivíduo pertence. As instâncias representam os indivíduos. Cada instância descreve um membro de uma certa classe. Propriedades são atributos que possibilitam afirmar fatos gerais sobre membros de uma classe e fatos específicos sobre um indivíduo. Com o uso de taxonomia e propriedades, é possível relacionar as classes e indivíduos.

[1] Disponível em: https://bvsms.saude.gov.br/bvs/saudelegis/gm/2011/prt2073_31_08_2011.html
[2] Disponível em: https://www.conass.org.br/biblioteca/wp-content/uploads/2011/02/NT-37-Padrões-de-Interoperabilidade-versão-2011.pdf
[3] Disponível em: http://www.w3.org/TR/owl-guide/

2.8 TRABALHOS RELACIONADOS

A partir das bases de dados citadas na seção 1.3, foram feitas buscas sobre trabalhos relacionados à proposta deste projeto. As palavras-chave utilizadas foram "*interoperability*","*cloud computing*","*fog computing*","*edge computing*", "*data reduction*", "*health*"e combinações das mesmas. Além dos trabalhos que foram encontrados nas buscas preliminares, os trabalhos que eles citam também foram acessados. Deste conjunto de trabalhos, quatro trabalhos foram selecionados para serem discutidos como trabalhos relacionados.

Um dos nossos objetivos principais foi a redução de dados sem perdas. Idrees e Idrees(2022) propõem um método de compressão de dados. Ele foi utilizado para diminuir a quantidade de dados enviados por dispositivos de IoT de eletroencefalogramas (EEG). Em seu projeto, os dados a serem reduzidos são produzidos por um sistema de monitoramento de saúde (SMS). Esse sistema utiliza aparelhos físicos acoplados ao corpo do paciente para coletar dados e enviá-los para processamento através da rede. As três partes essenciais do sistema, utilizado por Idrees e Idrees (2022), são nodos de sensores médicos, *gateway* e servidores de *back end*. Os nodos de sensores médicos consistem de sensores vestíveis ou implantáveis no corpo humano. Os sensores são utilizados para coletar diversos dados relacionados à condição física do paciente. O *gateway* conecta os sensores médicos aos servidores da *Cloud*. Ele é o ponto crítico da estrutura. Os dados gerados pelos sensores não seriam acessíveis se o *gateway* parasse de funcionar corretamente. Os servidores de *back end* incluíam um servidor de *Cloud* e um servidor de saúde. Os servidores de *Cloud* foram responsáveis por reunir os dados, processá-los e transmiti-los para os servidores de saúde. Por sua vez, os servidores de saúde analisavam os dados médicos e monitoravam a condição física dos pacientes. A técnica de redução que utilizaram consistia de duas etapas: clusterização e compressão. Para clusterização, adotaram o algoritmo hierárquico de clusterização. Para a compressão, escolheram o algoritmo de codificação de Huffman. Periodicamente, os dados dos sensores eram coletados no *gateway* inteligente de *Fog*. Esses dados eram agrupados em diversos *clusters*, de acordo com as similaridades encontradas pelo algoritmo hierárquico de clusterização. O algoritmo de codificação de Huffman era aplicado em cada *cluster* para comprimir os dados enviados. O trabalho de Idrees e Idrees (2022) teve como contribuição um novo procedimento de compressão sem perdas para reduzir dados produzidos por sensores EEGs. Eles empregaram a compressão proposta no *gateway* de *Fog* da arquitetura utilizada. Por fim, realizaram extensivos experimentos, utilizando a linguagem de programação Python e dados reais coletados de pacientes através de sensores EEGs.

Kahdim e Manaa (2022) projetaram um método diferente de compressão de dados para aplicações de assistência médica. O intuito foi diminuir a quantidade de dados enviados do nível de sensores de IoT para o nível de *Fog*. Os componentes físicos utilizados para os testes foram um sensor de temperatura, um sensor de batimentos cardíacos, um

sensor de oxigênio, um Raspberry Pi e um computador pessoal, agindo como um servidor de *Fog*. Raspberry Pi é um poderoso, barato e adaptável mini-computador. Pelas suas características mencionadas anteriormente, bem como pelo fato de oferecer suporte para uma ampla variedade de portas de entrada e saída, ele foi escolhido como o dispositivo principal da solução de Kahdim e Manaa (2022). É no Raspberry Pi em que os dados são agrupados e compressos, e os sensores são monitorados. Tal como Idrees e Idrees (2022), a linguagem de programação escolhida por Kahdim e Manaa (2022) foi Python. Eles explicam que a motivação para escolher essa linguagem é o fato de ela ser adaptável, poderosa e útil para diversos tipos de tarefas. O fluxo de dados da proposta se inicia com a leitura dos dados gerados por sensores IoT. Após esses dados serem agregados no Raspeberry Pi, é aplicado o algoritmo de compressão Zstandard. Esses dados comprimidos são enviados para um servidor de *Fog*. No nível da *Fog*, os dados são descomprimidos, analisados e, por fim, armazenados. Como conclusão, Kahdim e Manaa (2022) afirmam que, com o uso de compressão Zstandard, o sistema proposto pode reduzir significativamente a quantidade de dados médicos enviados pelo Raspberry Pi para a *Fog*.

A interoperabilidade entre aplicações é outro aspecto crítico neste projeto. Pensando na comunicação entre máquinas e aplicações legadas da área de saúde, Rubí e Gondim (2020) desenvolveram uma plataforma que as possibilitaram interoperar. Considerada uma plataforma de Internet of Medical Things, ela fez uso da *Open Eletronic Health Records*(OpenEHR) e semânticas de rede de sensores semânticos. OpenEHR é um padrão para modelagem de registros eletrônicos de saúde. Com o uso de ontologias OWL baseadas no padrão OpenEHR, é possível padronizar a semântica dos dados compartilhados na plataforma. A solução consiste em um servidor de armazenamento e um controlador de armazenamento. Eles realizam a mediação das comunicações entre as aplicações legadas e as novas aplicações. As aplicações legadas são ligadas diretamente ao servidor de armazenamento, esse servidor possui um controlador que faz a comunicação com o provedor de serviço da rede. O servidor de armazenamento traduz os dados produzidos pelas aplicações legadas para o padrão OpenEHR, com isso, esses dados podem ser utilizados por novas aplicações que se baseiam nesse padrão. O contrário também é realizado, os dados no padrão OpenEHR, enviado pelas aplicações novas, são traduzidos para a semântica das aplicações legadas. O controlador também é responsável pela atualização dos termos no servidor de armazenamento. O controlador atualiza o registro quando recebe uma mensagem de observação. Rubí e Gondim (2020) apresentaram como contribuição principal uma ontologia OpenEHR estendida, que alinha o domínio de aplicações padronizadas no modelo OpenEHR com aplicações legadas.

Como explicado anteriormente, a interoperabilidade viabiliza que aplicações compartilhem dados entre si, no entanto, é uma qualidade que enfrenta desafios para ser implementada e utilizada. Rinty, Prodhan e Rahman (2022) discutem sobre o fato de países em desenvolvimento terem dificuldade em aplicar interoperabilidade em *e-Health*.

Visando resolver esse problema, eles propuseram um *framework* aprimorado de *e-Health* para países em desenvolvimento. A proposta utiliza memória distribuída e mecanismos de manipulação de dados heterogêneos com baixo acoplamento para alcançar seu objetivo. O padrão de estrutura proposto pelo padrão *Health Level Seven International*(HL7) foi o escolhido para o *framework*. HL7 é um conjunto de normas, padrões e definições internacionais utilizados na transferência e comunicação entre praticantes de assistência médica. Ele foi escolhido por ser um dos modelos de design mais importantes entre os padrões de dados de saúde. A proposta consiste em dois módulos controlando o acesso e a transição de dados, o módulo *Health System Administrator* e o *Local Administrator*. O módulo chamado *Health System Administrator* comunica diretamente com o usuário final. Ele recebe e distribui as solicitações feitas pelos usuários finais entre as diferentes instâncias de *Local Administrator*. O *Health System Administrator* também controla o limite de buscas concorrentes para impedir que aconteçam sobrecargas. Cada servidor de Registro Eletrônico de Saúde, ou *Electronic Health Record*(EHR), possui sua própria instância de *Local Administrator*. O *Local Administrator* é responsável por analisar as solicitações e traduzi-las em declarações executáveis que o servidor de EHR compreende. Rinty, Prodhan e Rahman (2022) apontam esta configuração como o que possibilita o baixo acoplamento e a interoperabilidade entre dados médicos e interface de busca.

Percebe-se que, entre os trabalhos pesquisados, aqueles que apresentam os mesmos temas de nossa proposta não conciliam a interoperabilidade com as técnicas de redução de dados. Idrees e Idrees (2022) abordam a redução sem perdas, no entanto, eles não tratam sobre as implicações que tal abordagem teria na comunicação com outras aplicações. O mesmo é observado no trabalho de (KAHDIM; MANAA, 2022). Eles apresentaram um método de redução de dados para aplicações de saúde que também utiliza a compressão de dados. O que difere entre os dois trabalhos é o algoritmo de compressão e o fluxo dos dados. Quanto à interoperabilidade, ambas as propostas não discutem sobre a mesma. Em contrapartida, Rubí e Gondim (2020) e Rinty, Prodhan e Rahman (2022) focam na questão da interoperabilidade. Rubí e Gondim (2020) discutiram sobre e expuseram uma solução para a falta de interoperabilidade das aplicações legadas de saúde. Enquanto a solução de Rinty, Prodhan e Rahman (2022), aborda as dificuldades de interoperabilidade das aplicações de saúde em países em desenvolvimento. Contudo, ambas as propostas não levam em consideração casos em que as aplicações utilizam técnicas de redução de dados.

2.9 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foram discutidos os conceitos necessários para o entendimento da proposta deste documento e trabalhos que apresentam abordagens relacionadas à nossa proposta. Os conceitos discutidos foram Computação na *Cloud*, Computação na *Fog*, Computação na *Edge*, técnicas de redução de dados, interoperabilidade e ontologias. Os trabalhos

relacionados discutidos foram dois relacionados à interoperabilidade, o primeiro por Rubí e Gondim (2020) e o seguinte por Rinty, Prodhan e Rahman (2022), e dois trabalhos que discursam sobre o uso de técnicas de redução de dados, sendo um feito por Kahdim e Manaa (2022) e outro feito por Idrees e Idrees (2022).

O Capítulo 3 explica sobre o modelo de arquitetura que propomos como solução para o problema, como ocorre a interoperabilidade, qual a ontologia utilizada pela proposta e a técnica de redução de dados.

3 SOLUÇÃO PROPOSTA

Levando em consideração as informações do Capítulo 2, percebe-se que a interoperabilidade e técnicas de redução de dados são benéficas ao funcionamento e eficiência das aplicações de saúde. A interoperabilidade proporciona às aplicações a possibilidade de acessarem um escopo de dados muito maior, quando comparado ao escopo de dados que conseguiriam captar sozinhas. Entretanto, como exemplificado na Figura 3.1, lidar com grandes quantidades de dados aumenta consideravelmente os custos de transporte e armazenamento.

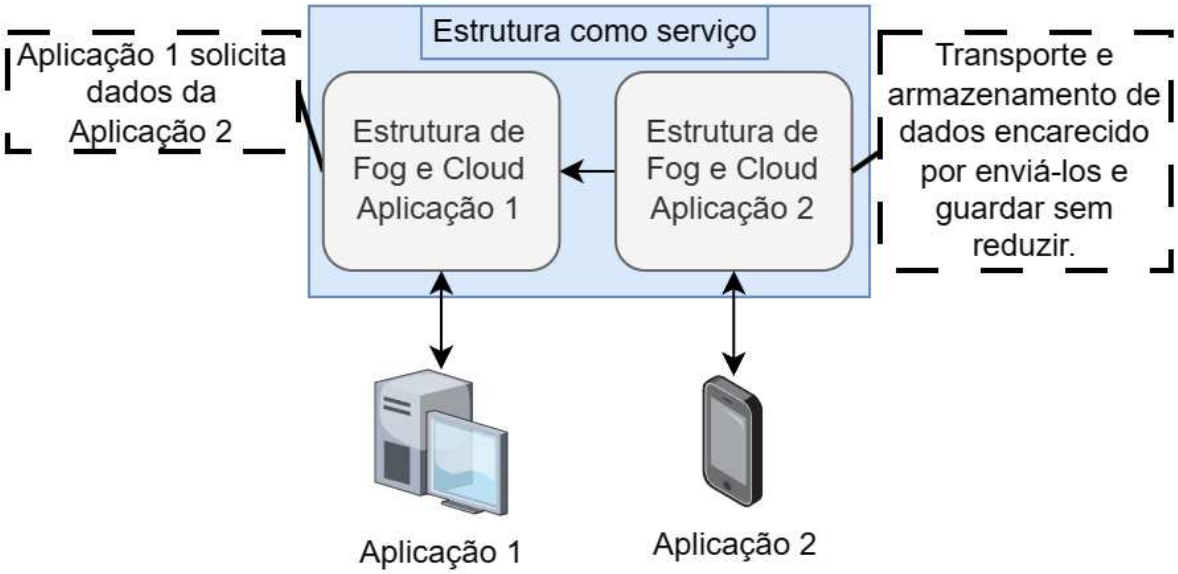

Figura 3.1 - Exemplo de contexto de interoperabilidade de duas aplicações que contrataram uma estrutura *Fog-Cloud* como serviço.

Na Figura 3.1, a Aplicação 1 e a Aplicação 2 contrataram uma estrutura como serviço. A estrutura é composta por servidores de *Fog* e um servidor de *Cloud*. A Aplicação 1 deseja que a Aplicação 2 compartilhe os dados que ela capta. Para atender ao pedido da Aplicação 1, a Aplicação 2 guarda e disponibiliza na *Cloud* e na *Fog* todos os dados que ela capta. Todavia, quanto maior for o volume de dados captados, maior será o custo de armazenamento e transporte de dados. Técnicas de redução de dados diminuem significativamente os custos com tráfego e armazenamento de dados, mas cria novos desafios. O uso de técnicas de redução de dados pode gerar conflitos com a capacidade de interoperar das aplicações. As Figuras 3.2 e 3.3 demonstram que diferentes tipos de técnicas de redução resultam em diferentes tipos de conflitos.

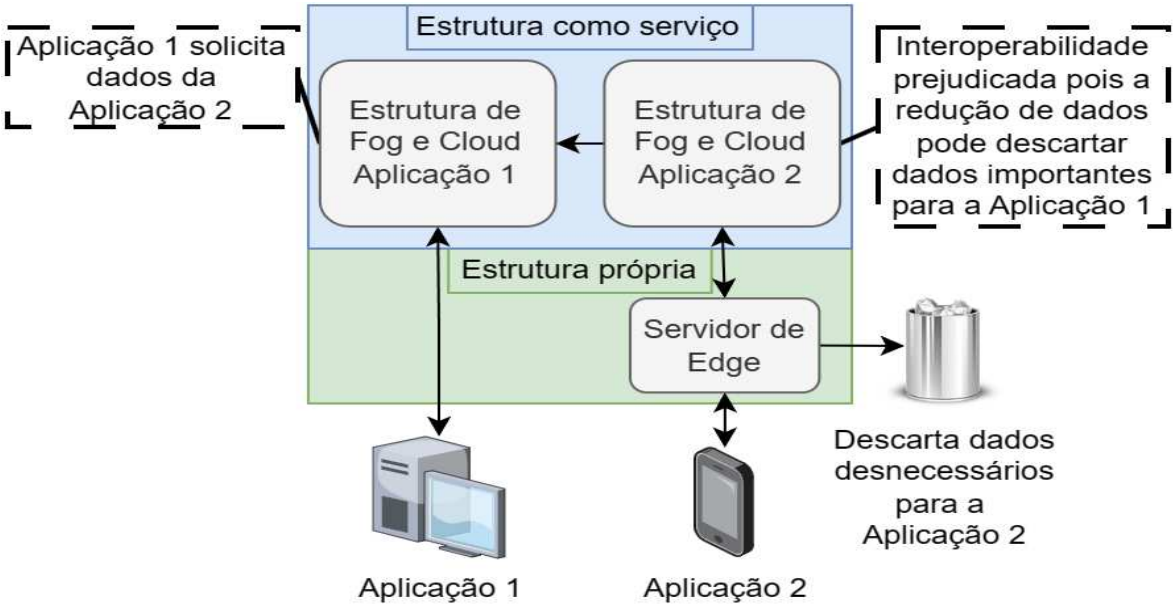

Figura 3.2 - Diagrama exemplificando como o descarte de dados pode prejudicar a interoperabilidade entre aplicações.

No exemplo representado pela figura 3.2, dados importantes para a Aplicação 1 não necessariamente serão importantes para a Aplicação 2. Quando a Aplicação 1 descarta esses dados, a Aplicação 2 perde o motivo para compartilhar dados com a Aplicação 1. Técnicas que modificam os dados são a outra opção. Assim como técnicas de descarte, elas também impõem obstáculos à interoperabilidade.

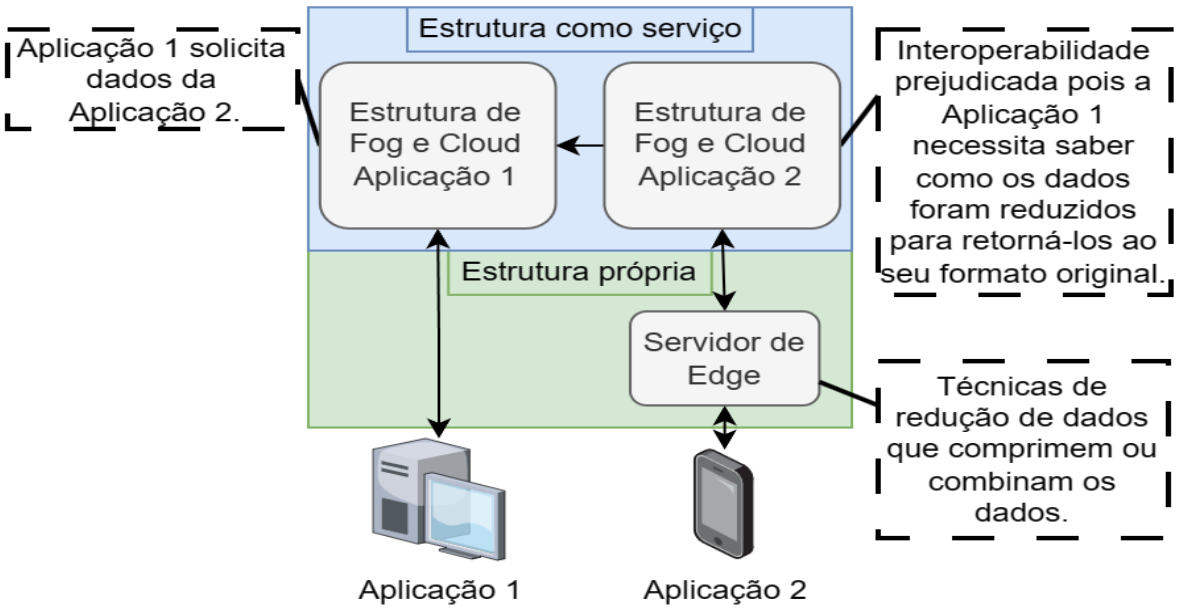

Figura 3.3 - Exemplo de como a redução de dados, ao modificá-los, pode impedir a interoperabilidade.

Assim como ilustrado na Figura 3.3, a Aplicação 1 poderia implementar o método necessário para restaurar os dados ao seu estado original. Em um contexto onde todas as aplicações utilizam essa técnica, a Aplicação 1 teria que implementar o método de restauração correto para cada nova aplicação com quem deseja compartilhar dados. No final, uma grande porção do código da Aplicação 1 seria inútil à sua funcionalidade principal. Focando em uma solução para esses conflitos, propomos uma arquitetura na qual a redução dos dados e interoperabilidade sintática são oferecidas como parte de um serviço. O diagrama da Figura 3.4 apresenta a abordagem em um modelo de alto nível.

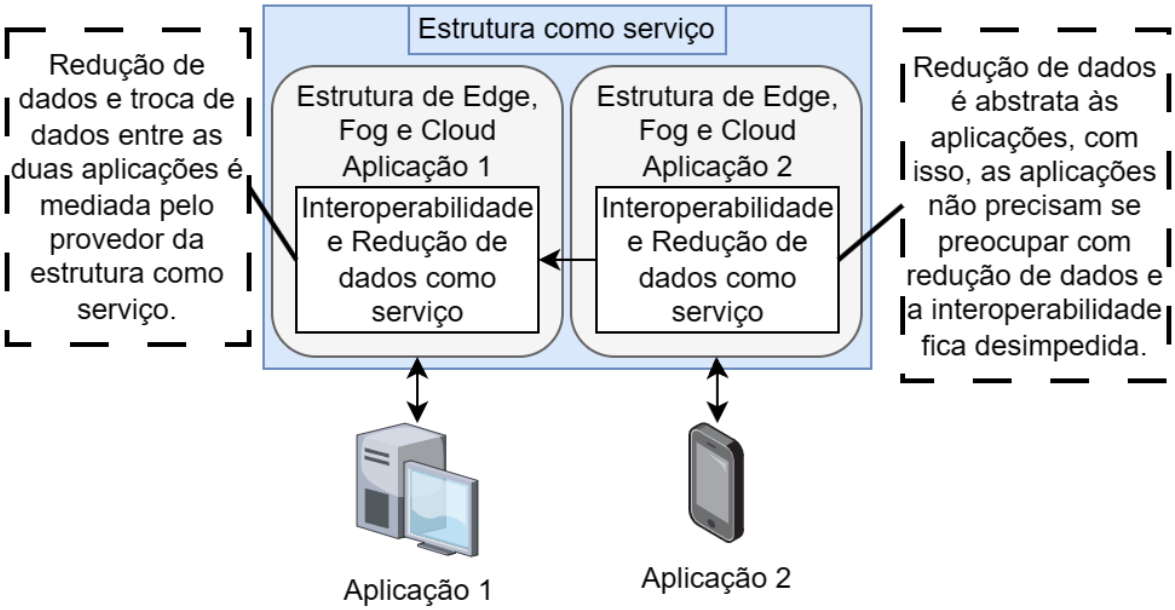

Figura 3.4 - Diagrama de alto nível da solução proposta.

A proposta define que essas responsabilidades sejam abstratas aos desenvolvedores das aplicações. Com isso, a responsabilidade pela implantação dessas funcionalidades caberia aos fornecedores do serviço de *Edge-Fog-Cloud*.

3.1 ARQUITETURA

A solução consiste em dois módulos implantados nos servidores de cada camada da arquitetura *Edge-Fog-Cloud*. A Figura 3.5 apresenta uma aplicação cliente-servidor utilizando os módulos, em uma estrutura com apenas a camada da *Cloud*.

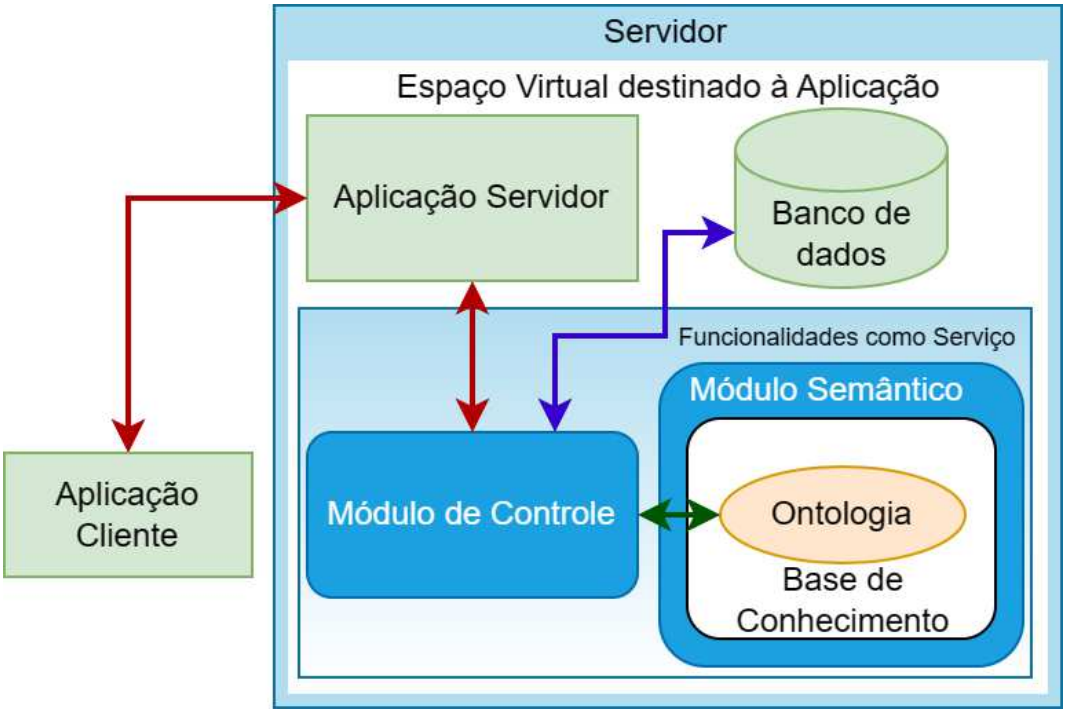

Figura 3.5 - Modelo de uma aplicação cliente-servidor, utilizando os módulos da proposta.

Neste modelo, a aplicação utiliza a solução para reduzir os dados, ocupando menos espaço no banco de dados. A solução consiste em dois módulos, Módulo Semântico e Módulo de Controle. O Módulo Semântico abriga uma ontologia. A ontologia é a estrutura utilizada para reduzir e expandir os dados. A ontologia foi escolhida a partir dos padrões regulamentados pelo governo federal brasileiro, dispostos pelo CPIISS[1]. O Módulo de Controle é responsável pela manutenção da ontologia, gerenciamento dos dados e fluxo dos dados. Ele atualiza a ontologia com novos termos, adiciona os dados originais à ontologia e extrai a versão reduzida desses dados.

A situação ótima de uso da proposta depende da existência de servidores de *Edge*. A Figura 3.6 apresenta o fluxo dos dados pela arquitetura e o momento em que os dados passam do seu formato original para reduzidos.

[1] Disponível em: https://www.conass.org.br/biblioteca/wp-content/uploads/2011/02/NT-37-Padrões-de-Interoperabilidade-versão-2011.pdf

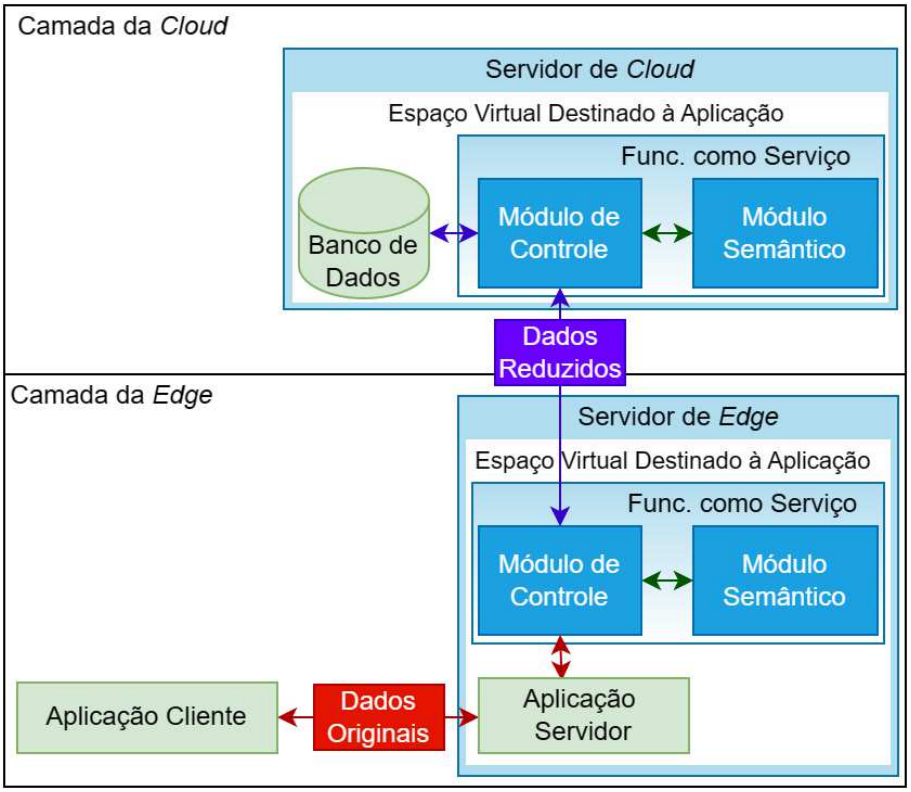

Figura 3.6 - Modelo do fluxo dos dados na arquitetura.

Existem duas possibilidades para que a proposta seja capaz de reduzir os custos do tráfego de dados até o servidor de *Cloud*. Na primeira, a aplicação envia os dados para um servidor de *Edge*. Na segunda, com a indisponibilidade de um servidor de *Edge*, os dados são enviados a um servidor de *Fog*. Após receber os dados, o servidor realiza a redução desses dados e os envia para o servidor da próxima camada. A Figura 3.6 apresenta um exemplo no qual a aplicação possui um banco de dados na *Cloud* e uma aplicação de processamento em um servidor de *Edge*. Com o uso da proposta, ela consegue reduzir os custos do tráfego de dados entre a *Edge* e a *Cloud* e os custos de armazenamento dos dados na *Cloud*. Acredita-se que, na circunstância em que não existem servidores de *Edge* disponíveis, a diminuição nos custos de transporte seria menor. Isso é resultado de os dados terem que trafegar por um percurso maior em seu formato original. Não obstante, o impacto nos custos de armazenamento continuaria o mesmo, afinal, o custo de armazenamento só leva em consideração o espaço que os dados ocupam.

Com a abordagem de redução de dados proposta, as aplicações não sabem que os dados foram reduzidos. Por causa da abstração, e o fato de não ser feito o descarte de dados, as aplicações estão habilitadas a compartilhar dados em um nível sintático. Essas características possibilitam que não haja empecilhos causados pela redução de dados. Se a responsabilidade de redução e expansão dos dados fossem das aplicações, o receptor dos dados necessitaria saber como foi feita a redução para ser capaz de expandir os dados.

Isso significa que ele deve desenvolver a abordagem de expansão dos dados correta para cada um com quem deseja compartilhar dados. Isso seria inviável nos casos em que o desenvolvedor deseja que sua aplicação compartilhe dados com uma grande quantidade de aplicações. A probabilidade é que cada aplicação utilize padrões e chaves de redução diferentes entre si. Portanto, visto que a redução é responsabilidade do provedor da arquitetura de *Edge-Fog-Cloud*, os desenvolvedores das aplicações não necessitam lidar com esse problema ao utilizarem a arquitetura proposta.

A centralização da responsabilidade da redução dos dados permite que as aplicações fiquem livres para compartilhar dados entre si. A Figura 3.7 apresenta uma situação na qual duas aplicações compartilham dados dentro de uma arquitetura *Edge-Fog-Cloud*.

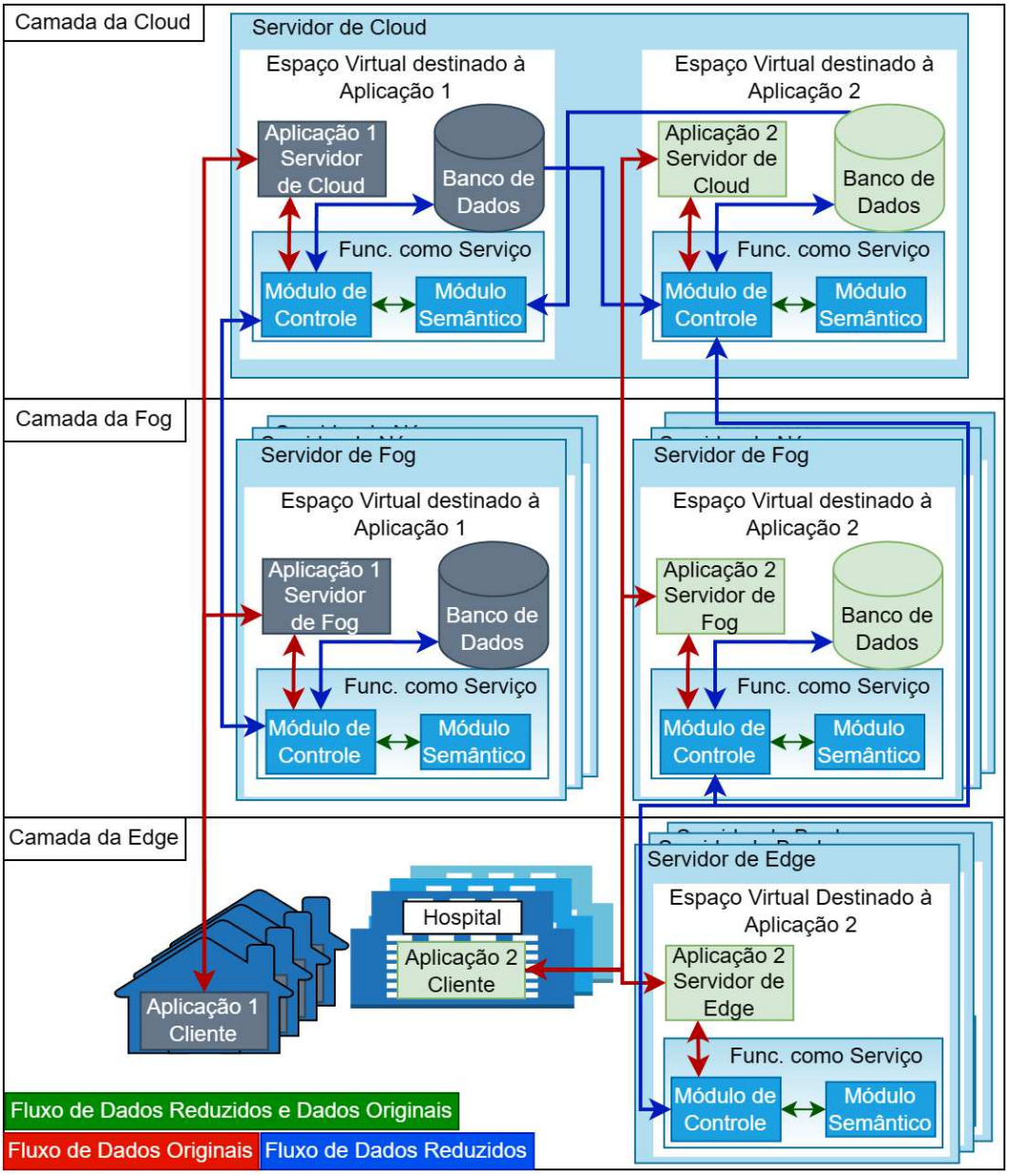

Figura 3.7 - Fluxo de dados onde duas aplicações utilizam a arquitetura para compartilhar dados.

As aplicações ilustradas na Figura 3.7 utilizam o modelo cliente-servidor e são relacionadas à área de saúde. A Aplicação 1 foi projetada para atender pacientes em suas casas. Ela coleta dados sobre o ambiente ao redor do paciente e sobre seu estado físico, além de servir como um meio de telemedicina. A Aplicação 2 dá suporte a profissionais da área de saúde no atendimento de seus pacientes. Através dela é feito o controle dos atendimentos, o controle dos dados dos pacientes e são geradas estatísticas para tomada de decisão. A Aplicação 1 não contratou servidores de *Edge*, enquanto a Aplicação 2 faz uso de todas as camadas. Os servidores de *Edge* servem como o ponto inicial de processamento dos dados, nele os dados são reduzidos e os processamentos específicos à aplicação são

realizados. Os servidores da *Fog* também realizam processamento e redução de dados. Além do mais, eles servem como um banco de dados parcial, que guarda os dados mais buscados na região na qual ele se situa. O servidor de *Cloud* se situa no último nível da arquitetura; diferente dos outros servidores, ele é único. Ele guarda uma cópia de todos os dados importantes para a aplicação, e realiza processamentos que exigem um poder computacional muito maior que o servidor de *Fog* e *Edge* poderiam oferecer.

3.2 INTEROPERABILIDADE

Relembrando, o módulo de controle, presente em todas as camadas, é responsável pelo fluxo de dados dentro da arquitetura. Oferecido como serviço, pelo provedor da arquitetura, o módulo de controle faz o intermédio da troca de dados entre as aplicações. No pior caso para a troca de dados, a aplicação necessita passar por todas as três camadas da arquitetura para encontrar os dados. Assim como é demonstrado na Figura 3.8.

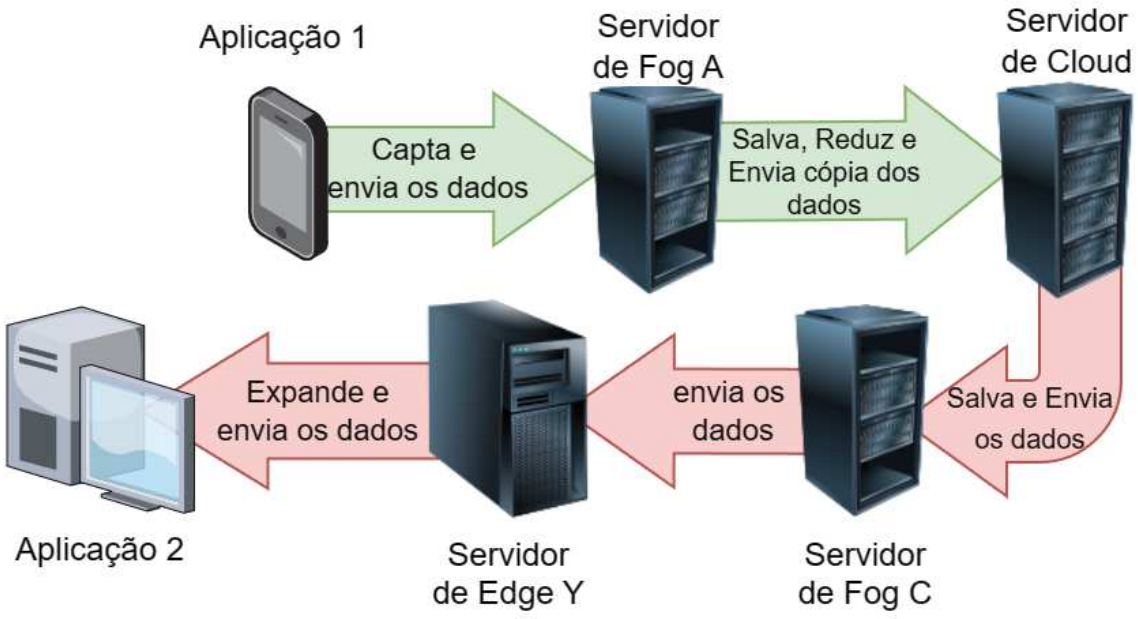

Figura 3.8 - Pior caso de interoperabilidade.

Na figura, os dados foram salvos no Servidor de *Fog* A e copiados para o Servidor de *Cloud* 1, no entanto, o Servidor de *Fog* C era o servidor de *Fog* mais próximo à aplicação que solicitou os dados. Nesse caso, foi necessário passar por todas as camadas para encontrarmos os dados requisitados pela Aplicação 2. Em contrapartida, no melhor caso, representado pela Figura 3.9, a aplicação encontra os dados no servidor mais perto de sua localização. Diferente do pior caso, o Servidor de *Fog* A é o servidor mais próximo de ambas as aplicações. Esse é considerado o melhor caso pois a jornada dos dados até a aplicação solicitante foi menor.

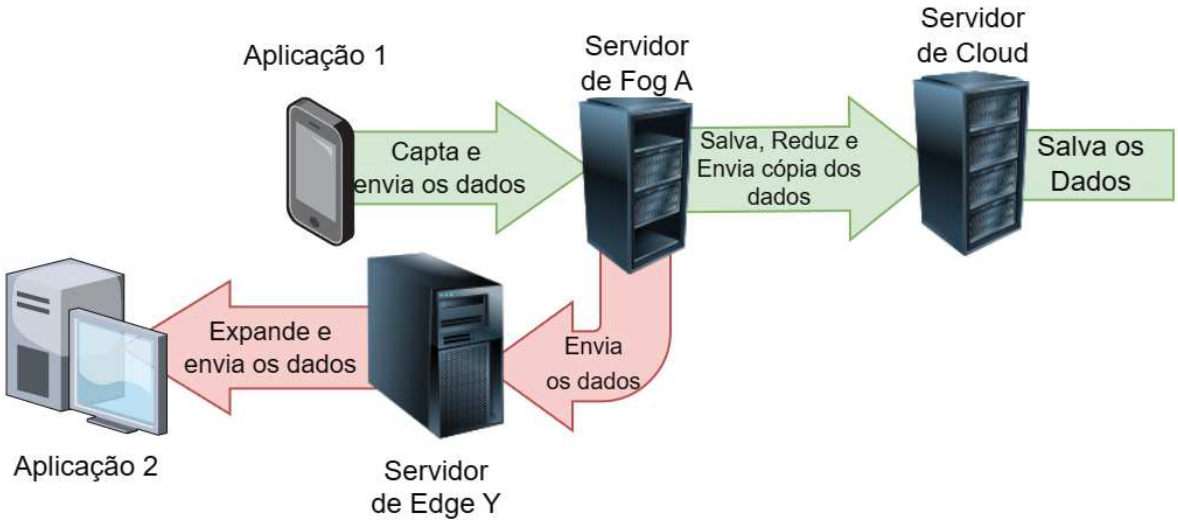

Figura 3.9 - Melhor caso de interoperabilidade.

Existem três possíveis formas de começar o processo de compartilhamento de dados. A primeira começa a partir da solicitação de compartilhamento de uma aplicação situada em um servidor de *Edge*. Na segunda, a solicitação é iniciada por uma aplicação em um servidor de *Fog*. A terceira começa com a interoperabilidade sendo requisitada pela aplicação em um servidor de *Cloud*. Cada processo de interoperabilidade possui etapas distintas a serem seguidas. Antes de detalhá-las, é preciso definir uma etapa comum a todos eles. Nessa etapa, a Aplicação 1 captou dados do Paciente A. Esses dados foram armazenados, no seu formato reduzido, no Servidor de *Fog* A e copiados para o Servidor de *Cloud* 1. A Figura 3.10 ilustra essa etapa.

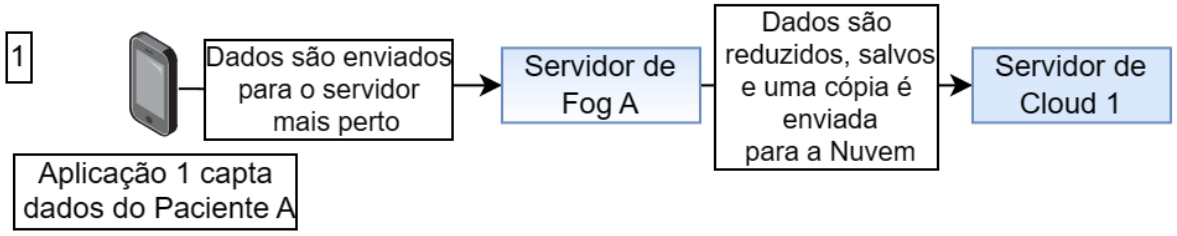

Figura 3.10 - Aplicação 1 capta e envia os dados do Paciente A para a arquitetura.

Após definir os locais onde os dados foram armazenados, o próximo passo é explicar as etapas de interoperabilidade entre a Aplicação 1 e Aplicação 2. A Figura 3.11 apresenta um modelo de alto nível do primeiro caso, quando a interoperabilidade é iniciada da *Edge*.

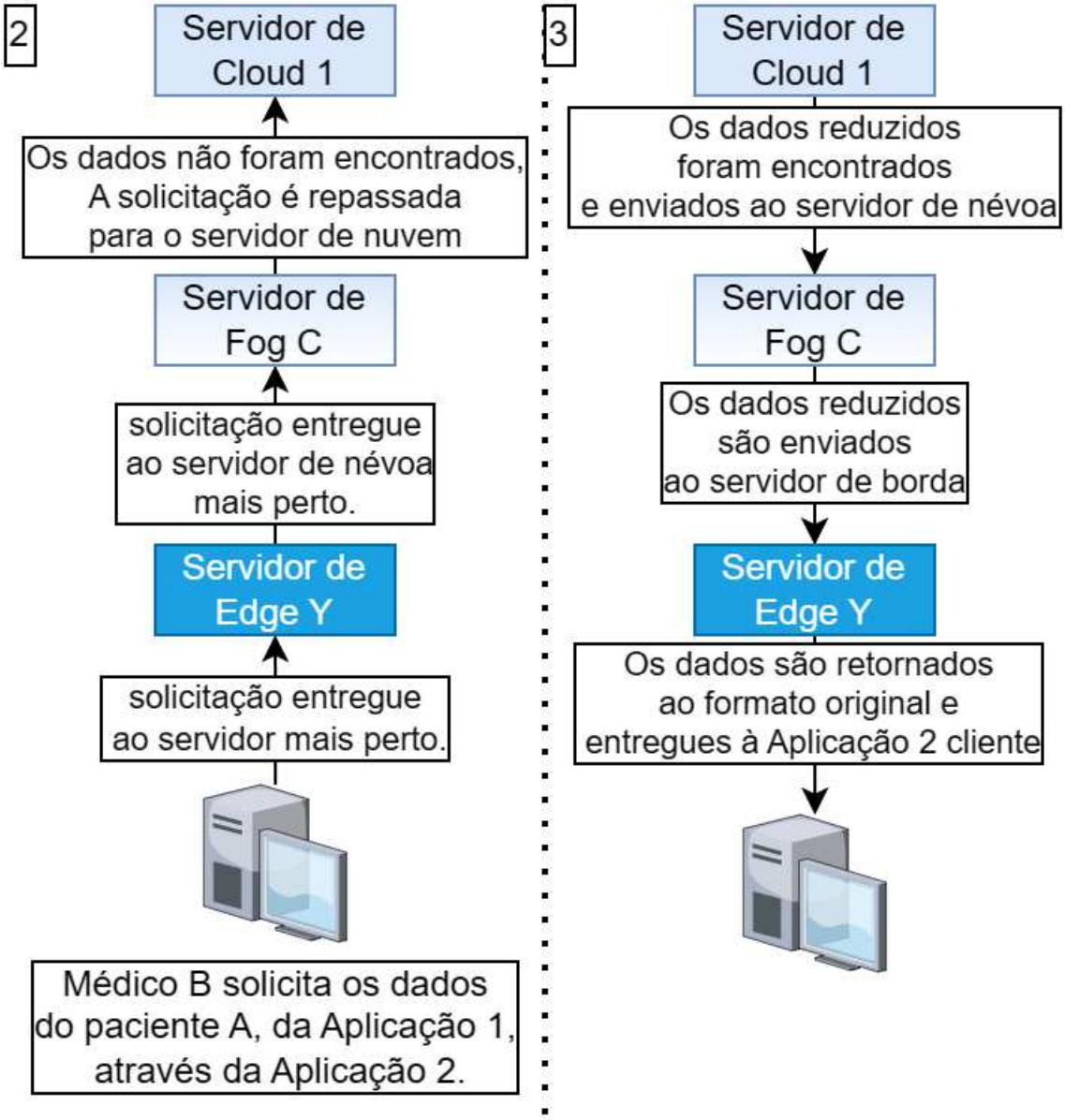

Figura 3.11 - Interoperabilidade iniciada em um servidor de *Edge*.

Esse processo possui mais etapas que os outros dois, no entanto, os dados percorrem um caminho maior em seu formato reduzido do que nos outros dois processos. A seguir é detalhado, em uma lista, o funcionamento interno dos servidores nesse processo:

1. Aplicação 2 no Servidor de *Edge* Y solicita ao seu Módulo de Controle os dados do Paciente A, pertencentes à Aplicação 1;

2. O Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y pergunta ao Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 o endereço do servidor de *Fog* mais próximo, contendo banco de dados da Aplicação 1;

a) O servidor mais perto é o próprio Servidor de *Cloud* 1, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna ao Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y o endereço do próprio Servidor de *Cloud* 1;

b) O Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y envia a solicitação dos dados ao Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1;

c) O Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 verifica se o seu acesso ao banco de dados da Aplicação 1 no Servidor de *Cloud* 1 está livre;

d) O processo é finalizado.

   i. Negou o acesso, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna um erro ao Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y;

   ii. Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y retorna à Aplicação 2 no Servidor de *Edge* Y que seu acesso foi negado;

   iii. O processo é finalizado.

e) Aceitou o acesso, é feita a busca dos dados;

   i. Os dados não foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna um erro ao Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y;

   ii. Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y retorna à Aplicação 2 no Servidor de *Edge* Y que os dados não foram encontrados;

   iii. O processo é finalizado.

f) Os dados foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna os dados reduzidos ao Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y;

g) Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y transforma os dados reduzidos de volta para o seu formato original;

h) Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y retorna à Aplicação 2 no Servidor de *Edge* Y os dados em seu formato original;

i) O processo é finalizado.

3. O Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna ao Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y o endereço do servidor de *Fog* mais próximo;

4. O Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y envia a solicitação dos dados ao Módulo de Controle da Aplicação 2 no Servidor de *Fog* C;

5. O Módulo de Controle da Aplicação 2 no Servidor de *Fog* C verifica se o seu acesso ao banco de dados da Aplicação 1 no Servidor de *Fog* C está livre;

a) Negou o acesso, o Módulo de Controle da Aplicação 2 no Servidor de *Fog* C retorna um erro ao Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y;

b) Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y retorna à Aplicação 2 no Servidor de *Edge* Y que seu acesso foi negado;

c) O processo é finalizado.

6. Aceitou o acesso, É feita a busca dos dados;

a) Os dados foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Fog* C retorna os dados reduzidos ao Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y;

b) Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y transforma os dados reduzidos de volta para o seu formato original;

c) Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y retorna à Aplicação 2 no Servidor de *Edge* Y os dados em seu formato original;

d) O processo é finalizado.

7. Os dados não foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Fog* C envia a solicitação dos dados ao Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1;

8. O Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 verifica se o seu acesso ao banco de dados da Aplicação 1 no Servidor de *Cloud* 1 está livre;

a) Negou o acesso, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna um erro ao Módulo de Controle da Aplicação 2 no Servidor de *Fog* C;

b) O Módulo de Controle da Aplicação 2 no Servidor de *Fog* C retorna um erro ao Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y;

c) Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y retorna à Aplicação 2 no Servidor de *Edge* Y que seu acesso foi negado;

d) O processo é finalizado.

9. Aceitou o acesso, É feita a busca dos dados;

a) Os dados não foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna um erro ao Módulo de Controle da Aplicação 2 no Servidor de *Fog* C;

b) O Módulo de Controle da Aplicação 2 no Servidor de *Fog* C retorna um erro ao Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y;

c) Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y retorna à Aplicação 2 no Servidor de *Edge* Y que os dados não foram encontrados;

d) O processo é finalizado.

10. Os dados foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna os dados reduzidos ao Módulo de Controle da Aplicação 2 no Servidor de *Fog* C;
11. O Módulo de Controle da Aplicação 2 no Servidor de *Fog* C retorna os dados reduzidos ao Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y;
12. Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y transforma os dados reduzidos de volta para o seu formato original;
13. Módulo de Controle da Aplicação 2 no Servidor de *Edge* Y retorna à Aplicação 2 no Servidor de *Edge* Y os dados em seu formato original;
14. O processo é finalizado.

O segundo caso, quando o processo é requisitado pela aplicação presente em um servidor de *Fog*, realiza uma quantidade menor de passos que o processo realizado na primeira possibilidade. A Figura 3.12 expõe como ele se desenvolve.

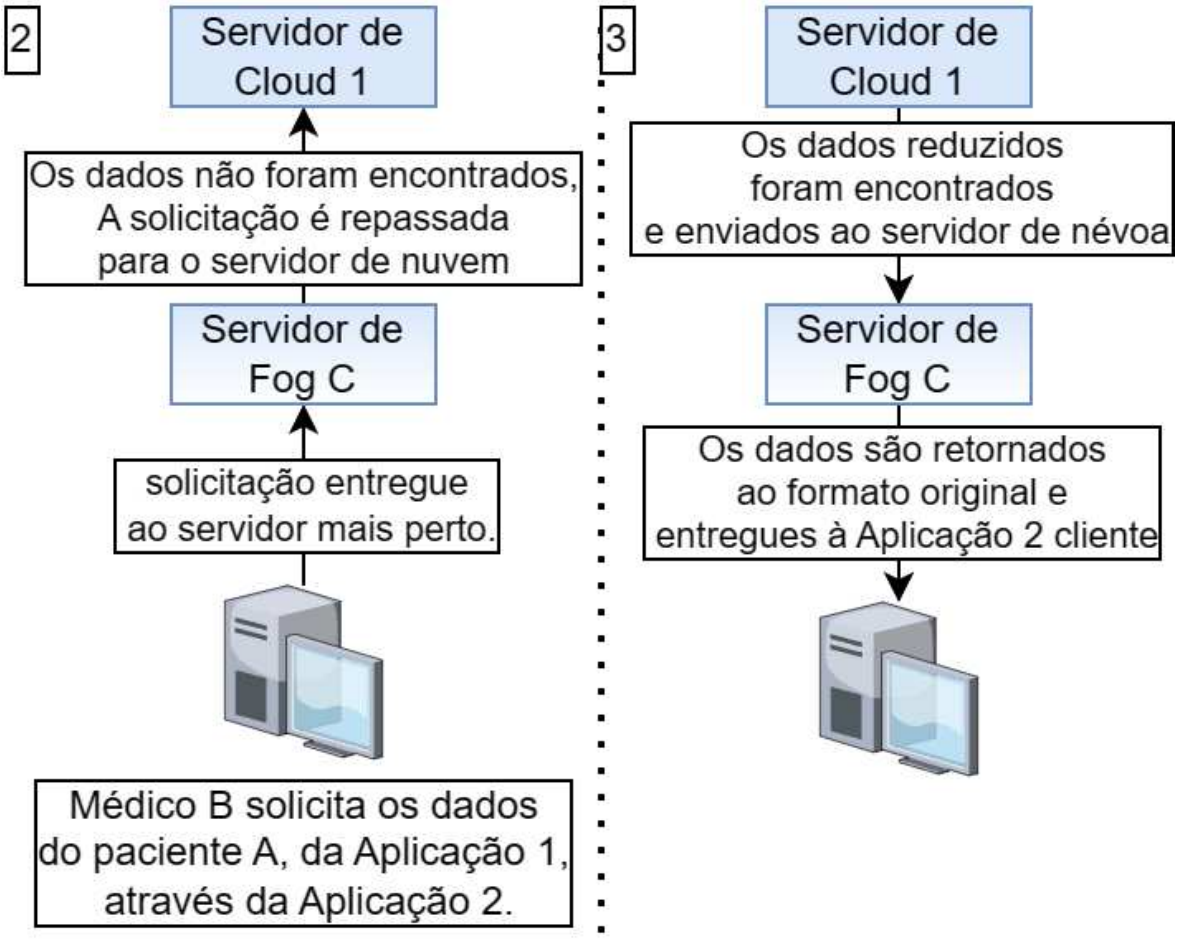

Figura 3.12 - interoperabilidade iniciada em um servidor de *Fog*.

A inexistência de servidores da *Edge* também reflete na quantidade de etapas a serem realizadas dentro dos servidores, como explicitado na lista a seguir:

1. Aplicação 2 no Servidor de *Fog* C solicita ao seu Módulo de Controle os dados do Paciente A, pertencentes à Aplicação 1;
2. O Módulo de Controle da Aplicação 2 no Servidor de *Fog* C verifica se o seu acesso ao banco de dados da Aplicação 1 no Servidor de *Fog* C está livre;
    a) O acesso foi negado, o processo é finalizado.
3. Aceitou o acesso, É feita a busca dos dados;
    a) Os dados foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Fog* C transforma os dados reduzidos de volta para o seu formato original;
    b) O Módulo de Controle da Aplicação 2 no Servidor de *Fog* C retorna à Aplicação 2 no Servidor de *Fog* C os dados em seu formato original;
    c) O processo é finalizado.
4. Os dados não foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Fog* C envia a solicitação dos dados ao Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1;
5. O Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 verifica se o seu acesso ao banco de dados da Aplicação 1 no Servidor de *Cloud* 1 está livre;
    a) Negou o acesso, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna um erro ao Módulo de Controle da Aplicação 2 no Servidor de *Fog* C;
    b) Módulo de Controle da Aplicação 2 no Servidor de *Fog* C retorna à Aplicação 2 no Servidor de *Fog* C que seu acesso foi negado;
    c) O processo é finalizado.
6. Aceitou o acesso, É feita a busca dos dados;
    a) Os dados não foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna um erro ao Módulo de Controle da Aplicação 2 no Servidor de *Fog* C;
    b) Módulo de Controle da Aplicação 2 no Servidor de *Fog* C retorna à Aplicação 2 no Servidor de *Fog* C que os dados não foram encontrados;
    c) O processo é finalizado.
7. Os dados foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna os dados reduzidos ao Módulo de Controle da Aplicação 2 no Servidor de *Fog* C;

8. Módulo de Controle da Aplicação 2 no Servidor de *Fog* C transforma os dados reduzidos de volta para o seu formato original;

9. Módulo de Controle da Aplicação 2 no Servidor de *Fog* C retorna à Aplicação 2 no Servidor de *Fog* C os dados em seu formato original;

10. O processo é finalizado.

Por último, a Figura 3.13 exemplifica o processo que possui menos etapas, que é solicitado pelo servidor de *Cloud*.

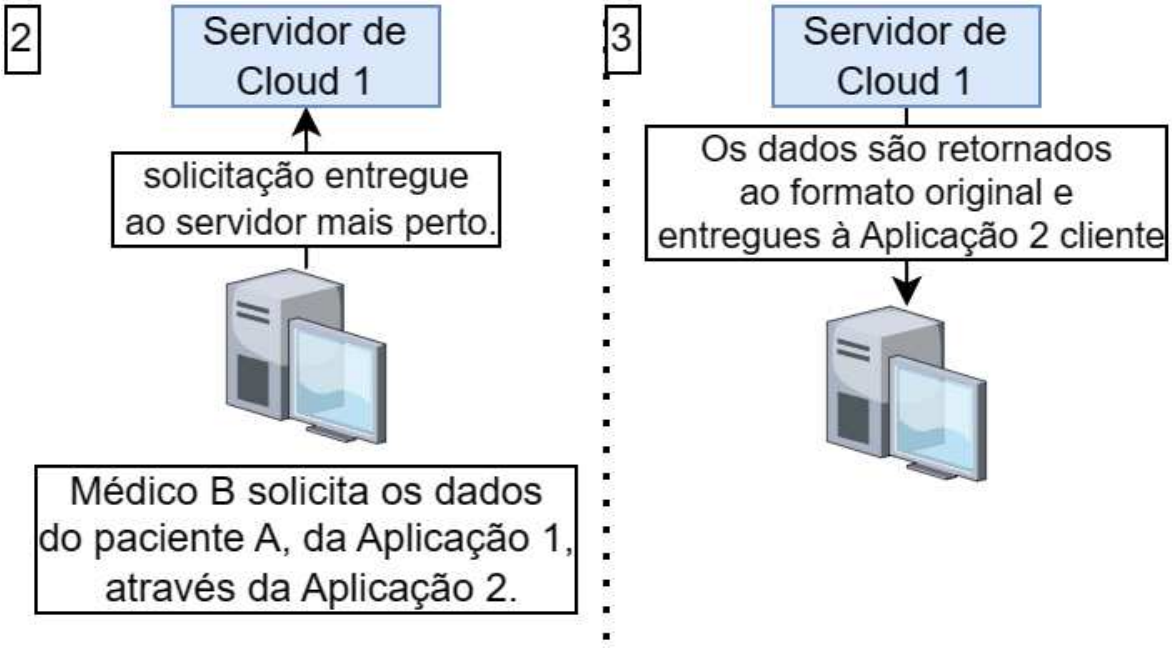

Figura 3.13 - interoperabilidade iniciada na *Cloud*.

Neste caso, a redução de dados não impacta o custo do transporte dos dados pela rede, pois os dados saem do Servidor de *Cloud* 1 em seu formato original e são encaminhados diretamente à aplicação cliente. A lista das etapas desse processo, que elaborada a seguir, elucida esse acontecimento.

1. Aplicação 2 no Servidor de *Cloud* 1 solicita ao seu Módulo de Controle os dados do Paciente A, pertencentes à Aplicação 1;

2. O Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 verifica se o seu acesso ao banco de dados da Aplicação 1 no Servidor de *Cloud* 1 está livre;

    a) Negou o acesso, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna à Aplicação 2 no Servidor de *Cloud* 1 que seu acesso foi negado;

b) O processo é finalizado.

3. Aceitou o acesso, É feita a busca dos dados;

   a) Os dados não foram encontrados, o Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna à Aplicação 2 no Servidor de *Cloud* 1 que os dados não foram encontrados;

   b) O processo é finalizado.

4. Os dados foram encontrados, Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 transforma os dados reduzidos de volta para o seu formato original;

5. Módulo de Controle da Aplicação 2 no Servidor de *Cloud* 1 retorna à Aplicação 2 no Servidor de *Cloud* 1 os dados em seu formato original;

6. O processo é finalizado.

Seguindo esses passos, nesses três casos, o Módulo de Controle garante que seja feito o intermédio da comunicação entre as aplicações implantadas na arquitetura. Ao realizar o intermédio, o Módulo de Controle certifica que a aplicação que fez a solicitação compreende os dados que serão entregues a ela, garantido a interoperabilidade, a redução e expansão dos dados.

3.3 ONTOLOGIA

Ontologias interpretaram um papel importante em nosso trabalho por possibilitarem o gerenciamento do vocabulário utilizado pelas aplicações. Com elas, ficou garantida a capacidade de interoperabilidade das aplicações. Apesar de não ser uma das funções propostas por ontologias, ela também auxiliou na redução dos dados. Essa abordagem diminuiu a complexidade do processo de redução dos dados. Ontologias também abrem possibilidades para trabalhos futuros. Uma dessas possibilidades é habilitar a interoperabilidade semântica entre aplicações, como Rubí e Gondim (2020) abordaram.

O Catálogo de Padrões de Interoperabilidade de Informações de Sistemas de Saúde, apresentado na Seção 2.6, lista o OpenEHR como o padrão semântico recomendado pelo governo para interoperabilidade entre aplicações de saúde no Brasil. A wiki oficial do padrão OpenEHR[2], recomenda o site da *Open Biological and Biomedical Ontology Foundry*[3] como um recurso relevante que diz respeito ao OpenEHR. Após estudar as ontologias disponibilizadas pelo site, a ontologia selecionada para este projeto foi a *Human Disease Ontology*, desenvolvida por Schriml et al. (2019). A principal função dessa ontologia é classificar as doenças raras e comuns que podem ser contraídas pelo ser humano moderno.

[2] Disponível em: https://openehr.atlassian.net/wiki/spaces/ontol/overview
[3] Disponível em: http://obofoundry.org/

Outrossim, seu escopo engloba os sintomas que o corpo humano pode apresentar, fenótipos, elementos causadores de doenças, suscetibilidades e outros termos relacionados à contração de doenças. Construído com a ferramenta Protégé[4], a Figura 3.14 demonstra o gráfico das principais classes da ontologia.

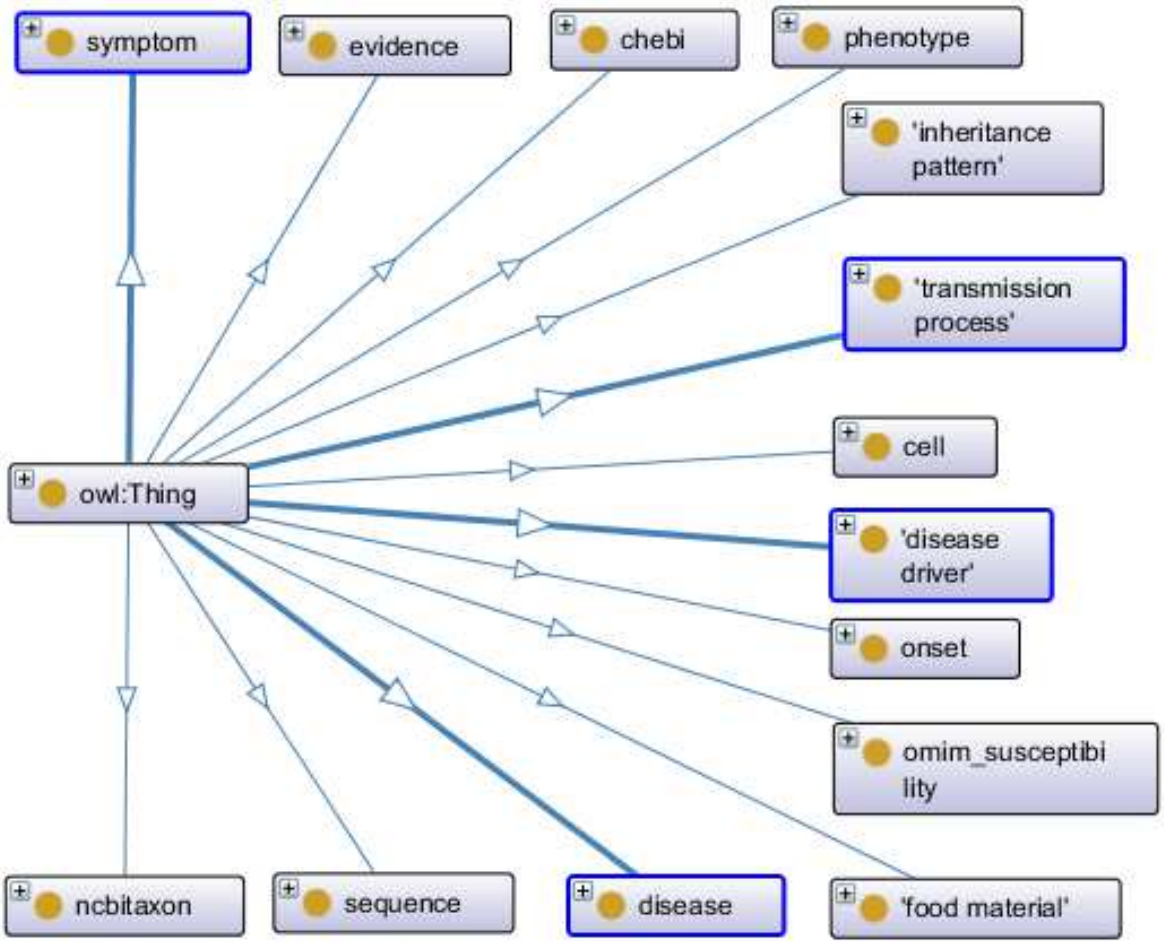

Figura 3.14 - Gráfico da estrutura com as principais classes da *Human Disease Ontology*.

Com o pressuposto de limitar o conjunto de dados com o qual a proposta trabalharia, este projeto aborda aplicações que especificamente auxiliam o atendimento médico. No cenário escolhido, o profissional de saúde estuda os sintomas que o paciente apresentou, os causadores da doença e o meio de transmissão/infecção. Por fim, ele diagnostica a doença de seu paciente. Como resultado, as classes da ontologia escolhidas foram *Symptom*, *Transmission Process*, *Disease Driver* e *Disease*. *Symptom* lista os diferentes tipos de sintomas que podem ser apresentados pelo paciente. *Transmission Process* diz respeito aos possíveis meios pelo qual o enfermo foi infectado. *Disease Driver* delimita os mecanismos genéticos ou ambientais que causam o surgimento de doenças. *Disease* enquadra as variadas doenças que o ser humano pode contrair.

[4] Disponível para download em: https://protege.stanford.edu/

3.4 REDUÇÃO DE DADOS

Toda classe ontológica é designada um valor no momento de sua criação, conhecido como *id*. Por padrão, o *id* é um valor único para cada classe. Igual a banco de dados, o *id* é um campo de valor único utilizado para identificar algo ao qual ele está associado. A Figura 3.15 apresenta como a ferramenta Protégé dispõe os campos de dados de uma classe.

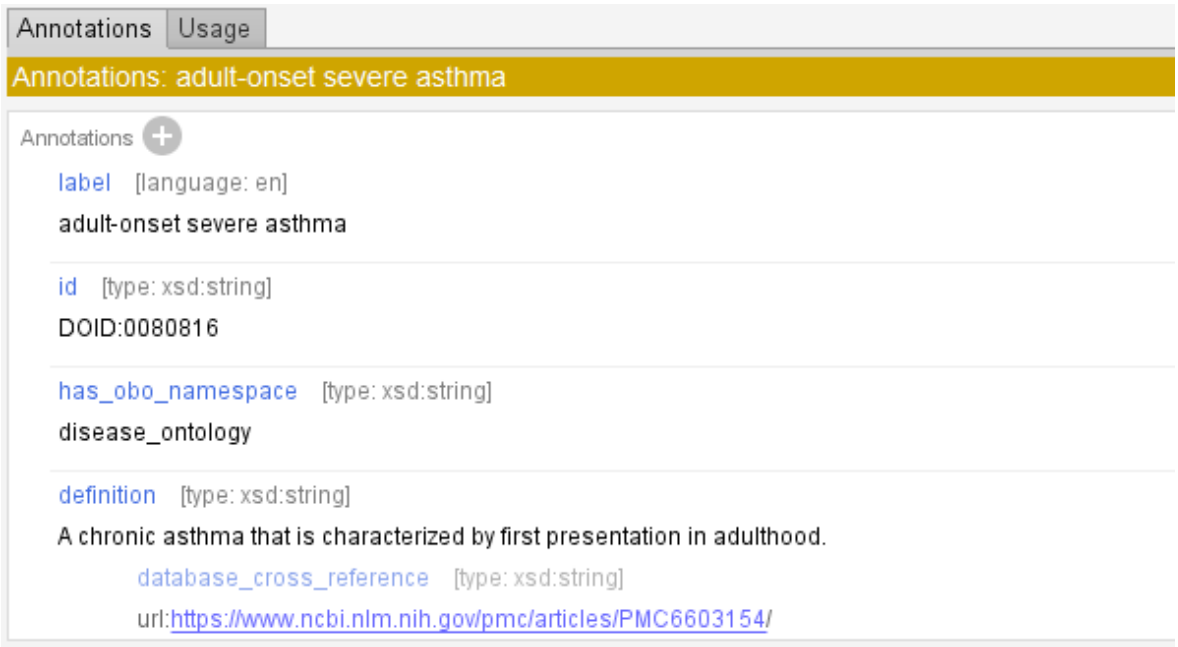

Figura 3.15 - Dados associados à classe *adult-onset severe asthma*, visualizados por meio da ferramenta Protégé.

Os campos de dados de nossa figura exemplo dizem respeito à classe *adult-onset severe asthma*, sendo seu *id* o segundo campo. O valor *id* é uma *string* composta por dois valores. A primeira parte é um conjunto de caracteres relacionado à classe raiz e a segunda é um número natural, separados por ':' (dois-pontos). O valor do *id* foi extraído para utilizá-lo como a versão reduzida do dado recebido. Quando um servidor de *Edge*/*Fog* recebe os dados originais, ele associa-os às suas devidas classes. Após esse processo ser devidamente concluído, o módulo de controle extrai da ontologia o *id* de cada classe. Pelo fato de o próprio *id* também ser uma *string*, o conjunto de caracteres foi substituído por um número natural e o símbolo de dois-pontos foi removido. O resultado desse processo é um número natural, que pode ser tratado como um inteiro sem sinal. Esses inteiros sem sinal são enviados ao servidor da próxima camada da arquitetura, até chegarem no servidor de *Cloud*. A Tabela 3.1 apresenta os valores escolhidos para cada um dos possíveis conjuntos de caracteres encontrados no *id*.

Tabela 3.1 – Conjuntos de caracteres do *id* e os números naturais utilizados para substituí-los.

Modelo	**Meses**	**Temperatura**
Disease	DOID	1
Disease Driver	DISDRIV	2
	CHEBI	3
	ExO	4
	NCIT	5
Symptom	SYMP	6
Transmission Process	TRANS	7

A Figura 3.16 ilustra o processo de redução dos dados. Utilizamos, mais uma vez, a interface do Protégé para facilitar a visualização do processo.

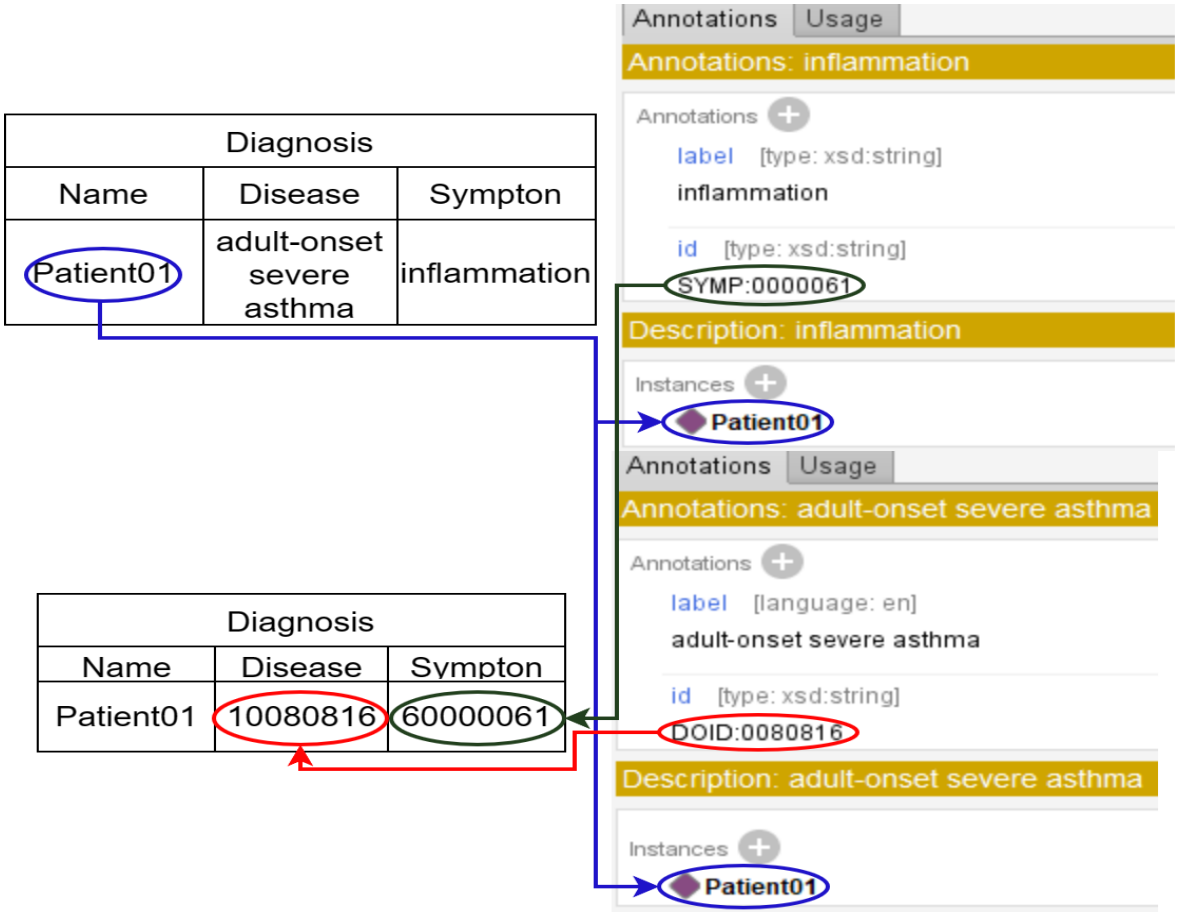

Figura 3.16 - Formato dos dados antes e depois de reduzidos.

Em SQL, um VARCHAR ocupa 1 byte para cada um de seus caracteres, mais 2 bytes que informam seu tamanho. O tamanho de um INT sem sinal é fixo em 4 bytes. Com esses 4 bytes, um INT sem sinal pode representar 4.294.967.296 (quatro bilhões, duzentos e noventa e quatro milhões, novecentos e sessenta e sete mil, duzentos e noventa e seis)

números diferentes. Em nosso exemplo, *adult-onset severe asthma* possui 25 caracteres, ou seja, ocupa 27 bytes. *Inflammation* possui 12 caracteres e ocupa 14 bytes. Neste caso, a redução diminuiu o tamanho dos dados de 41 bytes para 8 bytes, ou seja, diminuiu em aproximadamente 80,5% (oitenta, vírgula cinco cento). Vale apontar que esses valores levam em consideração apenas os dados que passaram pelo processo de redução.

3.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi apresentada a proposta deste projeto. Na introdução do capítulo, foram abordados o contexto avaliado, os problemas a serem solucionados e a arquitetura proposta. No corpo do capítulo, foi descrita detalhadamente a arquitetura proposta. Em sequência, foram apresentadas a descrição do processo de compartilhamento de dados na arquitetura proposta, a origem da ontologia utilizada, as características da ontologia e seu papel na arquitetura. Por fim, relatou-se o procedimento realizado pela técnica de redução de dados.

No próximo capítulo, serão tratados o ambiente de teste, o processo de construção do ambiente, detalhes sobre a execução dos testes e os resultados extraídos.

4 AMBIENTE DE TESTES E RESULTADOS

A estrutura necessária para testar o funcionamento da proposta exigiria altos gastos, os quais inviabilizariam a realização dos testes. Sendo assim, utilizou-se simuladores de contexto para avaliar a funcionalidade do conceito proposto. Por familiaridade, optou-se pelo simulador de contexto chamado Siafu. Em Costa, Dantas e David (2021), empregou-se Siafu para desenvolver uma simulação que produzisse dados para testes com a aplicação proposta. Em trabalhos anteriores, nos quais o autor desta dissertação participou, o programa se mostrou capaz de atender às exigências.

4.1 SIAFU - SIMUDOR DE CONTEXTO

Siafu é um simulador de contexto em larga-escala. Ele foi desenvolvido na linguagem Java e disponibilizado como código aberto. O seu ambiente proporciona a criação e a modificação de cenários, adição de agentes e execução de simulações. Descontinuado por seus desenvolvedores originais, mas atualizado pela comunidade de desenvolvedores, sua última versão está disponível no github[1]. A Figura 4.1 demonstra a interface provida pelo Siafu durante a execução de uma simulação.

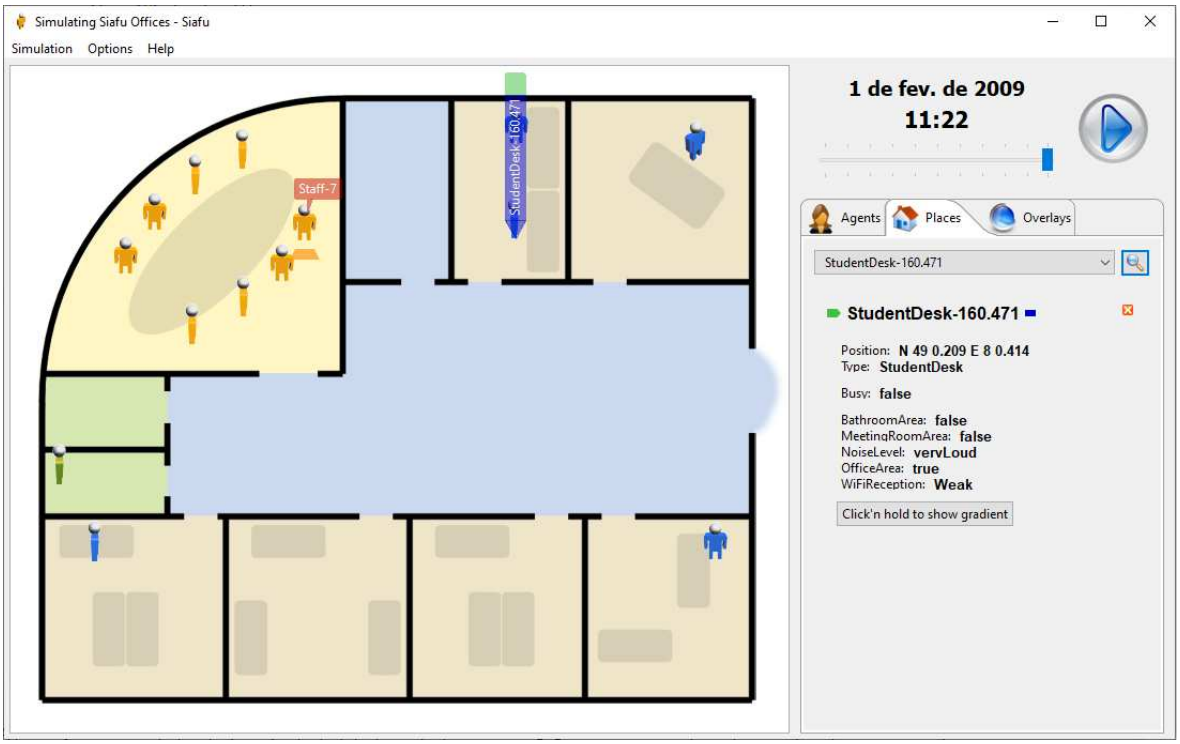

Figura 4.1 - Interface do Siafu executando a simulação nomeada *Office*, feita por Miguel Martin.

[1] Repositório do github:https://github.com/tfeijo/Siafu.

Existem 3 conceitos importantes no Siafu: agentes, locais e sobreposições. Agentes são os objetos que possuem a habilidade de se movimentar e carregar dados. Locais são lugares ou objetos inanimados que servem como marco de uma coordenada especial dentro do mapa da simulação. Sobreposições são utilizadas para aplicar valores sobre pontos específicos do mapa da simulação. A simulação, criada por Miguel Martin, representa um escritório. Salas para estudantes, salas para funcionários, banheiros e uma sala de conferência são os seus locais. Os agentes da simulação são os funcionários e estudantes/estagiários. A intensidade do sinal de *wi-fi* é uma das sobreposições presentes na simulação. Utilizando essa sobreposição, a simulação indica em quais pontos dentro do escritório o sinal de *wi-fi* é forte e em quais pontos o sinal é fraco. O objetivo da simulação é reproduzir o dia-a-dia de um escritório, onde os funcionários e estagiários trabalham, realizam reuniões e, ocasionalmente, vão ao banheiro.

4.2 CENÁRIO SIMULADO

O cenário construído baseou-se nas figuras apresentadas na Seção 3.1, especificamente, as Figuras 3.7, 3.10 e 3.11. Esses diagramas foram utilizados para simular o funcionamento da arquitetura proposta. A imagem de fundo é outro aspecto importante da simulação, escolhido durante seu planejamento. A imagem de fundo não afeta a execução, mas facilita reconhecer o contexto que a simulação tenta representar. Como é ilustrado na Figura 4.2, o mapa do Brasil foi escolhido para o contexto da simulação deste projeto.

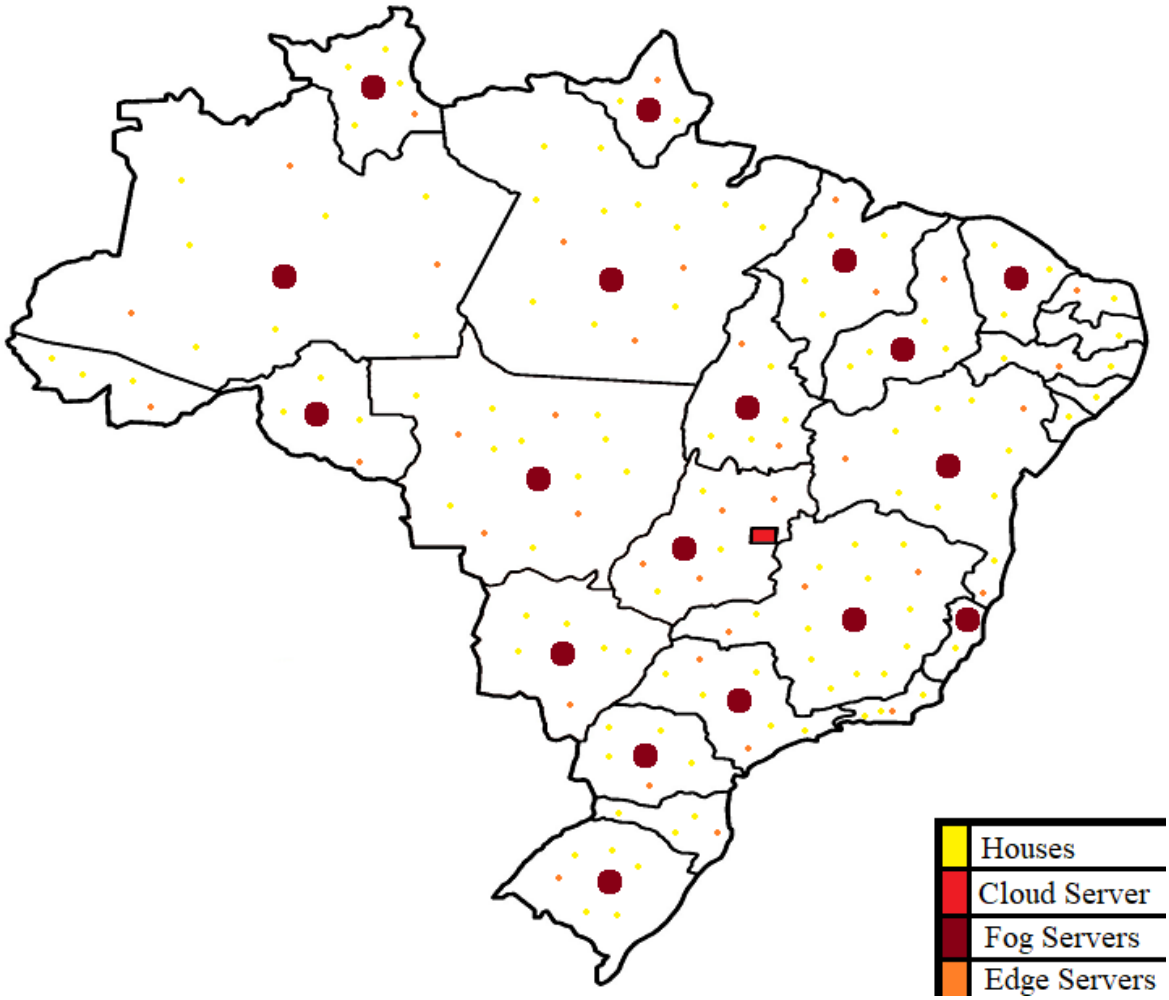

Figura 4.2 - Mapa utilizado como imagem de fundo na simulação desenvolvida para a proposta do trabalho.

Vários pontos, de diversas cores e tamanhos, e um retângulo foram espalhados pelo mapa. Os pontos pequenos representam os locais na borda de nossa arquitetura. Os pontos amarelos são as residências de pacientes. Esses pacientes fazem uso da aplicação chamada Aplicação 1. Os pontos alaranjados são hospitais, que possuem servidores de *Edge* em sua localidade. Os hospitais utilizam a Aplicação 2 e os servidores de *Edge* recebem, enviam, reduzem e expandem os dados da Aplicação 2. Os pontos maiores, de cor vermelha escura, interpretam o papel dos servidores de *Fog*. Os servidores de *Fog* recebem, enviam, armazenam, reduzem e expandem os dados de ambas as aplicações. Eles recebem os dados enviados por servidores de *Edge* e casas que estejam nas suas imediações. Os servidores de *Fog* também encaminham uma cópia dos dados para o servidor de *Cloud*. O retângulo vermelho claro indica a localização do servidor de *Cloud*. Ele realiza os mesmos processos que os servidores de *Fog*, mas seu poder computacional é muito maior e é responsável pelo mapa inteiro. O servidor de *Cloud* é o ponto final do fluxo de dados. O posicionamento dos servidores, casas e hospitais não representa qualquer infraestrutura do mundo real.

Os agentes foram a próxima etapa para o desenvolvimento de nossa simulação. Diferente da simulação *Office*, criada por Miguel Martin, os agentes da simulação deste projeto não representavam pessoas. Eles representam os dados que trafegam pela rede da arquitetura, como demonstrado na Figura 4.3.

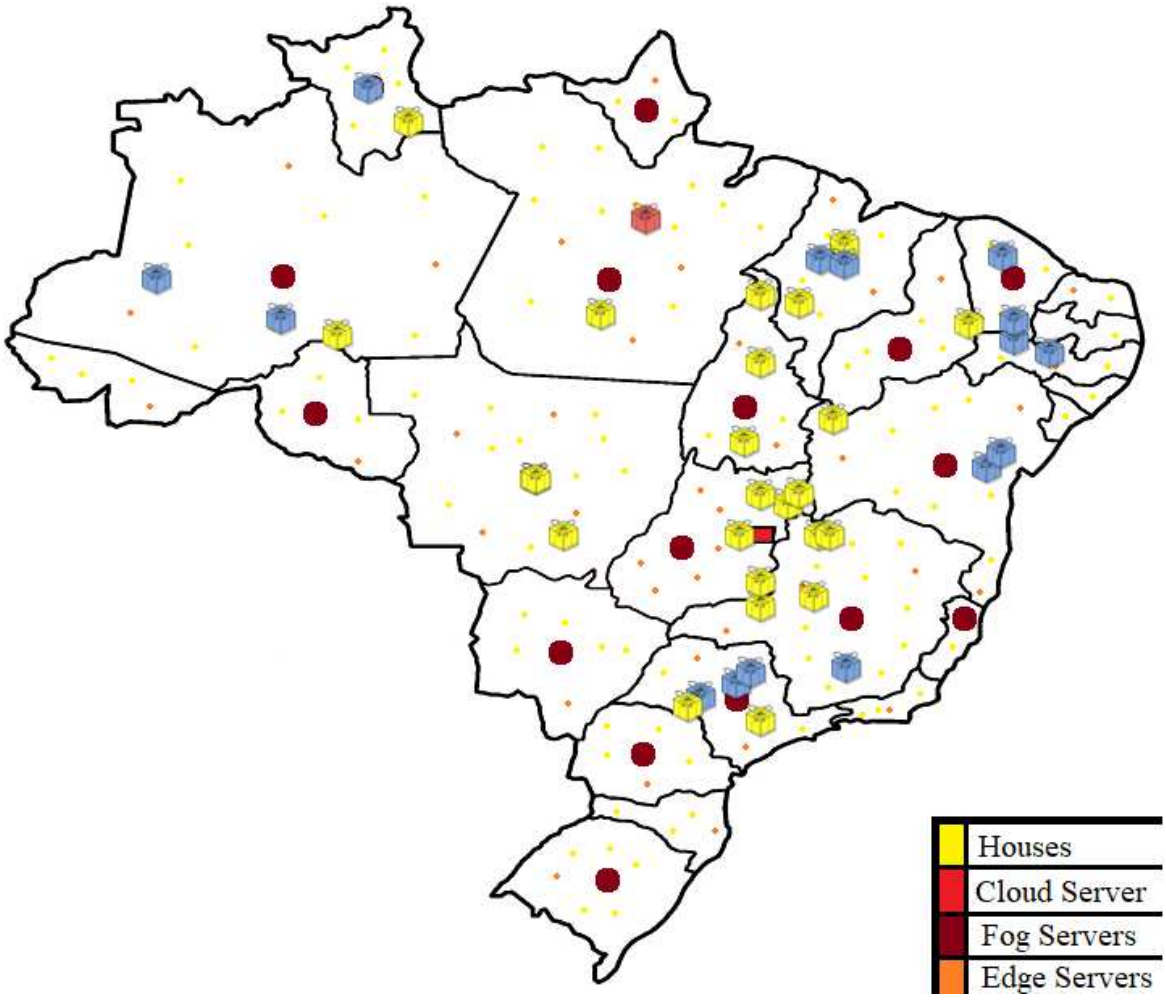

Figura 4.3 - Simulação demonstrando agentes, agindo como pacotes de dados trafegando pela arquitetura.

Como apontado anteriormente, as Figuras 3.10 e 3.11 foram usadas como base para o desenvolvimento da simulação, especificamente a forma como os agentes são tratados. Assim como na Figura 3.10, a Aplicação 1 envia dados para o servidor de *Fog* mais perto. O servidor de *Fog* reduz os dados, armazena e envia uma cópia dos dados reduzidos para o servidor de *Cloud*. Na simulação, a Aplicação 2 também envia dados, no entanto, diferente da Aplicação 1, os dados são enviados para o servidor de *Edge*. O servidor de *Edge* realiza a redução dos dados e os envia para o servidor de *Fog* mais próximo. Com os dados já reduzidos, o servidor de *Fog* só necessita armazená-los e enviar uma cópia para o servidor de *Cloud*. O servidor de *Cloud* apenas armazena os dados de ambas as aplicações.

As cores dos pacotes foram utilizadas para diferenciar visualmente os tipos de dados que os agentes estão carregando. Os pacotes azuis são dados não-reduzidos, os amarelos são dados reduzidos e o pacote vermelho indica os dados compartilhados entre as duas aplicações, por meio da interoperabilidade sintática.

Representado pelo pacote vermelho, o processo de interoperabilidade foi construído com base na Figura 3.11. Assim como na figura, a Aplicação 2 entrega a solicitação ao servidor de *Edge*. O servidor de *Edge* transfere a solicitação ao servidor de *Fog* mais perto. Caso os dados não estejam no servidor de *Fog*, o servidor de *Fog* encaminha a solicitação para o servidor de *Cloud*. Se os dados forem encontrados no servidor de *Cloud*, eles retornam pelo mesmo caminho até chegarem no servidor de *Edge*. No servidor de *Edge*, os dados são expandidos de volta para o seu formato original e entregues à Aplicação 2. Sobreposições não foram utilizadas, pois esse recurso não se mostrou necessário para o nosso contexto.

Além de possibilitar que os agentes se movimentem dentro da simulação e locais, sobreposições e agentes sejam adicionados ou removidos; Siafu permite que variáveis sejam associadas aos agentes e locais da simulação. a Tabela 4.1 lista as variáveis que foram associadas aos locais da simulação, a descrição de quais valores foram atribuídos a cada variável e a quais locais essas variáveis foram associadas.

Tabela 4.1 – Nomes das variáveis, suas descrições e locais aos quais elas foram associadas.

Variável	**Descrição**	**Locais**
APP1PACKAMOUNT	A quantidade de pacotes que o servidor recebeu da Aplicação 1.	servidor de *Cloud* servidor de *Fog*
APP2PACKAMOUNT	A quantidade de pacotes que o servidor recebeu da Aplicação 2.	servidor de *Cloud* servidor de *Fog*
APP1TOTALSIZE	A quantidade de dados, em bytes, que o servidor recebeu da Aplicação 1.	servidor de *Cloud* servidor de *Fog*
APP2TOTALSIZE	A quantidade de dados, em bytes, que o servidor recebeu da Aplicação 2.	servidor de *Cloud* servidor de *Fog*
APP1LASTPACKAGE	O último pacote que o servidor recebeu da Aplicação 1.	servidor de *Fog*
APP2LASTPACKAGE	O último pacote que o servidor recebeu da Aplicação 2.	servidor de *Fog*
INTEROPERABILITYPACK	Os dados, no seu formato original, do pacote que o servidor recebeu, da solicitação de interoperabilidade que a Aplicação 2 fez à Aplicação 1.	servidor de *Edge*
RINTEROPERABILITYPACK	Os dados reduzidos do pacote que o servidor recebeu, da solicitação de interoperabilidade que a Aplicação 2 fez à Aplicação 1.	servidor de *Edge*

Da mesma forma, variáveis foram associadas aos agentes da simulação. Diferente dos locais, todos os agentes necessitam possuir as mesmas variáveis, mesmo que não façam uso dela. Na Tabela 4.2 são apresentadas essas variáveis e a descrição de cada uma.

Tabela 4.2 – Variáveis associadas a todos os agentes e a descrição de cada uma.

Variável	**Descrição**
PDATA	Os dados que o pacote/agente está carregando
PSIZE	O tamanho, em bytes, dos dados que o pacote/agente está carregado
ORIGIN	O local de origem do pacote. Campo necessário para funcionamento da simulação.
ACTIVITY	Indica a atual atividade do agente. Campo necessário para funcionamento da simulação.
FOGTARGET	Servidor de *Fog* onde estão localizados os dados solicitados para interoperabilidade. Campo necessário para funcionamento da simulação.
TEMPDEST	Destino temporário pelo qual o pacote/agente necessita passar para chegar ao destino final. Campo necessário para funcionamento da simulação.

Apenas as variáveis PDATA e PSIZE foram utilizadas para os testes realizados na simulação. As demais variáveis foram adicionadas para que a simulação pudesse funcionar de acordo com o que fora projetado. PDATA e PSIZE são importantes para os testes, pois determinam a velocidade de movimento dos agentes dentro da simulação. O valor de PDATA é composto por valores aleatórios retirados da ontologia. Uma subclasse de *Disease*, uma de *Disease Driver*, uma de *Symptom* e uma de *Transmission Process* são escolhidas e o conjunto de seus *labels* é atribuído a PDATA. Como discutido na Seção 3.4, o tamanho de um VARCHAR é um byte para cada um de seus caracteres mais dois bytes de controle. Como PSIZE é o inteiro que diz respeito ao tamanho de PDATA em bytes, isso significa que PSIZE é a quantidade de caracteres em PDATA mais dois. PDATA pode ter seus dados reduzidos, nesse caso, o seu conteúdo passa de um VARCHAR para quatro valores INT sem sinal. Lembrando que o tamanho de um inteiro é igual a quatro bytes. Com PDATA reduzido, o cálculo para o valor de PSIZE passa a ser a quantidade de inteiros vezes quatro, ou seja, 4 bytes para cada inteiro. No contexto de nossa simulação, o valor de PSIZE sempre será dezesseis, quando o seu respectivo PDATA for um valor reduzido.

A velocidade de um agente dentro da simulação é ditada por um número inteiro positivo. Estabelecer uma velocidade 'X' ao agente, significa que o agente moverá X posições na grade por iteração da simulação. O valor 11(onze) foi escolhido como velocidade padrão para pacotes de dezesseis bytes. Valores mais altos impactavam negativamente o funcionamento da simulação, comprometendo os resultados dos testes. Baseando-se em pacotes de rede, é possível afirmar: quanto maior for o tamanho dos dados do pacote, mais tempo levará para completar a transmissão dos dados. Criando um paralelo na simulação,

menor será a velocidade do agente que representa o pacote de tamanho maior. Sendo assim, foi aplicada a regra de três, com grandezas inversamente proporcionais. A Equação 4.1 representa o cálculo da velocidade para pacotes com dados não reduzidos.

$$velocidade = (11 * 16)/PSIZE \tag{4.1}$$

Em outras palavras, se a velocidade de um pacote de dezesseis bytes é onze, então, a velocidade de um pacote com tamanho PSIZE será o resultado de onze vezes dezesseis, dividido por PSIZE.

4.3 TESTES E RESULTADOS

Com a interoperabilidade garantida pelo pacote de cor vermelha, o próximo passo foi comprovar que a redução de dados impacta o desempenho da arquitetura. Para isso, a simulação foi executada de duas formas. A primeira não utilizou redução de dados, enquanto a segunda aplicou a abordagem de redução proposta. Ao final de cada simulação, foi coletada a quantidade de pacotes e quantidade de dados, em bytes, que o servidor de *Cloud* recebeu de ambas as aplicações. Esses valores são apresentados pelas variáveis APP1PACKAMOUNT, APP2PACKAMOUNT, APP1TOTALSIZE e APP2TOTALSIZE, descritos na Tabela 4.1. Limitar a quantidade de agentes transitando simultaneamente foi outra configuração exclusiva para os testes. Se a quantidade de agentes transitando na simulação for igual à quantidade máxima permitida, um novo agente não poderá ser criado até que outro agente chegue ao seu destino final. Ao limitar a quantidade máxima de agentes, assegurou-se que o total de pacotes recebidos pelo servidor de *Cloud* esteja sujeito à velocidade dos pacotes. Por consequência, se a redução de dados diminuir efetivamente o tamanho dos dados dos pacotes, os agentes serão mais velozes e mais pacotes serão entregues. Ademais, a quantidade máxima de agentes foi dividida entre as duas aplicações. Se a quantidade de agentes atribuídos à aplicação for igual à quantidade máxima permitida, a aplicação não poderá criar outro agente até que um de seus agentes chegue ao seu destino final. Essa configuração garante que as características da estrutura contratada por uma aplicação, não afete os resultados da estrutura da outra aplicação. Uma vez que a Aplicação 1 contratou uma estrutura de *Fog* e *Cloud*, enquanto a Aplicação 2 contratou a estrutura completa de *Edge*, *Fog* e *Cloud*.

Assim, 8(oito) simulações, com duração de 42(quarenta e duas) horas cada, foram realizadas. Quatro simulações fizeram uso da abordagem de redução de dados proposta, enquanto as quatro restantes não utilizaram qualquer tipo de redução. Nas Figuras 4.4 e 4.5 são demonstradas, respectivamente, a quantidade de pacotes enviados pela Aplicação 1 e a quantidade de pacotes enviados pela Aplicação 2, recebidos pelo servidor de *Cloud*.

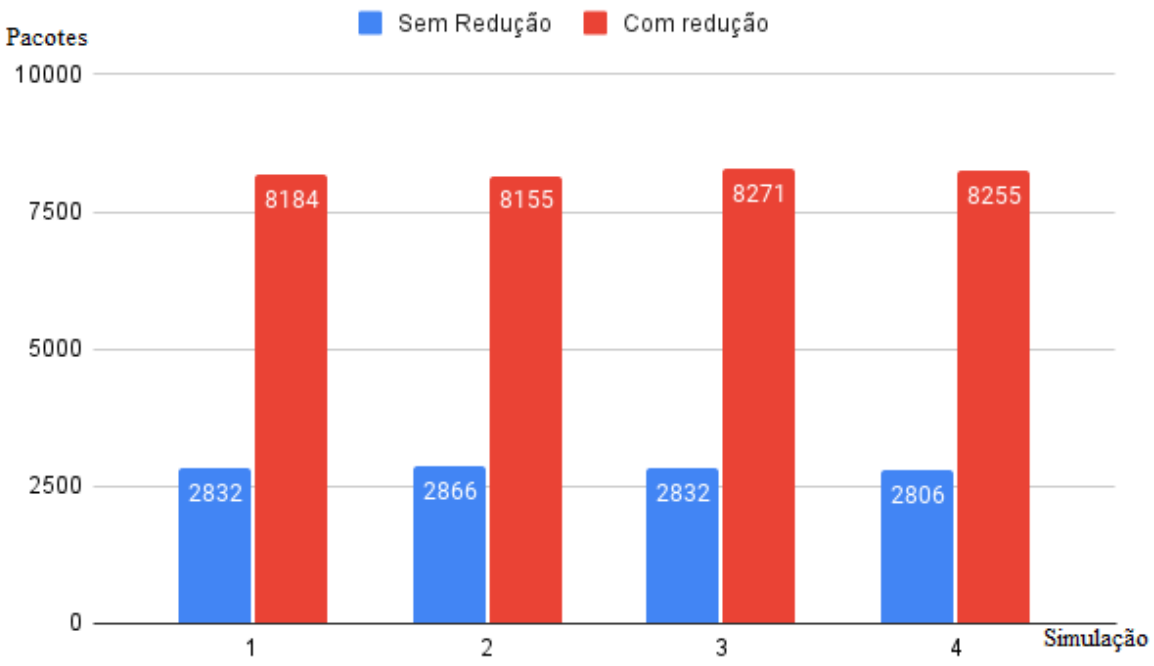

Figura 4.4 - Quantidade de pacotes enviados pela Aplicação 1, sem redução de dados e com redução de dados, recebidos pelo servidor de *Cloud*.

Relembrando a estrutura utilizada pela Aplicação 1, demonstrada na Figura 3.7, ela não faz uso de servidores de *Edge*. Isso significa que, na simulação com redução, os dados ainda precisam transitar a distância entre a camada da *Edge* e o servidor de *Fog* no seu formato não-reduzido. Não obstante, pela Figura 4.4, percebem-se que, em todas as iterações da simulação com redução, houve um aumento significativo da quantidade de pacotes entregues na faixa de tempo observada, quando comparadas às simulações sem redução de dados. A média de pacotes nas simulações com redução ficou em, aproximadamente, 3(três) vezes a média nas simulações sem redução.

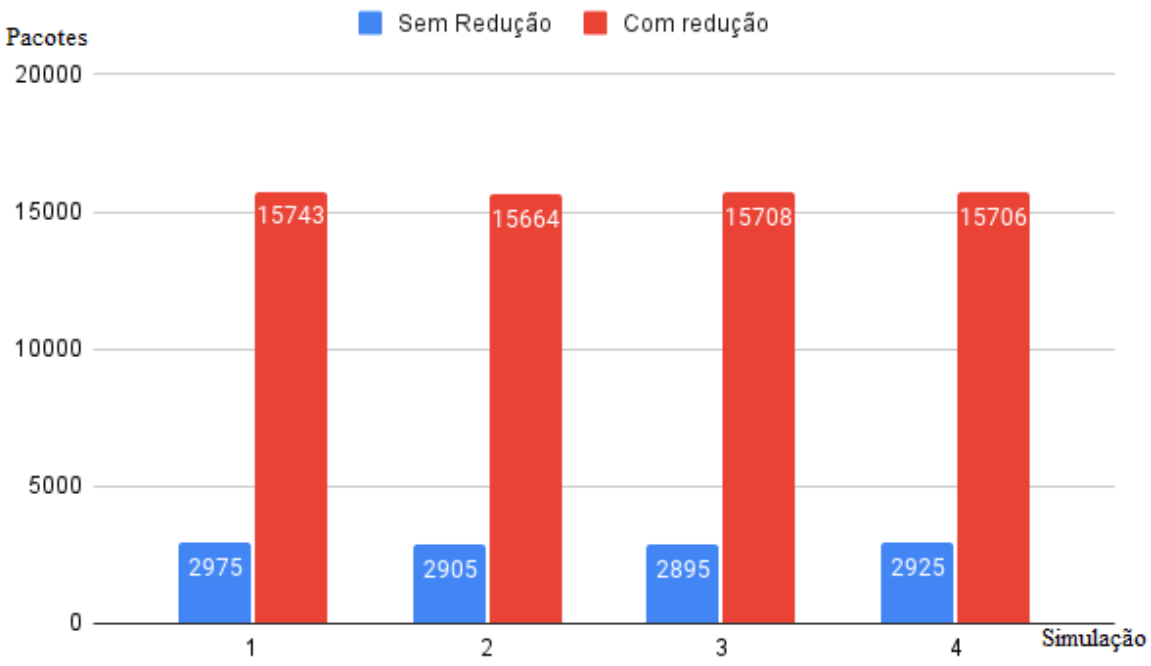

Figura 4.5 - Quantidade de pacotes enviados pela Aplicação 2, sem redução de dados e com redução de dados, recebidos pelo servidor de *Cloud*.

Na versão da simulação que utiliza a redução de dados, constatou-se um aumento ainda maior na quantidade de pacotes que o servidor de *Cloud* recebeu da Aplicação 2. Há de se lembrar que, a Aplicação 2, diferente da Aplicação 1, fez uso de servidores de *Edge*. O uso desses servidores minimiza a extensão que os pacotes necessitam trafegar com seus dados não-reduzidos. Utilizar servidores de *Edge* aumentou a média de pacotes entregues pela Aplicação 2 em quase 2(duas) vezes a quantidade de pacotes entregues pela Aplicação 1. Ao mesmo tempo em que a média da quantidade de pacotes entregues pela Aplicação 2 com redução ficou em cerca de 5(cinco) vezes a média de pacotes entregues pela Aplicação 2 sem redução.

Outro ponto importante observado foi a quantidade de dados, em bytes, que trafegaram pela rede e, consequentemente, foram armazenados no servidor de *Cloud*. Assim como foi feito para a quantidade de pacotes entregues, as quantidades de bytes enviados até o servidor de *Cloud* por cada aplicação foram observados separadamente. Na Figura 4.6, dispomos um gráfico com o total do fluxo de bytes da Aplicação 1, para cada uma das 8 simulações.

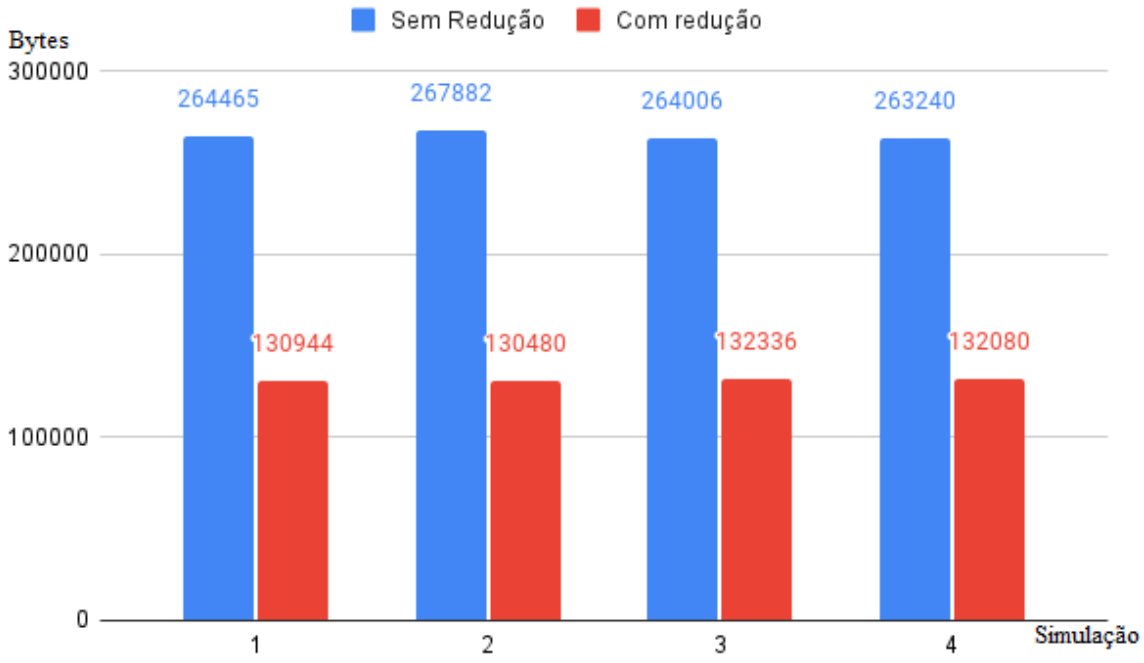

Figura 4.6 - Quantidade de bytes enviados pela Aplicação 1, sem redução de dados e com redução de dados, recebidos pelo servidor de *Cloud*.

Mesmo que cerca de 3(três) vezes mais pacotes tenham sido entregues, o total de bytes que necessitaram trafegar pela rede até o servidor de *Cloud* foi menor. A média do total de bytes, das simulações com redução, foi aproximadamente 50%(cinquenta por cento) da média das simulações sem redução. Esses resultados, para a Aplicação 1, foram observados em todas as simulações. A Figura 4.7 apresenta resultados da Aplicação 2, abordando seus dados da mesma maneira que foi feito com a Aplicação 1.

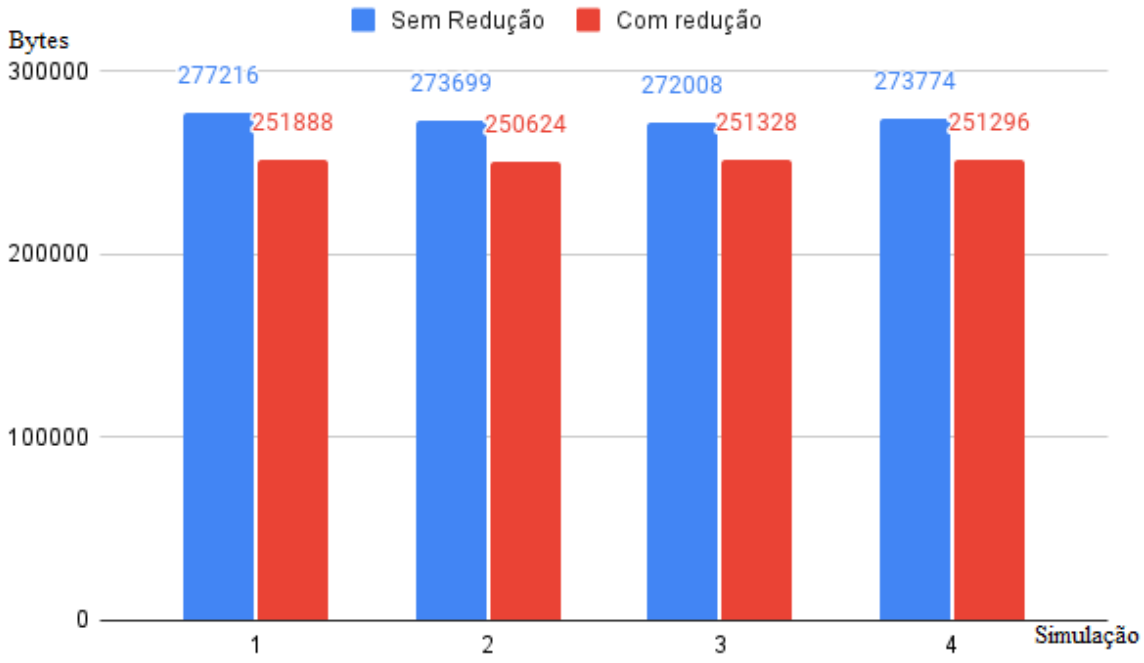

Figura 4.7 - Quantidade de bytes enviados pela Aplicação 2, sem redução de dados e com redução de dados, recebidos pelo servidor de *Cloud*.

Percebe-se que os resultados da quantidade total de bytes da Aplicação 2, que trafegaram pela arquitetura, foram muito maiores que os resultados da Aplicação 1. O valor da média do total de bytes que trafegaram pela rede até o servidor de *Cloud*, das simulações com redução, foi cerca de 90%(noventa por cento) do valor da média das simulações sem redução. Percebe-se que a quantidade de bytes entregues com redução conseguiu se equiparar à quantidade de bytes entregues sem redução. Esse acontecimento foi devido ao aumento significativo da quantidade de pacotes entregues na versão com redução.

Os resultados, ilustrados nas Figuras 4.4, 4.5, 4.6 e 4.7, apontam que a abordagem de redução de dados proposta teve um impacto no tráfego e no armazenamento de dados. Com a redução em uso, mais informações foram entregues, enquanto menos dados trafegaram pela rede e foram armazenados. Além do mais, o uso de servidores de *Edge* aumentaram significativamente a eficácia da abordagem.

Se os resultados dos testes fossem transladados para o ambiente físico real, eles significariam que os custos de armazenamento e transmissão de dados foram diminuídos. Isso confirma a eficácia da redução de dados e da estrutura *Edge-Fog-Cloud*, dentro do ambiente simulado. Sendo assim, a proposta obteve ambas a redução de dados e a interoperabilidade em uma mesma arquitetura. Não houveram conflitos em sua funcionalidade, ou esforço por parte das aplicações para compartilhar e reduzir dados

simultaneamente.

4.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi abordado o ambiente de testes, utilizado para apoiar a proposta, e os resultados dos testes. O capítulo iniciou com a explicação sobre a ferramenta utilizada para construir o ambiente, o simulador de contexto chamado Siafu. A simulação consiste da arquitetura proposta funcionando em um simulacro do território brasileiro. Os dados extraídos da simulação foram a quantidade de bytes e a quantidade de pacotes que trafegaram pela arquitetura. Cerca de 3(três) vezes mais pacotes foram entregues pela Aplicação 1 nas simulações com dados reduzidos. A Aplicação 2 entregou 5(cinco) vezes mais pacotes com redução de dados do que sem redução de dados. Aproximadamente, houve uma diminuição de 50%(cinquenta por cento) da quantidade de bytes enviados pela Aplicação 1 que trafegaram nas simulações com redução, quando comparado às simulações sem redução. Em contrapartida, a diminuição foi de 10%(dez por cento) da quantidade de bytes enviados pela Aplicação 2 que trafegaram nas simulações com redução, quando comparado às simulações sem redução. Com esses resultados, observou-se que as aplicações foram capazes de compartilhar dados, enquanto os dados eram reduzidos.

O próximo capítulo finaliza este trabalho de pesquisa com as conclusões sobre a solução proposta e algumas sugestões de desenvolvimentos futuros.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho abordou os desafios de empregar interoperabilidade em paralelo com redução de dados em *edge-fog-cloud computing*, com foco em *e-Health*. Ele propôs uma arquitetura *Edge-Fog-Cloud* na qual as funcionalidades são oferecidas como um serviço para aplicações médicas e de saúde. Essas funcionalidades possuem a peculiaridade de garantir certas qualidades, em nosso caso, interoperabilidade e redução de dados. O objetivo foi abordar a conciliação de interoperabilidade sintática e redução de dados, qualidades que entram em conflito quando abordadas com estratégias comuns. A conciliação destas duas qualidades foi vista como necessária pelo fato de ambas serem benéficas para as aplicações da saúde.

Para atingir o objetivo proposto, foram pesquisados trabalhos relacionados à *e-Health* que fizessem uso de interoperabilidade e/ou redução de dados. Também pesquisou-se por trabalhos gerais que abordavam interoperabilidade, técnicas de redução de dados, Computação em *Cloud*, Computação em *Fog*, Computação em *Edge*, abordagens híbridas para Computação em *Edge-Fog-Cloud* e Web Linguagem Ontológica. Com a base de conhecimento necessária completa, optou-se por utilizar o Siafu, para simular a solução funcionando em um contexto prático. Os dados extraídos da solução foram usados para verificar se a solução estava efetivamente reduzindo os dados, enquanto os aplicativos na simulação compartilham dados entre si. Os resultados dos testes apontam que a solução é capaz de realizar tal feito.

A interoperabilidade contém muitos níveis. Cada nível exige mais esforço das aplicações para extrair novas informações dos dados compartilhados. Sendo a sintática um dos níveis mais baixos e também o nível que foi abordado neste documento. Seria interessante aplicar esta solução a níveis superiores de interoperabilidade. Tal empreendimento pode abrir uma nova gama de informações para aplicações relacionadas à saúde, ao mesmo tempo em que aproveitam os benefícios da redução de dados. Seguinte, a abordagem poderia ser modificada para atender a interoperabilidade entre aplicações de outras áreas, além de aplicações de *e-Health*.

Apesar deste projeto utilizar especificamente ontologias para reduzir dados, seria possível utilizar outras técnicas de redução com a arquitetura proposta. Uma possível vertente para este projeto seria realizar um estudo, para apontar quais técnicas de redução poderiam ser utilizadas. Ainda sobre desenvolvimentos relacionados à redução de dados, há a possibilidade de desenvolver uma abordagem capaz de selecionar a técnica de redução de dados de acordo com o contexto da aplicação a ser atendida.

O desenvolvimento desta pesquisa possibilitou na participação da publicação dos seguintes artigos:

- Uma Abordagem Computacional em Ambientes Assistidos Baseada em Atividades Físicas e Biometeorologia de Costa, Dantas e David (2021), **em Anais Estendidos**

do XXI Simpósio Brasileiro de Computação Aplicada à Saúde;

- Sistema de Apoio ao Diagnóstico Baseado em Sintomas da Covid-19 e Biometeorologia de Costa et al. (2022), em **Anais do XVII Simpósio Brasileiro de Sistemas Colaborativos**;
- Nutrinprice: Uma Plataforma Colaborativa para Apoiar a Seleção de Produtos Alimentícios de Alves et al. (2022a), em **Anais do XVII Simpósio Brasileiro de Sistemas Colaborativos**;
- An Architecture for Food Product Recommendation Focusing on Nutrients and Price de Alves et al. (2022b), em **Intelligent Information Systems**.
- Functionalities as a Service - An Approach to Conciliate Interoperability and Data Reduction in E-Health de Costa et al. (2023), em **Anais do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho**;

REFERÊNCIAS

ABUKHOUSA, E.; MOHAMED, N.; AL-JAROODI, J. e-health cloud: opportunities and challenges. *Future internet*, MDPI, v. 4, n. 3, p. 621–645, 2012.

AL-ANSI, A. et al. Survey on intelligence edge computing in 6g: Characteristics, challenges, potential use cases, and market drivers. *Future Internet*, Multidisciplinary Digital Publishing Institute, v. 13, n. 5, p. 118, 2021.

ALAM, T. Cloud computing and its role in the information technology. *IAIC Transactions on Sustainable Digital Innovation (ITSDI)*, v. 1, n. 2, p. 108–115, 2020.

ALVES, R. et al. Nutrinprice: uma plataforma colaborativa para apoiar a seleção de produtos alimentícios. In: *Anais do XVII Simpósio Brasileiro de Sistemas Colaborativos*. Porto Alegre, RS, Brasil: SBC, 2022. p. 9–16. ISSN 2326-2842. Disponível em: <https://sol.sbc.org.br/index.php/sbsc/article/view/19470>.

ALVES, R. d. D. et al. An architecture for food product recommendation focusing on nutrients and price. In: WEERDT, J. D.; POLYVYANYY, A. (Ed.). *Intelligent Information Systems*. Cham: Springer International Publishing, 2022. p. 1–9. ISBN 978-3-031-07481-3.

CAO, K. et al. An overview on edge computing research. *IEEE access*, IEEE, v. 8, p. 85714–85728, 2020.

COSTA, J. et al. Functionalities as a service - an approach to conciliate interoperability and data reduction in e-health. In: *Anais do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho*. Porto Alegre, RS, Brasil: SBC, 2023. p. 193–204. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/wscad/article/view/26520>.

COSTA, J. P.; DANTAS, M. A.; DAVID, J. M. Uma abordagem computacional em ambientes assistidos baseada em atividades físicas e biometeorologia. In: *Anais Estendidos do XXI Simpósio Brasileiro de Computação Aplicada à Saúde*. Porto Alegre, RS, Brasil: SBC, 2021. p. 121–126. ISSN 2763-8987. Disponível em: <https://sol.sbc.org.br/index.php/sbcas_estendido/article/view/16112>.

COSTA, J. P. da et al. Sistema de apoio ao diagnóstico baseado em sintomas da covid-19 e biometeorologia. In: *Anais do XVII Simpósio Brasileiro de Sistemas Colaborativos*. Porto Alegre, RS, Brasil: SBC, 2022. p. 17–24. ISSN 2326-2842. Disponível em: <https://sol.sbc.org.br/index.php/sbsc/article/view/19471>.

EYSENBACH, G. et al. What is e-health? *Journal of medical Internet research*, JMIR Publications Inc., Toronto, Canada, v. 3, n. 2, p. e833, 2001.

HORRIDGE, M. et al. A practical guide to building owl ontologies using protégé 4 and co-ode tools edition1. 3. *The university of Manchester*, v. 107, 2009.

IDREES, S. K.; IDREES, A. K. New fog computing enabled lossless eeg data compression scheme in iot networks. *Journal of Ambient Intelligence and Humanized Computing*, Springer, p. 1–14, 2022.

IROJU, O. et al. Interoperability in healthcare: benefits, challenges and resolutions. *International Journal of Innovation and Applied Studies*, ISSR Innovative Space of Scientific Research Journals, v. 3, n. 1, p. 262–270, 2013.

KAHDIM, A. N.; MANAA, M. E. Design an efficient internet of things data compression for healthcare applications. *Bulletin of Electrical Engineering and Informatics*, v. 11, n. 3, p. 1678–1686, 2022.

PIOLI, L. et al. An overview of data reduction solutions at the edge of iot systems: a systematic mapping of the literature. *Computing*, Springer, p. 1–23, 2022.

RINTY, M. R.; PRODHAN, U. K.; RAHMAN, M. M. A prospective interoperable distributed e-health system with loose coupling in improving healthcare services for developing countries. *Array*, Elsevier, v. 13, p. 100114, 2022.

RUBÍ, J. N. S.; GONDIM, P. R. d. L. Interoperable internet of medical things platform for e-health applications. *International Journal of Distributed Sensor Networks*, SAGE Publications Sage UK: London, England, v. 16, n. 1, p. 1550147719889591, 2020.

SABIREEN, H.; NEELANARAYANAN, V. A review on fog computing: Architecture, fog with iot, algorithms and research challenges. *Ict Express*, Elsevier, v. 7, n. 2, p. 162–176, 2021.

SCARPATO, N. et al. E-health-iot universe: A review. *management*, v. 21, n. 44, p. 46, 2017.

SCHRIML, L. M. et al. Human disease ontology 2018 update: classification, content and workflow expansion. *Nucleic acids research*, Oxford University Press, v. 47, n. D1, p. D955–D962, 2019.

SHULL, J. G. Digital health and the state of interoperable electronic health records. *JMIR medical informatics*, JMIR Publications Inc., Toronto, Canada, v. 7, n. 4, p. e12712, 2019.

SRIVASTAVA, P.; KHAN, R. A review paper on cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, v. 8, n. 6, p. 17–20, 2018.

WELTY, C.; MCGUINNESS, D. L.; SMITH, M. K. Owl web ontology language guide. *W3C recommendation, W3C (February 2004) http://www. w3. org/TR/2004/REC-owl-guide-20040210*, p. 48, 2004.

YOUSEFPOUR, A. et al. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, Elsevier, v. 98, p. 289–330, 2019.

APÊNDICE A – PUBLICAÇÕES COMO AUTOR PRINCIPAL

DA COSTA, João Pedro de S. J.; DANTAS, Mario A. R.; DAVID, José Maria N.; BRAGA, Regina M. M.; MORENO, Marcelo F.; MENEZES, Victor Stroele de Andrade; ALVES, Rian das Dores; DA SILVA, Izaque Esteves. Sistema de Apoio ao Diagnóstico Baseado em Sintomas da COVID-19 e Biometeorologia. In: SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS (SBSC), 17. , 2022, Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2022 . p. 17-24. ISSN 2326-2842. DOI: https://doi.org/10.5753/sbsc.2022.19471.

Resumo: A pandemia da COVID-19 tem demandado vários estudos sobre estratégias para combatê-la. Contudo, no que diz respeito à identificação de infectados, existe a possibilidade de pessoas apresentarem os sintomas sem estarem com a doença. Um dos fatores que influenciam na manifestação de sintomas, relacionados à COVID-19, são as condições climáticas. Este trabalho apresenta uma arquitetura de suporte à decisão, que tem o objetivo de elucidar a influência das condições climáticas sobre os sintomas da COVID-19 e, com o apoio de videoconferências, auxiliar a colaboração para tomada de decisões no combate à pandemia. Desenvolver uma base de conhecimento, através de técnicas inteligentes, mostrou-se necessário para alcançar nosso objetivo.

Qualis: B1

COSTA, João Pedro de S. J. da; DANTAS, Mário A. R.; DAVID, José Maria N.; SILVA, Fernando de Almeida. Functionalities as a Service - An Approach to Conciliate Interoperability and Data Reduction in E-Health. In: SIMPÓSIO EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO (WSCAD), 24. , 2023, Porto Alegre/RS. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2023 . p. 193-204. DOI: https://doi.org/10.5753/wscad.2023.235668.

Abstract: Interoperability and data reduction have been proven beneficial to health and medical applications that deal with large datasets. Still, the conflicts between these qualities turned out to be a problem for their conciliation. This paper presents an Edge-Fog-Cloud architecture that offers functionalities as a service. These functionalities can guarantee certain qualities depending on the necessities of the client, such as, in our case, interoperability and data reduction. With the use of context simulators, we found that it was possible to significantly increase the output of data delivered to the servers, and decrease the size of the data that transitions in the network and is stored in the servers' databases, without interfering with the syntactic interoperability.

Qualis: B3

COSTA, João Pedro de Souza Jardim da; DANTAS, Mário Antônio Ribeiro; DAVID, José Maria Nazar. Uma Abordagem Computacional em Ambientes Assistidos Baseada em Atividades Físicas e Biometeorologia. In: CONCURSO DE TRABALHOS DE INICIAÇÃO CIENTÍFICA - SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO APLICADA À SAÚDE (SBCAS), 21. , 2021, Evento Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021 . p. 121-126. ISSN 2763-8987.
DOI: https://doi.org/10.5753/sbcas.2021.16112.

Resumo: A importância de realizar atividades físicas para preservar a saúde pessoal é um conhecimento comum a toda sociedade humana. Uma pergunta a ser considerada seria: o desempenho, ao realizar essas atividades, é afetado pelo ecossistema que cerca as pessoas? Afetar o desempenho significa afetar o rendimento e, em contrapartida, implica afetar a eficiência na realização da atividade.Este trabalho de pesquisa apresenta uma proposta de uma aplicação que visa amparar profissionais da área de educação física no acompanhamento de seus alunos, enquanto leva em consideração os elementos meteorológicos que os afetam. Outra contribuição diferencial foi a simulação desenvolvida visando uma melhor visualização do ambiente.

Qualis: A4

APÊNDICE B – PUBLICAÇÕES COMO CO-AUTOR

ALVES, Rian das Dores; DAVID, José Maria; BRAGA, Regina Maciel; SIQUEIRA, Kennya; STROELE, Victor; BARBOSA, Guilherme; **DA COSTA, João Pedro de S. J.**; DA SILVA, Izaque Esteves. NutrinPrice: uma plataforma colaborativa para apoiar a seleção de produtos alimentícios. In: SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS (SBSC), 17. , 2022, Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2022 . p. 9-16. ISSN 2326-2842.
DOI: https://doi.org/10.5753/sbsc.2022.19470.

Resumo: Parte da população apresenta deficiência nutricional devido às más condições de alimentação. O custo dos produtos alimentícios e o desconhecimento dos nutrientes são fatores relevantes para a má nutrição. Selecionar produtos ricos nutricionalmente e com baixo custo são desafios. Este trabalho propõe uma plataforma para captação, organização e visualização de dados e preços dos produtos consumidos. A solução busca, por meio da colaboração de usuários e supermercados, prover informações referentes aos produtos, de modo a apoiar a seleção desses produtos. Como resultado, buscamos mitigar o problema da má nutrição na sociedade. Um estudo de viabilidade foi conduzido a partir dos dados capturados.

Qualis: B1

ALVES, Rian das Dores; DAVID, José Maria; BRAGA, Regina Maciel; SIQUEIRA, Kennya; BARBOSA, Guilherme; **DA COSTA, João Pedro de S. J.**; STROELE, Victor; BARRÉRE, Eduardo. An Architecture for Food Product Recommendation Focusing on Nutrients and Price.

Abstract: Part of the world's population is nutrient deficient, a phenomenon known as hidden hunger. Poor eating conditions cause this deficiency, leading to illnesses and recovery difficulties. Malnourished patients are more easily affected by Covid-19 and have a difficult recovery after the illness. An effective food choice has the price and nutritional value of food products as the most relevant factors, with the price being the most relevant, considering the context of countries such as Brazil. Thus, having identified a scenario in which the access and food price mainly cause malnutrition. This work proposes an architecture, called Nutri'n Price, to recommend high nutritional foods with low costs. The architecture encompasses a network of ontologies, inference algorithms, information retrieval and collaborative filtering techniques to recommend the best foods according to nutrient choice, price, and user contextual information. A prototype of a mobile application was developed to evaluate the feasibility of the proposed architecture.

Qualis: A3