

UNIVERSIDADE CATÓLICA DE PELOTAS
CENTRO DE CIÊNCIAS SOCIAIS E TECNOLÓGICAS
MESTRADO EM ENGENHARIA ELETRÔNICA E COMPUTAÇÃO

VICTOR AVILA RATUCHENEI

**Framework iMercur: Explorando
Plataformas de Processamento Contextual
Diracionadas ao Usuário Final na Ciência
de Situação dos Artefatos do SANEP**

Dissertação apresentada como requisito parcial para
a obtenção do grau de Mestre em Engenharia
Eletrônica e Computação

Orientação: Prof. Dr. Adenauer Correa Yamin
Colaboração: Profa. Dra. Fernanda Pinto Mota

Pelotas
2022

Catalogação da publicação
Ficha elaborada a partir de dados fornecidos pelo(a) autor(a)
Bibliotecária da UCPEL: Cristiane de Freitas Chim CRB 10/1233

Ratuchenei, Victor Avila

Framework iMercur: Explorando Plataformas de Processamento Contextual Direcionadas ao Usuário Final na Ciência de Situação dos Artefatos do SANEP. / Victor Avila Ratuchenei. - Pelotas: UCPEL, 2022.

81 f.

Orientador: Adenauer Correa Yamin.

Dissertação (mestrado) – Universidade Católica de Pelotas, Programa de Pós-Graduação, Mestrado em Engenharia Eletrônica e Computação. – Pelotas, BR-RS, 2022.

1. iMercur. 2. Ciência de Contexto e Situação.
3. Análise de dados. 4. Internet das Coisas. I. Yamin,
Adenauer Correa. II. Título.

UNIVERSIDADE CATÓLICA DE PELOTAS

Reitor: Prof. José Carlos Pereira Bachettini Júnior

Pró-Reitor-Acadêmico: Prof. Ezequiel Insaurriaga Megiato

Coordenador de Pesquisa e Pós-Graduação Stricto Sensu: Prof. Ricardo Tavares Pinheiro

Diretora do Centro de Ciências Sociais e Tecnológicas: Profa. Ana Cláudia Vinholes Lucas

Coordenadora do Mestrado em Engenharia Eletrônica e Computação: Profa. Alexandra Lackmann Zimpeck

*“Os grandes navegadores devem sua
reputação aos temporais e às tempestades.”*

— EPICURO —

AGRADECIMENTOS

À Universidade Católica de Pelotas, a qual faz parte da minha trajetória acadêmica.

Ao Serviço Autônomo de Saneamento de Pelotas (SANEP) pelo financiamento concedido durante o período do curso.

Ao Professor Adenauer Correa Yamin, pela orientação, pela atenção, pelos ensinamentos, pelas oportunidades e confiança depositada durante todo o percurso.

À Professora Fernanda Pinto Mota, pelo apoio e orientação no desenvolvimento desta dissertação.

A todos os professores do Mestrado em Engenharia Eletrônica e Computação - PGEEC, que tive o privilégio que conhecer.

A todos os colegas pela parceria durante este período.

Muito obrigado.

RESUMO

Para superação dos desafios inerentes à gestão dos recursos hídricos necessários à qualidade de vida das cidades modernas, se mostram indispensáveis soluções tecnológicas que possam garantir maiores subsídios para a tomada de decisões referentes à administração da infraestrutura de saneamento básico. Neste sentido, o *framework* proposto nesta Dissertação, denominado iMercur, tem como premissa central contribuir com as pesquisas em desenvolvimento no ME-EC/UCPel para atender demandas de gerenciamento dos artefatos do SANEP empregados no saneamento básico. Neste sentido, a pesquisa desenvolvida nesta Dissertação priorizou a utilização de protocolos padrão da Internet para as comunicações, bem como o emprego de soluções *open-source* de software para composição da arquitetura proposta. O iMercur disponibiliza as informações sensoriadas, tanto ao público em geral, quanto para os colaboradores do SANEP, oferecendo uma *interface* concebida para ser de uso intuitivo, o que facilita a manipulação por parte de usuários com diferentes perfis de interesse. Além disso, o iMercur, irá oferecer informações textuais e gráficas para caracterizar o comportamento dos diferentes sensores previstos, de forma independente ou combinada. De modo mais específico, o iMercur tem por objetivo geral contribuir com os serviços de Ciência de Contexto e Situação hoje empregados para inferência do estado em que se encontram os diferentes artefatos utilizados no saneamento básico no município de Pelotas pelo SANEP. Destaca-se como principais contribuições deste trabalho: (i) integrar pelo uso de mensageria diferentes plataformas de computação científica ao serviço de processamento de contexto e situação, (ii) possibilitar a criação de regras pela própria comunidade usuária, bem como (iii) prover a instanciação destas regras junto a um conjunto de sensores. O protótipo para avaliação do iMercur explora um cenário de uso que contempla o emprego do Software Octave, enquanto plataforma para computação científica, com amplo suporte ao processamento numérico. A avaliação dessas contribuições foi realizada por meio do emprego de uma ferramenta que sistematiza as opiniões dos usuários. Para tanto, foi utilizado o Modelo TAM, o qual permitiu observar a facilidade de uso do *framework* e a utilidade percebida. Os resultados atingidos são promissores e apontam para continuidade da pesquisa.

Palavras-chave: iMercur. Ciência de Contexto e Situação. Análise de dados. Internet das Coisas.

iMercur Framework: Exploring End User Contextual Processing Platforms in SANEP

Artifact Situation Awareness

ABSTRACT

In order to overcome the challenges inherent in managing water resources necessary for the quality of life in modern cities, technological solutions that can guarantee more significant subsidies for decision-making regarding the management of basic sanitation infrastructure are indispensable. In this sense, the framework proposed in this Dissertation, called iMercur, has as its central premise to contribute to research under development at MEEC/UCPel to meet the demands of managing SANEP artifacts used in basic sanitation. In this sense, the research developed in this Dissertation prioritized using standard Internet protocols for communications, as well as the use of open-source software solutions to compose the proposed architecture. iMercur makes the sensed information available to both the general public and SANEP collaborators, offering an interface designed to be intuitive to use, which facilitates manipulation by users with different interest profiles. In addition, iMercur will offer textual and graphical information to characterize the different sensors' behavior, independently or in combination. More specifically, the general objective of iMercur is to contribute to the Context and Situation Awareness services currently used to infer the status of the different artifacts used in basic sanitation in the municipality of Pelotas by SANEP. The following stand out as the main contributions of this work: (i) to integrate, through the use of messaging, different scientific computing platforms to the context and situation processing service, (ii) to enable the creation of rules by the user community itself, as well as (iii) to provide the instantiation of these rules with a set of sensors. The iMercur evaluation prototype explores a usage scenario that includes the use of Octave Software, as a platform for scientific computing, with broad support for numerical processing. The evaluation of these contributions was carried out using a tool that systematizes users' opinions. For that, the TAM Model was used, which allowed observing the ease of use of the framework and the perceived usefulness. The achieved results are promising and point to the continuation of the research.

Keywords: iMercur. Context and Situation Awareness. Data analysis. Internet of Things.

LISTA DE ABREVIATURAS E SIGLAS

ABM	<i>Agent-Based Model</i>
AJAX	<i>Asynchronous JavaScript and XML</i>
ANN	<i>Artificial Neural Network</i>
API	<i>Application Programming Interface</i>
CEEMDAN	<i>Complete Ensemble Empirical Mode Decomposition with Adaptive Noise</i>
CLI	<i>Command Line Interface</i>
ECA	Evento-Condição-Ação
EXEHDA	<i>Execution Environment for Highly Distributed Applications</i>
G3PD	Grupo de Pesquisa em Processamento Paralelo e Distribuído
GRU	<i>Gated Recurrent Unit</i>
HTTP	<i>Hypertext Markup Language</i>
IBM	<i>International Business Machines</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IoT	<i>Internet of Things</i>
IP	<i>Internet Protocol</i>
IPSO	<i>Internet Protocol for Smart Object</i>
ITU	<i>International Telecommunications Union</i>
JNMM	<i>Joint Non-homogeneous Markov Model</i>
JSON	<i>JavaScript Object Notation</i>
MIT	<i>Massachusetts Institute of Technology</i>
MM	<i>Homogeneous Markov Model</i>
MOM	<i>Middleware Orientado a Mensagens</i>
MQTT	<i>Message Queuing Telemetry Transport</i>
NFC	<i>Near Field Communication</i>

PEOU	<i>Perceived Ease of Use</i>
PHP	<i>Hypertext Preprocessor</i>
PU	<i>Perceived Usefulness</i>
REST	<i>Representational State Transfer</i>
RFID	<i>Radio Frequency Identification</i>
RIC	Repositório de Informações Contextuais
RIGIM	Repositório de Informações Gerenciais do iMercur
RMS	<i>Root Mean Squared</i>
RMSE	<i>Root Mean Squared Error</i>
SANEP	Serviço Autônomo de Saneamento de Pelotas
SCADA	<i>Supervisory Control And Data Acquisition</i>
SD	<i>System dynamics</i>
SGBD	Sistema Gerenciador de Banco de Dados
SGML	<i>Standard Generalized Markup Language</i>
SIM	<i>State Independent Model</i>
SQL	<i>Structured Query Language</i>
SVR	<i>Support Vector Regression</i>
TAM	<i>Technology Acceptance Model</i>
TCP	<i>Transmission Control Protocol</i>
TI	Tecnologia da Informação
TO	Tecnologia Operacional
UID	<i>User IDentifier</i>
WISP	<i>Wireless Identification and Sensing Platform</i>
WWW	<i>World Wide Web</i>
XML	<i>eXtensible Markup Language</i>

LISTA DE FIGURAS

Figura 2.1 Domínio de Aplicações da IoT	17
Figura 2.2 Ciclo de Vida para Ciência de Contexto.....	19
Figura 2.3 Níveis de Abstração das Informações Contextuais.....	24
Figura 2.4 Visão Geral dos Sistemas de Mensageria	26
Figura 3.1 Visão da Proposta do Trabalho TR1	29
Figura 3.2 Visão da Proposta do Trabalho TR2	30
Figura 3.3 Visão da Proposta do Trabalho TR3	31
Figura 3.4 Visão da Proposta do Trabalho TR4	33
Figura 3.5 Visão da Proposta do Trabalho TR5	34
Figura 4.1 Arquitetura de Software do <i>Middleware</i> EXEHDA	39
Figura 4.2 Ambiente na IoT Provido pelo <i>Middleware</i> EXEHDA	41
Figura 5.1 <i>Framework</i> iMercur na Arquitetura de Software Concebida pelo G3PD para o SANEP.....	43
Figura 5.2 Visão Geral do <i>Framework</i> iMercur	44
Figura 5.3 Visão Geral das Unidades do <i>Framework</i> iMercur.....	45
Figura 5.4 Unidade de Inferência de Situação	47
Figura 5.5 JSON de Criação de uma Regra	48
Figura 5.6 JSON de Atualização de uma Regra.....	48
Figura 5.7 API do iMercur para Trocas de Dados	49
Figura 5.8 Diagrama de Sequência da Unidade de Inferência e Situação	50
Figura 5.9 Tela de Cadastro de Regras.....	51
Figura 5.10 Tela de Listagem de Regras	52
Figura 5.11 Tela de Execução de Regras	52
Figura 5.12 Tela de Execução de Regras com Gráfico	53
Figura 5.13 Unidade de Mensageria	54
Figura 5.14 Tela de Listagem de Canais	55
Figura 5.15 Tela de Criação de um Canal	56
Figura 5.16 Diagrama dos Módulos da Unidade de Processamento Científico	57
Figura 5.17 Diagrama de Sequência da Unidade de Processamento Científico	58
Figura 5.18 Exploração da Unidade de Processamento Científico	59
Figura 5.19 Legenda de Cores do Modelo C4	61
Figura 5.20 JSON de Configuração de um Conector.....	67
Figura 5.21 JSON de Configuração de um Fluxo de Dados	67
Figura 6.1 Visão Geral do Modelo TAM	70
Figura 6.2 Análise Descritiva das Respostas do Instrumento de Avaliação TAM	74

LISTA DE TABELAS

Tabela 3.1 Comparação dos Artigos Relacionados.....	36
Tabela 5.1 Métodos da API REST para a Configuração de Regras	48
Tabela 5.2 Métodos da API REST para a Configuração dos Canais	54
Tabela 5.3 Métodos da API REST para a Configuração dos Streams de Dados	66
Tabela 6.1 Valores Numéricos para a Escala Likert.....	72
Tabela 6.2 Afirmativas do Instrumento de Avaliação TAM	72

SUMÁRIO

1 INTRODUÇÃO	12
1.1 Objetivos	13
1.2 Estrutura do Texto	14
2 ESCOPO DO TRABALHO	15
2.1 Internet das Coisas.....	15
2.2 Ciência de Contexto e Situação.....	18
2.2.1 Ciência de Contexto	18
2.2.2 Ciência de Situação.....	22
2.3 Sistemas de Mensageria.....	25
2.4 Considerações Finais do Capítulo	27
3 TRABALHOS RELACIONADOS	28
3.1 Descrição dos Trabalhos Relacionados	28
3.2 TR1 - Urban Water Demand: Statistical Optimization Approach to Modeling Daily Demand	28
3.3 TR2 - An Innovative Hourly Water Demand Forecasting Preprocessing Framework with Local Outlier Correction and Adaptive Decomposition Techniques	30
3.4 TR3 - Dynamic Behavioral Modeling, Simulation and Analysis of Household Water Consumption in an Urban Area: a Hybrid Approach.....	31
3.5 TR4 - Predictive Classification of Water Consumption Time Series using Non-homogeneous Markov Models	32
3.6 TR5 - Big Data Analytics and IoT in Operation Safety Management in Under Water	33
3.7 Considerações Finais do Capítulo	35
4 MIDDLEWARE EXEHDA: ARQUITETURA E FUNCIONALIDADES	38
4.1 Aspectos Funcionais e Arquiteturais.....	38
4.2 Ambiente Ubíquo na IoT provido pelo <i>middleware</i> EXEHDA	40
4.3 Considerações Finais do Capítulo	42
5 IMERCUR: CONCEPÇÃO E TECNOLOGIAS.....	43
5.1 Visão geral	43
5.2 Arquitetura de Software Proposta	44
5.3 Unidade de Atendimento ao Usuário	46
5.4 Unidade de Inferência de Situação.....	46
5.5 Unidade de Mensageria	53
5.6 Unidade de Processamento Científico	56
5.7 Tecnologias Selecionadas Para o Desenvolvimento.....	58
5.8 Considerações Finais do Capítulo	68
6 IMERCUR: AVALIAÇÃO	69
6.1 Considerações Finais do Capítulo	74
7 CONSIDERAÇÕES FINAIS	75
7.1 Principais Conclusões	75
7.2 Publicações.....	76
7.3 Trabalhos Futuros.....	77
REFERÊNCIAS	78

1 INTRODUÇÃO

Os recursos hídricos historicamente garantiram às cidades, estrategicamente assentadas de modo próximo a leitos de fontes de água potável, os meios necessários para o seu sustento, suprindo o abastecimento de água para população, proporcionando o desenvolvimento da agricultura, a criação de animais e a navegação.

Os primeiros registros da utilização de métodos de saneamento datam de milhares de anos, onde as civilizações já evidenciavam sua importância na saúde pública. Com a chegada da Revolução Industrial, as cidades tornaram-se superpovoadas e diversos problemas surgiram pela falta de saneamento, entre os quais, doenças e epidemias que chegaram a dizimar mais da metade da população do continente europeu. Somente em 1843 as atenções se voltaram para as condições sanitárias e o desenvolvimento de estudos na área de saneamento básico e saúde social (NUNES; DIAZ, 2020).

No Brasil particularmente, com o avanço das epidemias trazidas da Europa, em 1894 surgiu o primeiro código sanitário do estado de São Paulo (MIRANZI et al., 2010). Mais tarde, no início do século XX, o engenheiro Saturnino de Brito, considerado o pai da Engenharia Sanitária e Ambiental no Brasil, foi responsável por várias obras para a construção de sistemas de distribuição de águas e coleta de esgotos em várias capitais (RIBEIRO; ROOKE, 2010).

A partir do desenvolvimento das cidades, do crescimento populacional e das mudanças dos padrões de consumo humano nos últimos séculos, a água vem sofrendo um revés em seu ciclo natural causado pela ação humana. A poluição atmosférica e a contínua derrubada de florestas nativas, dando lugar a extensas áreas voltadas à pecuária e a agricultura, ampliam a problemática hídrica para além do abastecimento da população, atingindo fortemente a geração de energia elétrica a nível de nação, especificamente o Brasil, onde a maior parte de sua matriz energética está baseada em usinas hidrelétricas (ANEEL, 2021).

Para garantir a qualidade de vida e o desenvolvimento social e econômico da população, os desafios inerentes a gestão dos recursos hídricos de uma cidade como Pelotas, se mostram necessárias implementações tecnológicas para gerenciamento da aquisição e tratamento de informações, garantindo maior exatidão para a tomada de decisões, trazendo formas mais eficientes e criteriosas para dar suporte à gestão da infraestrutura de saneamento básico.

Dentro desse contexto, a Internet das Coisas (IoT) se mostra oportuna para o acompanhamento da situação dos artefatos empregados no saneamento básico, promovendo a integração do mundo físico ao digital, de maneira a criar uma rede na qual a captura e o processamento de informações possam acontecer de forma autônoma e regida por regras, provendo uma visão

da situação em que se encontram os diferentes artefatos envolvidos (PERERA, 2017).

Portanto, torna-se interessante que as informações previamente armazenadas em plataformas que registram dados contextuais possam interagir com plataformas de computação científica para o processamento de contextos complexos. Com esta motivação, é concebido o *framework* iMercur para promover a integração destas plataformas de diferentes naturezas, possibilitando a extração de informações a partir de um processamento científico explorando computação numérica, permitindo, inclusive, que esse processamento possa ser realizado por infraestruturas de computação distribuídas, com as plataformas a serem integradas estando alojadas em diferentes equipamentos.

O iMercur está sendo concebido sob a perspectiva do Convênio UCPel-SANEP. Este convênio busca atender às necessidades do Serviço Autônomo de Saneamento de Pelotas (SANEP), explorando o emprego de soluções tecnológicas que agreguem recursos de hardware-/software, desenvolvidas pelos cursos da Área de Tecnologia da UCPel.

Atualmente o SANEP é responsável pela captação, tratamento e distribuição de água potável, coleta, tratamento e destinação de resíduos sólidos, coleta e tratamento de esgotos sanitários e pela macrodrenagem urbana. Possui aproximadamente 800 funcionários que atuam nos diversos departamentos, divisões e setores da autarquia.

1.1 Objetivos

Considerando isto, o *Framework* iMercur tem por objetivo geral contribuir com os serviços de Ciência de Contexto e Situação hoje empregados no SANEP para inferência do estado em que se encontram os diferentes artefatos utilizados no saneamento básico da cidade de Pelotas.

Considerando este objetivo geral foram elencados alguns objetivos específicos, os quais estão descritos a seguir:

- conceber o iMercur, tendo como premissa que a sua manipulação possa ser feita diretamente pelo usuário final, em particular os integrantes do corpo técnico-administrativa do SANEP;
- disponibilizar facilidades *on-line* para criação e edição de regras de processamento contextual a serem empregadas pelas plataformas de computação científica gerenciadas pelo iMercur;

- facultar ao usuário final o emprego de processadores contextuais alojados em diferentes plataformas de hardware, considerando a natureza das análises a serem feitas sobre os dados sensoriados;
- permitir ao usuário final a associação das regras de processamento contextual criadas aos vários sensores disponíveis;
- prover suporte para o gerenciamento da ativação de regras no iMercur a partir da publicação dos dados sensoriados e/ou por deliberação dos usuários;
- suportar as premissas do *Middleware EXEHDA* para uma operação distribuída em larga escala, tanto para o sensoriamento como para a atuação.

Tendo como base estes objetivos, será desenvolvido um modelo arquitetural para o *framework* iMercur que possa atender as demandas previstas de configurações, mensageria e processamento de contextos complexos, considerando uma integração com a base de software em desenvolvimento no MEEC para atendimento do Convênio UCPel-SANEP.

1.2 Estrutura do Texto

Após este capítulo de introdução onde as principais motivações e objetivos do trabalho foram apresentados, o texto desta Dissertação está estruturado em mais cinco capítulos, descritos a seguir.

No capítulo 2, o escopo do trabalho aborda os tópicos de interesse na área de Internet das Coisas, Ciência de Contexto e Situação e também é apresentado os conceitos e sistemas de Mensageria.

No capítulo 3, a revisão de literatura é realizada acerca dos trabalhos relacionados, destacando tendências e lacunas de pesquisa.

O *middleware* EXEHDA e os seus principais componentes são descritos no capítulo 4.

No capítulo 5 é apresentada a proposta do *framework* iMercur, bem como são apresentadas as tecnologias utilizadas na abordagem.

O modelo de avaliação é abordado no capítulo 6 juntamente com os resultados.

Por fim, no capítulo 7 são apresentadas as principais conclusões, publicações e trabalhos futuros.

2 ESCOPO DO TRABALHO

Neste capítulo serão introduzidas as áreas de estudo e pesquisa que embasam esta Dissertação. Na seção 2.1 é abordada a origem da IoT e sua evolução, encabeçada pelas aplicações que deram impulso ao seu desenvolvimento. Finalizando esta seção é feita a caracterização da IoT a partir de diferentes visões. Na seção 2.2 serão introduzidos os conceitos de Ciência de Contexto, suas principais funções, aplicações e desafios. Também é caracterizada como ocorre a aquisição de contextos, as responsabilidades e a forma de coleta. A Ciência de Contexto e Situação é abordada em suas características e abstrações. Por fim, a seção 2.3 revisa os principais conceitos sobre mensageria.

2.1 Internet das Coisas

O termo “Internet das Coisas” (do inglês, *Internet of Things – IoT*) foi cunhado pela primeira vez por Kevin Ashton em uma apresentação feita na *Procter and Gamble* em 1999, vinculando a nova ideia de RFID (identificação por radiofrequência) na cadeia de suprimentos da *Procter and Gamble*, uma vez que a Internet era mais do que apenas uma boa forma de atrair a atenção dos executivos. Ele mencionou: “A Internet das Coisas tem o potencial de mudar o mundo, assim como a Internet fez”. Logo, o MIT Auto-ID Center caracterizou a IoT em 2001. Posteriormente, a IoT foi formalmente apresentada pelo Relatório de Internet da União Internacional de Telecomunicações (ITU) em 2005.

A IoT representa o futuro da computação e das comunicações ao possibilitar a integração de diversos dispositivos tecnológicos ao cotidiano das pessoas. O seu efeito é significativo em todos os aspectos da vida, como por exemplo: na saúde ao possibilitar sensoriamento dos pacientes, na agricultura ao facilitar a operação de máquinas agrícolas de forma automatizada, na Educação ao fornecer dispositivos para ensino a distância, entre outros. Sendo assim, a principal característica da IoT é fornecer a comunicação entre bilhões de dispositivos, a qualquer hora e momento (RAYES; SAMER, 2016).

Marques, Garcia e Pombo (2017) definem que na visão da IoT há várias “coisas” que incluem não apenas dispositivos de comunicação, mas também todos os outros objetos físicos do planeta passíveis de serem conectados e controlados por meio da Internet. O conceito de IoT atraiu significativamente a atenção de muitos pesquisadores nos últimos anos, com implementações nas áreas da saúde, agrícola, aeroespacial, por exemplo, sendo essencial ao modelo de cidades inteligentes e domótica. Dessa forma, as incessantes melhorias científicas possibi-

litam a construção de dispositivos inteligentes com enorme potencial de detecção e conexão, permitindo diversos aprimoramentos baseados no paradigma da IoT.

Segundo Atzori, Iera e Morabito (2010), a principal ideia associada à IoT é o alto impacto que ela tem no comportamento de usuários. Do ponto de vista de um usuário privado, os efeitos mais óbvios da introdução da IoT são visíveis nos campos de trabalho e doméstico. Nesse contexto, domótica, vida assistida, *e-health*, aprendizagem aprimorada são apenas alguns exemplos de possíveis cenários de aplicação em que a IoT tem um papel preponderante. Da mesma forma, da perspectiva dos usuários de negócios, as consequências mais aparentes são igualmente visíveis em áreas como automação e manufatura industrial, logística, gestão de negócios e processos, transporte inteligente de pessoas e mercadorias, entre outros.

Atualmente, a IoT é alimentada pelo incremento de tecnologias, incluindo a convergência entre Tecnologia da Informação (TI) e Tecnologia Operacional (TO), a introdução de negócios baseados na Internet, o aumento da quantidade de dispositivos móveis inteligentes e de aplicativos de redes sociais, a transformação digital massiva, as interfaces de usuário que permitem que as pessoas se comuniquem por um simples toque, comando de voz ou mesmo um comando de observação, a adoção de tecnologia mais rápida, a maior demanda por aplicativos e soluções de segurança. Frente a expansão geral da tecnologia, a segurança da IoT é vista como um desafio e uma oportunidade de negócios ao mesmo tempo com áreas que abrangem a preservação dos dados em repouso, a proteção do transporte dos dados, a proteção de APIs/interfaces entre sistemas e várias fontes de dados e, claro, o controle de sensores e atuadores (MARQUES; GARCIA; POMBO, 2017).

Neste cenário tecnológico onde a IoT é protagonista, podemos observar o resultado da fusão de diferentes visões que caracterizam o modelo da IoT: visão orientada para as coisas, visão orientada para a Internet e visão orientada para a semântica. Envolvidas pela ideia básica da presença pervasiva de uma variedade de objetos com capacidade de interação e cooperação entre eles para atingir um objetivo comum. A visão orientada para as coisas foca em dispositivos autônomos inteligentes que usam tecnologias como objetos NFC e RFID. A visão orientada para a Internet centra-se na ideia de manter os dispositivos ligados à rede, tendo um endereço único e utilizando protocolos padrão. A visão orientada para a semântica foca no armazenamento, busca e organização das informações geradas pela IoT, procurando criar soluções de modelagem de arquiteturas de dados e ambientes para lidar de forma eficiente com as informações produzidas.

Orientada por estas três visões, o trabalho de Hejazi et al. (2018) apresenta a IoT como uma rede de coisas circundantes que se conectam à Internet, como vários sensores, veículos e dispositivos que podem ser monitorados, detectados e controlados. As coisas são incorporadas

aos sensores para sentir o ambiente e se comunicar com outras coisas. O ambiente é monitorado e as coisas podem ser detectadas, para serem identificadas exclusivamente e para executar qualquer ação predefinida. Desse modo, os usuários podem ser notificados, acessar as coisas pela internet e tomar medidas para controlar o ambiente.

Segundo Al-Fuqaha et al. (2015), a IoT permite que objetos físicos vejam, ouçam, processem e realizem trabalhos, fazendo com que eles “conversem”, compartilhem informações e coordenem decisões. A IoT transforma esses objetos tradicionais em inteligentes, explorando suas tecnologias associadas, como computação ubíqua e pervasiva, dispositivos incorporados, tecnologias de comunicação, redes de sensores, protocolos e aplicativos da Internet. Objetos inteligentes, juntamente com suas tarefas, constituem aplicações específicas de domínio (mercados verticais), enquanto a computação ubíqua e os serviços analíticos formam serviços independentes do domínio de aplicação (mercados horizontais). A Figura 2.1 ilustra a visão geral da IoT, em que cada aplicação de um domínio específico está interagindo com serviços independentes de domínio, enquanto em cada domínio os sensores e atuadores se comunicam diretamente uns com os outros.



Figura 2.1 – Domínio de Aplicações da IoT
Fonte: Adaptada de (AL-FUQAHÁ et al., 2015)

A indústria emergente da Internet das Coisas está rapidamente encontrando seu caminho no campo da ciência e tecnologia, economia, negócios e vida social, visando melhorar a

qualidade de vida em geral. Conectando coisas, buscando dados, analisando e respondendo ao ambiente; essas qualidades tornam a IoT uma parte da vida humana (ROUTH; PAL, 2018). Com todos esses atributos, a IoT fortalece a integração entre os sistemas computacionais e o ambiente físico, com potencial para produzir informações contextuais em massa (QIU et al., 2018; PERERA et al., 2014).

2.2 Ciência de Contexto e Situação

Esta seção discorre sobre os conceitos de Ciência de Contexto e Situação, trazendo as análises apresentadas por diferentes autores, priorizando aquelas abordagens que apresentam melhor alinhamento com os objetivos deste trabalho.

2.2.1 Ciência de Contexto

A Ciência de Contexto é a capacidade que um sistema computacional tem de interpretar as informações pertinentes do meio, assim como responder aos estímulos interagindo com o meio, quando possível (TEMDEE; PRASAD, 2018).

Embora as primeiras definições de contexto começaram a ser apresentadas na década de 90, várias definições têm sido propostas e discutidas. Uma das definições mais aceitas e utilizadas por pesquisadores da área é a encontrada no artigo seminal de Dey (2001). Segundo o autor, entende-se por contexto “Qualquer informação que possa ser utilizada para caracterizar a situação de entidades (pessoa, lugar ou objeto) que sejam consideradas relevantes para a interação entre um usuário e uma aplicação, incluindo o usuário e a aplicação”.

Para Lopes et al. (2014), contexto é definido como toda informação que pode ser obtida computacionalmente com elevada importância para a aplicação. Nesse sentido, o contexto pode elencar aspectos de acordo com sua relevância, dependendo das vantagens envolvidas para a aplicação. Enquanto Ciência de Contexto é a capacidade de um sistema em usar o contexto para fornecer serviços e/ou informações de interesse (TEMDEE; PRASAD, 2018).

As funções pertinentes à Ciência de Contexto são a capacidade de coletar informações para o contexto de interesse e armazenar os dados coletados tornando possível produzir informações de contexto, além de realizar ações a partir de regras de processamento contextual, realizando atuações especificadas por usuários (MACHADO et al., 2019), possibilitando aos sistemas cientes de contexto adquirir contexto, raciocinar sobre ele e alterar o comportamento

do sistema para a situação de mudança do usuário. Estes sistemas devem ser flexíveis, adaptativos e capazes de atuar automaticamente para ajudar o usuário na realização de suas atividades.

Por regra geral, dentro das diversas propostas de *middleware* clientes de contexto, a vida de um contexto tem duração de um ponto inicial, que é sua obtenção, até sua destruição, finalizando seu ciclo de vida. Durante este ciclo do contexto, existem seis fases bem definidas, citam-se: aquisição, modelagem, raciocínio, distribuição, repositório e visualização (LI et al., 2015). O fluxograma do ciclo de vida do contexto está representado na Figura 2.2.

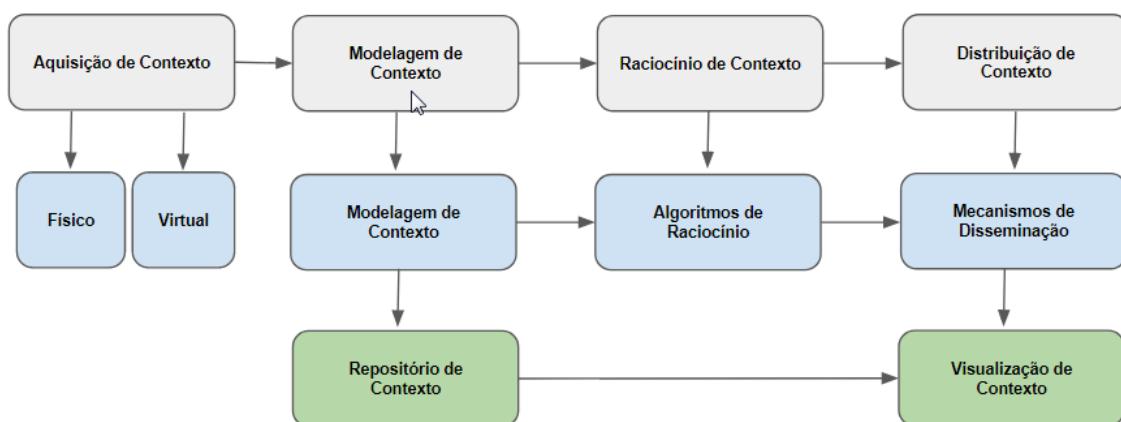


Figura 2.2 – Ciclo de Vida para Ciência de Contexto

Fonte: Adaptada de (LI et al., 2015)

A maneira de realizar Ciência de Contexto começa com a aquisição de vários tipos de dados sensoriados, seguida pelo processo de formalização e inferência e, finalmente, termina com a distribuição de contextos úteis para as aplicações. No estágio de modelagem e raciocínio de contexto, os dados de contextos históricos precisam ser registrados para uso ou consultas adicionais e também podem ser visualizados pelos usuários.

Atualmente, a Ciência de Contexto tem sido definida como a capacidade da infraestrutura computacional em fornecer informações relevantes para as aplicações ajustarem seu comportamento, a partir dos dados contextuais coletados. Desta forma, a Ciência de Contexto caracteriza-se por possuir duas grandes frentes: a aquisição e o tratamento dos dados que expressam informações relevantes sobre o contexto; e o ajuste do comportamento das aplicações frente às alterações de contexto (KNAPPMEYER et al., 2013).

A aquisição diz respeito a forma de obtenção das informações contextuais, a qual pode ocorrer de três formas:

- sensoriada, a informação é adquirida por meio de sensores, por exemplo, a temperatura, nível de ruído, localização;

- derivada, onde a informação é obtida através do tratamento dos dados brutos, em tempo de execução da aplicação, por exemplo, a média da temperatura em um determinado ambiente;
- provida, a informação é explicitamente disponibilizada à aplicação, por exemplo, fornecer os dados de usuário por meio de um formulário.

O processo de aquisição de contexto pode tornar-se complexo devido à grande variedade de sensores que podem estar envolvidos, bem como à natureza dinâmica da informação contextual. Em função do modo como os dados são capturados, os sensores podem ser classificados em três grupos (ALEGRE; AUGUSTO; CLARK, 2016):

- sensores físicos, correspondem aos sensores de hardware, os quais são capazes de capturar praticamente qualquer dado físico, por exemplo: sensores de temperatura, sensores de fumaça, câmeras, microfones, sistema de posicionamento global (GPS);
- sensores virtuais, nesse grupo, a origem das informações de contexto é um software. Isso significa que é possível determinar, por exemplo, a localização de uma pessoa não somente através do uso de sistemas de localização com sensores físicos, mas também através de sensores virtuais que geram informações de localização, tais como: sistemas de agenda, reservas de viagens, correio eletrônico, mensagens instantâneas;
- sensores lógicos, esses sensores fazem uso de um conjunto de fontes de informação, combinando sensores físicos e virtuais com informação adicional obtida em bases de dados. Por exemplo, um sensor lógico pode ser construído para detectar a posição atual de uma pessoa através da análise dos logins em microcomputadores e de um banco de dados mapeando os dispositivos fixos.

Desta forma, a camada de aquisição visa abstrair das aplicações a complexidade da coleta de dados, além de possibilitar a reutilização de sensores e a separação entre obtenção e utilização das informações de contexto (ALEGRE; AUGUSTO; CLARK, 2016).

Devido ao aumento drástico do uso de dispositivos de IoT, devem ser adotadas estratégias que promovam a escalabilidade e otimização da rede. Com esta finalidade, são exploradas técnicas de coleta de dados, as quais são classificadas quanto à responsabilidade e a frequência (PERERA et al., 2014).

Quanto à responsabilidade, o processo de coleta das informações de contexto pode ocorrer pelo método *Pull*, onde o procedimento de coleta da informação contextual é disparado a

partir do *middleware*, o qual faz uma consulta diretamente ao sensor. Nessa estratégia de coleta, o software que gerencia o comportamento do sensor não precisa realizar computações elaboradas, pois não cabe a ele avaliar o momento oportuno de realizar a coleta. Pelo método *Push*, o procedimento de coleta é disparado a partir do sensor, o qual tem a iniciativa de enviar os dados capturados para o *middleware* através de um processo de publicação. Essa estratégia proporciona um processo de aquisição de maneira reativa, podendo ser disparada através de eventos do ambiente.

Quanto à frequência de aquisição, o processo de coleta das informações contextuais pode ocorrer de maneira instantânea ou periódica. Na coleta instantânea, a coleta é disparada através de eventos do ambiente e acontece instantaneamente. Na coleta periódica é obedecido a períodos especificados de acordo com as necessidades da aplicação do usuário.

As duas frequências de coleta, instantânea e periódica, podem ser implantadas tanto através do método *Pull* quanto do método *Push*. A coleta de forma instantânea associada ao método *Push* promove uma reação rápida e mostra-se adequada em eventos críticos onde é necessária uma tomada de decisão rápida.

É fundamental que a ação de aquisição de contexto permaneça em constante execução, sendo esperado que seja executada de forma independente das aplicações que a utilizem, viabilizando, desta forma, que diversas aplicações possam fazer uso das mesmas informações contextuais.

O processamento do contexto abrange aspectos relacionados à interpretação, agregação, armazenamento, consulta e inferência das informações contextuais obtidas por uma etapa de aquisição, considerando o modelo do contexto empregado. O propósito central do processamento do contexto é viabilizar a compreensão dos contextos de interesse das aplicações, apoiando o processo de identificação de situações, e a consequente tomada de decisões para ajustes no comportamento das aplicações (BIBRI, 2015).

Este processo de interpretação de contexto pode ser direto, como derivar o nome de uma rua a partir de suas coordenadas geográficas, ou complexo e oneroso, como inferir o humor de um usuário baseado em seu perfil e na atividade em que ele está realizando. Além disso, o ambiente da IoT é extremamente dinâmico e as informações contextuais podem estar distribuídas em diferentes lugares ou produzidas por dispositivos com alto grau de mobilidade. Essa complexidade faz com que exista a necessidade de um suporte computacional às aplicações, de maneira a auxiliá-las na realização de interpretações de contextos (BIBRI, 2015; ALEGRE; AUGUSTO; CLARK, 2016).

Assim, as atividades de processamento do contexto devem ser abstraídas das aplicações

e um módulo interpretador torna-se, portanto, um componente essencial em uma plataforma de suporte a tais aplicações. Ele deve ser capaz de obter e prover informações contextuais em diferentes níveis de abstração, conforme a necessidade do usuário e de suas aplicações. Uma aplicação pode necessitar tanto de informações brutas, de mais baixo nível, como de informações mais abstratas e elaboradas, de mais alto nível, provenientes de um processo de refinamento e interpretação (GANDODHAR; CHaware, 2018).

A necessidade de manter o histórico de informações de contexto é um requisito ligado à sua aquisição de informações, bem como à disponibilidade contínua dos componentes de captura de informações de contexto. Um histórico de contexto pode ser utilizado para estabelecer tendências e predizer valores futuros de informações contextuais. Sem o armazenamento dessas informações, análises desse tipo não são possíveis de realizar.

A distribuição de contexto é responsável por disseminar informações de contexto úteis para aplicativos correspondentes. Dois mecanismos típicos de distribuição, *subscribe/publish* e *polling*, são amplamente utilizados em soluções atuais (LAZIDIS; TSAKOS; PETRAKIS, 2022). *Publish-Subscribe* é também chamado de Notificação. Aplicativos interessados em determinadas informações de contexto podem se inscrever no *middleware* e ser notificados quando ocorrerem atualizações das informações de contexto registradas. As aplicações consumidoras de contexto podem fazer consultas às suas informações contextuais que estão interessadas, a qualquer momento. Dependendo das técnicas de modelagem e raciocínio usadas, diferentes métodos de consulta podem ser empregados.

2.2.2 Ciência de Situação

Uma situação consiste na interpretação de um conjunto de elementos contextuais instanciados relacionando cada um de forma a prover alguma informação válida em um intervalo de tempo específico.

O avanço das tecnologias de sensoriamento possibilitou progressos significativos na concepção de sensores de tamanhos menores, mais leves, com menor custo e maior autonomia de suas baterias. Estes sensores coletam dados no ambiente, permitindo que o sistema computacional possa fornecer serviços personalizados, de acordo com o contexto de interesse das aplicações. Esses dados podem ser referentes a informações do ambiente como, por exemplo, temperatura, luminosidade e umidade, bem como do usuário, por exemplo, localização, velocidade de deslocamento e sinais vitais (KRUMM et al., 2010).

O interesse na Ciência de Situação cresceu rapidamente desde o seu uso inicial, na

aviação, para muitas áreas diferentes, tais como o controle de tráfego aéreo, operações militares, transporte, sistemas de energia, aplicação da lei, gestão de emergência, cuidados de saúde, transporte, mineração e operações com petróleo e gás (ENDSLEY, 2015).

Vários modelos importantes de Ciência de Situação foram introduzidos ao longo dos anos (Adams, Tenney and Pew (2017); Durso and Gronlund (1999); Smith and Hancock (1995)) com muitas semelhanças em termos de seus focos sobre a importância dos objetivos, estruturas de memória, modelos mentais e atenção. Contudo, o modelo de Ciência de Situação de Endsley (2015) tem sido considerado um dos mais citados, sendo baseado em três níveis, como apresentado a seguir:

- percepção, é o primeiro passo para a compreensão do ambiente dinâmico. Perceber o estado, a dinâmica e os atributos relevantes deste ambiente torna-se fundamental;
- compreensão, compreender a situação baseia-se numa síntese dos elementos identificados no nível 1, através de reconhecimento de padrões, interpretação e avaliação;
- projeção, projetar o estado futuro, preocupado com o que acontecerá com base na dinâmica dos elementos. Isto é alcançado através do conhecimento do estado e da dinâmica dos elementos e da compreensão da situação dos níveis 1 e 2.

A primeira fase, atua na detecção e reconhecimento de atributos e estados de elementos relevantes do ambiente. Na segunda fase, compreensão sobre a situação, tem-se a síntese dos elementos percebidos pela primeira. Porém, essa fase vai além da simples aquisição da ciência sobre os elementos, nela busca-se atingir o entendimento sobre o significado dos elementos em relação a situação. E a terceira fase baseia-se na habilidade de projetar ações e prever estados futuros dos elementos do ambiente. Tal habilidade é adquirida através da combinação da primeira e segunda fases. Essas projeções e predições são recursos valiosos nas tomadas de decisões (CLOSS, 2017).

A camada de percepção é o nível 1, a compreensão é o nível 2 e a projeção é o nível 3. A percepção recebe informações do mundo real como um conjunto de palavras e repassa para a compreensão, que irá juntá-las, formando uma mensagem e entregará para a projeção, que buscará ações para a mensagem, situações desejadas, retornando a pesquisa para o mundo. É importante notar que a comunicação não ocorre de forma linear, ou seja, caso haja algum problema no nível 3, ele pode retornar direto para o nível 1 (LEE; KIRLIK; DAINOFF, 2013).

Dessa forma, quanto maior o número de dados captados, mais precisas e completas serão as informações de situação (KARAMAN; YALIMAN; OTO, 2017).

Também de forma vaga, a existência de dados de contexto vagos levam a uma modelagem e interpretação da situação. Entende-se por Ciência de Situação a percepção de uma ou mais situações com relação ao tempo ou espaço, a compreensão do seu significado e a projeção do seu estado depois da mudança de alguma variável, como um evento pré-determinado, por exemplo. A noção de situação é usada como um conceito de alto nível para representação de estado (KARCHOUD et al., 2017).

São chamadas de aplicações conscientes de situação, as aplicações que se utilizam de técnicas de sensibilidade à situação. Nestas aplicações, interpretações semânticas externas de contexto de baixo nível permitem uma especificação de alto nível do comportamento humano e da sua interação com o sistema.

Dessa forma é possível representar o estado de abstração da informação em níveis de uma pirâmide em camadas, conforme ilustra a Figura 2.3.

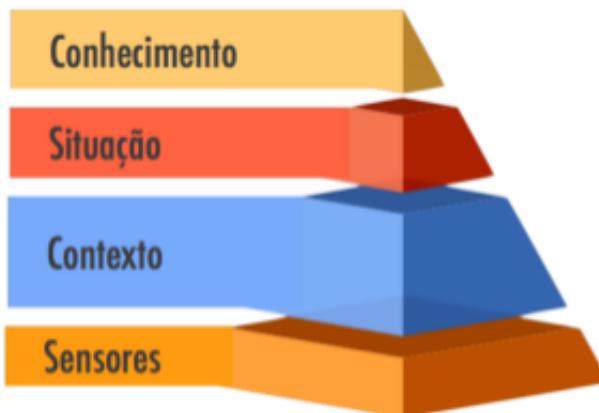


Figura 2.3 – Níveis de Abstração das Informações Contextuais

Fonte: Adaptada de Almeida et al. (2013)

Dados coletados por sensores representam a informação de maior quantidade e menor nível, esses dados brutos são transformados em informações contextuais, de forma a elevar o seu nível. Através desses dados contextualizados são identificadas situações que representam a informação em seu nível máximo, fornecendo assim conhecimento sobre o que está acontecendo para posterior tomada de decisão e ação em um sistema consciente de situação. Na pirâmide é caracterizado que as informações de contexto de baixo nível são semanticamente interpretadas por camadas de contexto de mais alto nível (situação), que geram conhecimento acerca do sistema.

2.3 Sistemas de Mensageria

Mensageria é uma abordagem que permite a comunicação *program-to-program*, de alta velocidade, assíncrona e com entrega confiável. Os programas se comunicam enviando pacotes de dados chamados mensagens uns para os outros. Os canais, também conhecidos como filas, são caminhos lógicos que conectam os programas e transmitem mensagens. Um canal se comporta como uma coleção ou matriz de mensagens, mas que é compartilhado entre vários computadores e pode ser usado simultaneamente por vários aplicativos. Um remetente ou produtor é um programa que envia uma mensagem gravando a mensagem em um canal. Um receptor ou consumidor é um programa que recebe uma mensagem lendo de um canal, marcando a mesma como lida (WU, 2019).

A mensagem em si é algum tipo de estrutura de dados, como uma *string*, uma matriz de bytes, um registro ou um objeto e pode ser interpretado como dados, como a descrição de um comando a ser invocado no receptor ou como a descrição de um evento ocorrido no emissor. Uma mensagem contém duas partes, um cabeçalho e um corpo. O cabeçalho contém meta-informações sobre a mensagem, por exemplo, quem a enviou, entre outras informações de controle usadas pelo sistema de mensageria e que também podem ser utilizadas pelas aplicações que consomem as mensagens. O corpo da mensagem contém as informações que são consumidas pelas aplicações conectadas ao sistema de mensageria. De forma geral, um sistema de mensageria promove a comunicação entre múltiplas aplicações, permitindo a integração por meio do compartilhamento de dados e processos de modo responsivo.

Os recursos de mensagens são normalmente fornecidos por um sistema de software separado, denominado sistema de mensagens ou *Middleware Orientado a Mensagens* (MOM). Um sistema de mensagens gerencia as mensagens da mesma forma que um sistema de banco de dados gerencia a persistência de dados. Assim como um administrador deve preencher o banco de dados com o esquema para os dados de um aplicativo, um administrador deve configurar o sistema de mensagens com os canais que definem os caminhos de comunicação entre os aplicativos. O sistema de mensagens então coordena e gerencia o envio e recebimento de mensagens. O objetivo principal de um banco de dados é garantir que cada registro de dados seja persistido com segurança e, da mesma forma, a principal tarefa de um sistema de mensagens é mover mensagens do computador do remetente para o computador do receptor de maneira confiável.

Mensageria em Sistemas Distribuídos

Considerando que as redes de computadores estão sujeitas a falhas assíncronas, geradas por motivos de diferentes naturezas, quando da movimentação de mensagens entre equipamentos se faz necessário uma gerência da sua efetiva realização.

Assim, apesar de um aplicativo estar pronto para enviar uma mensagem, não significa que o outro aplicativo está pronto para recebê-la. Mesmo se os dois aplicativos estiverem prontos, a rede pode não estar operacional ou pode transmitir os dados com algum erro inserido. Um sistema de mensageria supera essas limitações tentando repetidamente transmitir a mensagem até que tenha sucesso, uma visão deste procedimento pode ser vista na Figura 2.4.

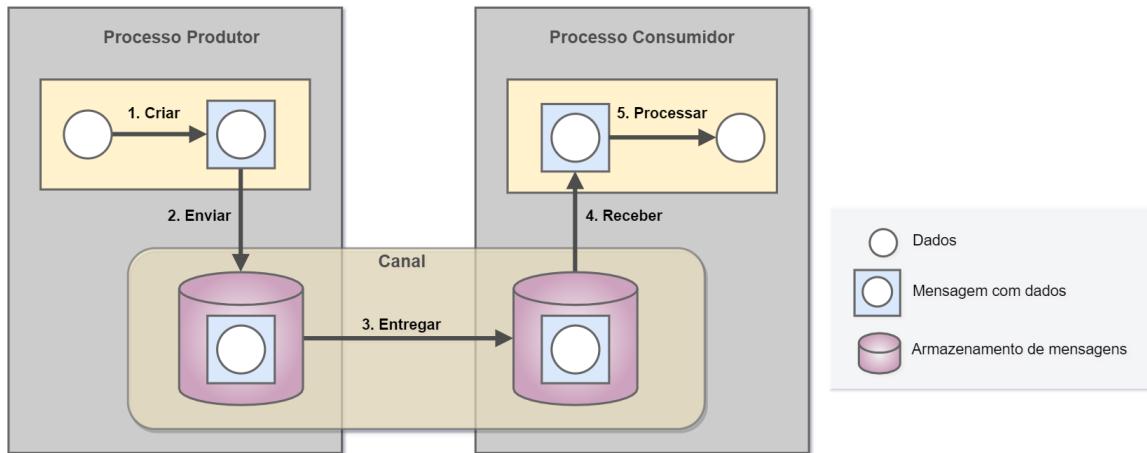


Figura 2.4 – Visão Geral dos Sistemas de Mensageria

Fonte: Adaptado de Hohpe and Woolf (2004)

Em essência, como mostra o diagrama, uma mensagem é transmitida em cinco etapas:

1. criar: o remetente cria a mensagem e a preenche com dado;
2. enviar: o remetente adiciona a mensagem a um canal;
3. entregar: o sistema de mensagens move a mensagem do computador do remetente para o computador do receptor, tornando-a disponível para o receptor;
4. receber: o receptor lê a mensagem do canal;
5. processar: o destinatário extraí os dados da mensagem.

Este diagrama da Figura 2.4 também ilustra dois conceitos importantes de mensagens:

1. enviar e esquecer: na etapa 2, o processo de envio envia a mensagem para o canal de mensagens. Assim que o envio for concluído, o remetente pode prosseguir para outro trabalho enquanto o sistema de mensagens transmite a mensagem em segundo plano. O remetente pode ter certeza de que o destinatário receberá a mensagem e não precisa esperar até que isso aconteça;
2. armazenar e encaminhar: na etapa 2, quando o processo de envio envia a mensagem para o canal de mensagem, o sistema de mensagens armazena a mensagem no computador do remetente, na memória ou no disco. Na etapa 3, o sistema de mensagens entrega a mensagem, encaminhando-a do computador do remetente para o computador do receptor e, em seguida, armazena a mensagem mais uma vez no computador do receptor. Este processo de armazenar e encaminhar pode ser repetido muitas vezes, conforme a mensagem é movida de um computador para outro, até chegar ao computador do receptor.

Ao agrupar os dados como uma mensagem e armazená-los no sistema de mensagens, os aplicativos delegam ao sistema de mensagens a responsabilidade de entregar os dados. Como os dados são agrupados como uma mensagem atômica, a entrega pode ser repetida até que seja bem-sucedida e o receptor possa ter a certeza de receber de maneira confiável exatamente uma cópia dos dados.

2.4 Considerações Finais do Capítulo

Este capítulo apresenta a Internet das Coisas, sua história e as iniciativas que fomentam o seu desenvolvimento, suas aplicações e seu impacto no novo paradigma que se apresenta à Internet. Nesta Dissertação, a visão da IoT se concretiza pela aquisição de contextos que irão alimentar repositórios de dados contextuais. A Ciência de Contexto é abordada, em suas duas grandes frentes: (1) aquisição e tratamento de dados; e (2) o ajuste do comportamento das aplicações. Para a Ciência de Situação é discutida a sua aplicação em diversos casos, os modelos que foram desenvolvidos ao longo dos anos e as premissas que permitem representar o estado de abstração das informações. Finalizando o capítulo, é discutida a mensageria enquanto uma abordagem para concepção de Sistemas Distribuídos, a qual, considerando a natureza do *Framework iMercur*, constitui parte importante da sua concepção.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados identificados por uma revisão de literatura na área foco desta dissertação. Estes trabalhos serviram como base para a concepção do iMercur, definição das tecnologias a serem empregadas, bem como para melhor posicionar as suas contribuições.

3.1 Descrição dos Trabalhos Relacionados

Nesta seção estão caracterizadas as motivações de cada um dos trabalhos selecionados, assim como as metodologias e abordagens empregadas para o desenvolvimento dos mesmos. Para tanto, seus diferentes aspectos, tanto científicos, como de apropriação tecnológica, foram considerados e analisados.

3.2 TR1 - Urban Water Demand: Statistical Optimization Approach to Modeling Daily Demand

No trabalho de Capt *et al.* (2021) os efeitos da dinâmica meteorológica e configurações socioculturais são expressos no modelo de demanda de água municipal, para modelagem de demanda de água urbana de alta precisão.

Foi considerado um modelo empírico de demanda de água urbana diária com base em uma demanda média per capita de 365 dias que incorpora funções e fatores para forças meteorológicas, sazonais, políticas e culturais.

Um modelo de regressão iterativa não linear da demanda diária de água foi calibrado e validado com dados históricos (2005-2015) para a cidade de El Paso, Texas, uma importante área urbana no sudoeste americano que tinha uma política consistente de conservação de água durante o período de estudo. O modelo inclui funções de resposta diária de temperatura e precipitação, que modificam a demanda em até 20% em relação à média anual, bem como fatores que capturam efeitos do mês do ano, dia da semana e feriados especiais.

A Figura 3.1 mostra a estrutura conceitual do modelo de demanda de água. Como uma visão geral, a previsão da demanda diária de água é baseada na demanda média anual de água (megalitros por dia), que é calculada com base na estimativa da população do dia (capita) e na média de consumo da unidade de 365 dias anterior (litros por dia per capita).

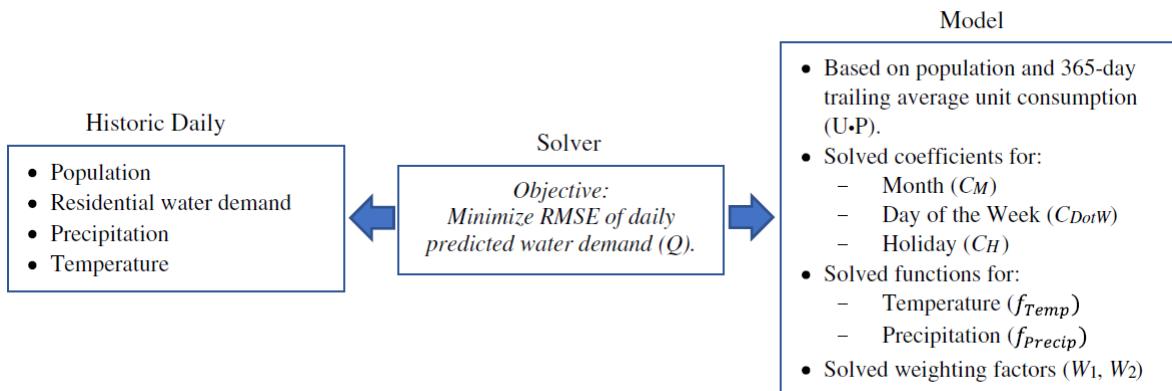


Figura 3.1 – Visão da Proposta do Trabalho TR1

Fonte: (CAPT et al., 2021)

Fatores meteorológicos e sociais capturam os efeitos do mês do ano, dia da semana, feriados especiais, temperatura média diária e precipitação anterior de 3 dias. A calibração do modelo foi melhorada minimizando o erro RMS (RMSE) das previsões diárias de demanda de água.

Um dos objetivos principais deste trabalho foi desenvolver um modelo preditivo de demanda de água que requer dados mínimos para calibração e validação, tanto para uso em plataformas computacionais simples quanto para ampla aplicabilidade entre municípios. Este modelo requer quatro conjuntos de dados de séries temporais diárias para calibração: demanda histórica de água, população, temperatura média diária e precipitação diária. Para a calibração do modelo foram considerados 5 anos (2006-2010) de dados de série temporal diárias para temperatura e precipitação, além disso, um ano anterior adicional (2005) foi necessário para demanda de água e população

O modelo usa fatores meteorológicos e sociais conhecidos para representar as forças que afetam as tendências de demanda de água com base em dados históricos de demanda de água decompostos em três constituintes: tendências de longo prazo, padrões sazonais e ruídos. O objetivo do modelo era criar conjuntos de parâmetros de ajuste que pudessem explicar esses constituintes a serem combinados para modelar a demanda diária de água. A partir da demanda de água observada, população, temperatura média diária e dados diários de precipitação, uma equação foi desenvolvida para sintetizar os três constituintes em termos matemáticos.

3.3 TR2 - An Innovative Hourly Water Demand Forecasting Preprocessing Framework with Local Outlier Correction and Adaptive Decomposition Techniques

O estudo de Hu *et al.* (2021) diz que a previsão precisa da demanda horária de água é essencial para uma operação eficaz e sustentável e para o gerenciamento econômico das redes de distribuição de água. A Figura 3.2 mostra a visão do método proposto e todo o processo do trabalho.

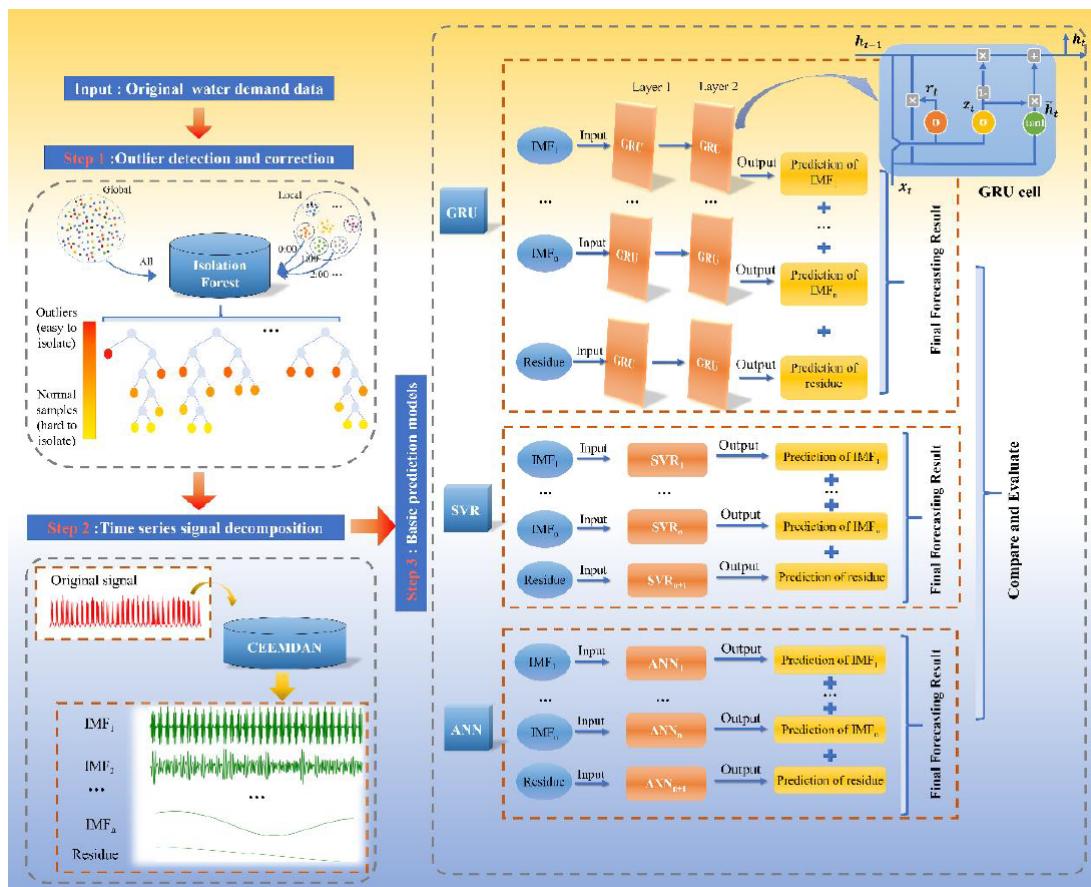


Figura 3.2 – Visão da Proposta do Trabalho TR2

Fonte: (HU et al., 2021)

Ao contrário da demanda mensal ou anual de água, a demanda horária de água tem mais flutuações e é facilmente afetada por eventos anormais de curto prazo. É necessário um método de pré-processamento eficaz para capturar os padrões de demanda de água por hora e eliminar a interferência de dados anormais.

Neste estudo é introduzida uma estrutura de pré-processamento que inclui um método de detecção e correção de *outlier* local: *Isolation Forest* (IF); uma técnica de decomposição de

sinal adaptativo: *Complete Ensemble Empirical Mode Decomposition with Adaptive Noise* (CE-EMDAN); e modelos básicos de previsão foram desenvolvidos. A fim de comparar um método promissor de aprendizado profundo, *Gated Recurrent Unit* (GRU), como um modelo básico de previsão com os modelos convencionais de previsão, *Support Vector Regression* (SVR) e *Artificial Neural Network* (ANN) foram usados.

Ao utilizar uma estrutura de pré-processamento para melhorar o desempenho dos modelos de previsão de demanda de água é demonstrado que a combinação dos modelos propostos melhora a precisão de previsão. As avaliações foram realizadas por meio de um estudo de caso, o qual foi empregado para validar o modelo, sendo obtidos resultados satisfatórios.

3.4 TR3 - Dynamic Behavioral Modeling, Simulation and Analysis of Household Water Consumption in an Urban Area: a Hybrid Approach

O trabalho de Alvi *et al.* (2018), considera o problema da escassez de água enfrentado pelo Paquistão. Com o desafio de fazer políticas drásticas de conservação da água para alcançar a gestão eficaz dos recursos hídricos disponíveis e o fornecimento eficiente de água, juntamente com a gestão responsável do lado da demanda.

Devido à falta de medidores modernos de água no Paquistão, o consumo de água não está sendo monitorado com precisão. Para atingir os objetivos relacionados a água no país, é proposta uma modelagem híbrida e um framework de simulação, consistindo em: (i) paradigma de *Agent-Based Model* (ABM) que lida com o comportamento e características dos indivíduos e (ii) paradigma *System dynamics* (SD) que responde por dinâmica do fluxo de água.

O principal objetivo da pesquisa é ajudar as autoridades a compreender e prever o consumo de água de curto e longo prazo, examinando diversos padrões de consumo de água em diferentes climas e, assim, melhorando o uso da água pelo lado da demanda de forma dinâmica, sujeito à disponibilidade de abastecimento de água.

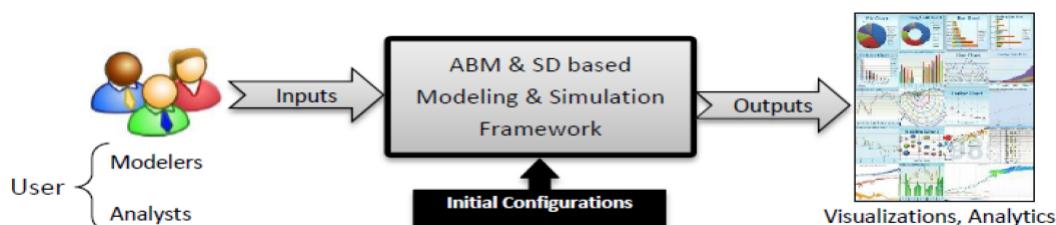


Figura 3.3 – Visão da Proposta do Trabalho TR3
Fonte: (ALVI et al., 2018)

A abordagem concebida propõe um paradigma de simulação híbrido que integra ABM com SD para alavancar o potencial de ambos os paradigmas. ABM se concentra em detalhes microscópicos do sistema, enquanto SD visa agregar os detalhes de modelagem em um nível macro.

O paradigma ABM contempla a vantagem de capturar detalhes essenciais em modelos de nível de entidade que imitam o comportamento humano e características como a faixa etária das pessoas e hábitos de consumo. ABM, no entanto, sofre de problemas de desempenho, especialmente ao lidar com populações em grande escala.

O paradigma SD, por sua vez, apresenta uma vantagem na construção de modelos em grande escala que podem ser simulados em intervalos de tempo muito longos. Também ajuda a modelar o comportamento não linear contínuo de um sistema complexo ao longo do tempo, como a quantidade de água consumida em uma atividade específica, mas carece de expressividade para comportamentos em nível de entidade.

O paradigma ABM é usado para modelar os agentes, como pessoas, casas e bairros, como entidades usando mapas de estado, enquanto o paradigma SD é usado para modelar o comportamento complexo de abastecimento e gestão de água usando estoques e fluxos.

3.5 TR4 - Predictive Classification of Water Consumption Time Series using Non-homogeneous Markov Models

O trabalho de Abadi *et al.* (2017) propõe um modelo de análise de dados de séries temporais registrados por meio de medidores inteligentes. Para isso, é empregado um modelo preditivo baseado no modelo de Markov não homogêneo *Joint Non-homogeneous Markov Model* (JNMM) com covariáveis exógenas, com a finalidade de apreender a dinâmica do comportamento de consumo de água e ser capaz de prever comportamentos de consumo futuros com um intervalo de tempo diário baseado em diferentes covariáveis exógenas pertinentes, tal como variáveis climáticas, tipo de calendário ou dia, sendo relevante o fato do dia em estudo ser um dia de semana ou fim-de-semana, ou ainda feriado ou férias escolares.

Os dados utilizados para a análise são séries temporais categóricas, onde cada série corresponde a um contador inteligente e cada categoria corresponde a um comportamento de consumo diário específico. Os experimentos são realizados com um conjunto de dados reais fornecidos por uma empresa de abastecimento de água na França.

Os pesquisadores observaram que as análises de dados de detecção de vazamento de água, de identificação de usos finais de água residenciais e de previsão da demanda de água a

curto prazo, não são satisfeitas com a coleta de informações de consumo mensal de água, sendo necessário a utilização de medidores inteligentes ou dispor de redes inteligentes de abastecimento de água, os quais tornam possível a realização dessas análises, como ilustrado na Figura 3.4.

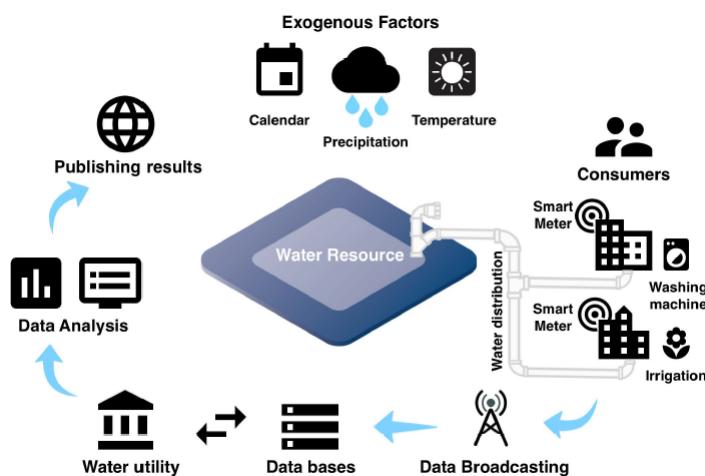


Figura 3.4 – Visão da Proposta do Trabalho TR4

Fonte: (ABADI et al., 2017)

Os experimentos realizados em um conjunto de dados do mundo real mostram o êxito do método proposto. A consideração de variáveis contextuais, ditas covariáveis exógenas, além das variáveis de consumo no modelo JNMM proposto permite capturar melhor a dinâmica dos dados de consumo, apresentando melhores resultados quando confrontados com outros dois modelos abordados para fins de comparação, sendo estes o Modelo Independente de Estado (SIM) e o Modelo homogêneo de Markov (MM).

Segundo os pesquisadores, o modelo JNMM proposto apresentou desempenho confiável, porém sua precisão ainda pode ser melhorada se a análise se concentrar em comportamentos de consumo de água mais homogêneos.

3.6 TR5 - Big Data Analytics and IoT in Operation Safety Management in Under Water

No trabalho de Nie *et al.* (2020) um método de Controle de Supervisão e Aquisição de Dados (SCADA) foi proposto usando análise de Big data e IoT para gerenciamento inteligente de recursos hídricos de uma rede de abastecimento, visando controlar proativamente o uso da água e reduzir os custos operacionais dos serviços, garantindo níveis mais sustentáveis no abastecimento de água.

Para os autores, o principal desafio na gestão da água é monitorar os níveis de água, vazamentos, qualidade da água e fluxo de água por vários canais, em que, por todos esses espaços, a IoT pode ser de grande valia. Frente a isso, juntamente com técnicas de análise de Big Data, torna possível aprimorar a gestão da água de várias maneiras, incluindo identificação de vazamentos de água, gestão de infraestrutura, monitoramento e segurança da qualidade da água, controle de qualidade dos recursos hídricos, consistência no uso e manutenção da infraestrutura, assim como promover sua manutenção preventiva.

A principal contribuição deste trabalho é: (1) fornecer um modelo matemático para o método proposto de Controle de Supervisão e Aquisição de Dados (SCADA); (2) validar a utilização da Internet das Coisas e análise de Big Data em uma estrutura de gerenciamento de água usando o modelo SCADA desenvolvido; e (3) realizar o monitoramento remoto via internet.

Para tanto, foi desenvolvido um protótipo utilizando um sistema embarcado com o microprocessador Raspberry Pi, utilizando a linguagem Python e sensores para coleta de dados do ambiente, permitindo receber e transmitir comandos dentro de uma rede. A Figura 3.5 mostra o método SCADA proposto para gerenciamento inteligente de água.

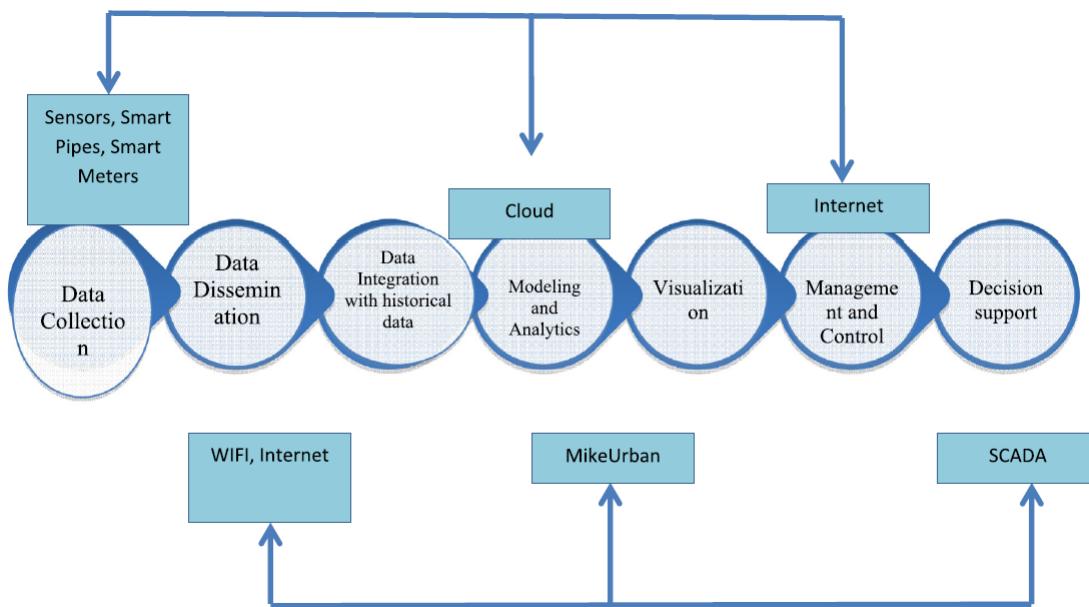


Figura 3.5 – Visão da Proposta do Trabalho TR5
Fonte: (NIE et al., 2020)

Para refinar o estudo, foram implementadas funções de simulação por software para questões de gerenciamento de redes de distribuição de água, por meio de algoritmos híbridos baseados em Modelos de Markov.

O estudo concluiu que existe uma grande quantidade de dados sobre abastecimento de água e esgoto que carecem de recursos tecnológicos que permitam que sejam processadas informações de forma rápida, isto inclui a coleta, armazenamento, análise e visualização de Big Data e sensores na IoT, de forma a melhorar o gerenciamento de processos e estratégias.

3.7 Considerações Finais do Capítulo

Os trabalhos relacionados discutidos nesta seção, de modo geral, abordam modelos para análise da demanda de água em cidades e/ou para monitoramento de seu consumo por parte da população. Todos os trabalhos são de nacionalidade estrangeira, notadamente, isto evidencia a carência de estudos mais aprofundados relativos à precisão de monitoramento e eficiência da infraestrutura de recursos hídricos no Brasil e em particular no município de Pelotas.

Os trabalhos estão em fase de consolidação da sua prototipação para disponibilização junto à comunidade usuária e apresentam limitações quanto a perspectiva de combinar dados sensoriados provenientes de várias origens, geograficamente distribuídas. Todos os trabalhos abordam uma perspectiva de investigação cujo modelo de avaliação das informações sensoriadas é predefinido quando da programação das regras empregadas.

Dentro dessa perspectiva, o iMercur introduz a capacidade de implementar diferentes modelos e estratégias de análises de dados, considerando os contextos de interesse e as visões gerenciais da comunidade técnico-administrativa do SANEP.

Neste sentido é importante registrar que na troca de experiências com os diferentes profissionais do SANEP, os mesmos evidenciaram que os modelos de análise dos dados coletados da infraestrutura de saneamento básico, são instrumentos importantes para avaliar as redes de distribuição de água, definindo logísticas operacionais de médio e longo prazo. O que corrobora com o objetivo do iMercur de facultar que as regras que compõem estes modelos possam ser definidas pelo usuário final.

Na Tabela 3.1 são comparados os trabalhos relacionados, quanto à presença ou não das características elencadas a seguir. Estas características foram selecionadas a partir de informações decorrentes de entrevistas do Grupo de Pesquisa em Processamento Paralelo e Distribuído (G3PD) com a comunidade do SANEP e seus desdobramentos ante a literatura da área.

- C1 - Acesso às informações contextuais sensoriadas de forma remota
- C2 - Gerenciamento de dados históricos de informações contextuais
- C3 - Alteração de regras pelo usuário final

- C4 - Personalização de regras pelos usuários em função do conjuntos de sensores empregado
- C5 - Utilização de conceito de ciência de contexto ou situação
- C6 - Restrito a modelos de predição
- C7 - Sensores coletados pela IoT
- C8 - Flexibilidade na Personalização do número de sensores
- C9 - Gerenciamento da Heterogeneidade de sensores e atuadores
- C10 - Restrito ao processamento de séries históricas
- C11 - Inserção e remoção de sensores em tempo de execução
- C12 - Gráfico personalizável
- C13 - Uso de API para mensageria
- C14 - Uso de mais de um processador contextual
- C15 - Uso de sensoriamento adaptativo

Tabela 3.1 – Comparaçao dos Artigos Relacionados.

	TR1	TR2	TR3	TR4	TR5	iMercur
C1	Não	Não	Não	Sim	Sim	Sim
C2	Sim	Sim	Não	Sim	Sim	Sim
C3	Não	Não	Sim	Não	Sim	Sim
C4	Não	Não	Não	Não	Não	Sim
C5	Não	Não	Não	Sim	Sim	Sim
C6	Sim	Sim	Sim	Sim	Sim	Não
C7	Não	Não	Não	Sim	Sim	Sim
C8	Não	Não	Não	Não	Não	Sim
C9	Não	Não	Não	Não	Sim	Sim
C10	Sim	Sim	Não	Sim	Sim	Não
C11	Não	Não	Não	Não	Não	Sim
C12	Sim	Sim	Sim	Sim	Sim	Sim
C13	Não	Não	Não	Não	Não	Sim
C14	Não	Não	Não	Não	Não	Sim
C15	Não	Não	Não	Não	Não	Sim

O trabalho TR1 estudo propõe um *framework* a partir da elaboração de um modelo estatístico utilizando diferentes técnicas, onde permite o gerenciamento de séries históricas sem personalização de regras e oferece o processamento de séries históricas somente para o mesmo conjunto de sensores. Permite a visualização de gráficos. O *framework* proposto não utiliza mensageria ou sensoriamento adaptativo, bem como, não trabalha com o conceito de ciência de contexto ou situação.

O trabalho TR2, assim como o anterior, também propõe um *framework*, que entretanto utiliza diferentes algoritmos de inteligência artificial a partir de dados de séries históricas. Os dados são consumidos a partir de regras fixas. Não havendo gerenciamento da heterogeneidade dos sensores. Permite a visualização de gráficos. Não oferece sensoriamento adaptativo e não explora os conceitos de ciência de contexto ou situação.

O trabalho TR3 propõe um *framework* de análise e simulação utilizando uma abordagem, que combina outros modelos definidos previamente, no caso são explorados dois modelos. Os parâmetros de consumo são estimados a partir do estudo de funções de probabilidades para o consumo de água residencial de uma família do estado da Califórnia. As regras podem ser ajustadas a partir de ajustes em tabelas de probabilidades utilizadas no estudo. Não trabalha com conceito de ciência de contexto e situação ou sensoriamento, oferecendo ao usuário a visualização gráfica dos dados simulados.

O trabalho TR4 utiliza dados reais de consumo residencial capturados por medidores inteligentes, coletados por IoT, provendo um modelo de previsão a partir do emprego de um modelo preditivo utilizando dados de séries temporais. Apresenta o conceito de ciência de contexto, permitindo a visualização gráfica das informações.

O trabalho TR5 propõe um sistema supervisório para o gerenciamento de dados reais de monitoramento de diferentes fontes, utiliza acesso a informações remotas e gerenciamento de dados históricos, sendo restrito a modelos de predição. Não permite a personalização de regras por conjunto de sensores e também não permite a alteração do número de sensores. Faz a utilização de conceito de ciência de contexto, além de possibilitar, como todos os outros trabalhos, uma visualização gráfica das informações.

4 MIDDLEWARE EXEHDA: ARQUITETURA E FUNCIONALIDADES

Neste capítulo é apresentada uma revisão sobre o *middleware* EXEHDA (*Execution Environment for Highly Distributed Applications*) e seus subsistemas. A qual contribui para o entendimento de como os componentes propostos no iMercur são mapeados nos componentes arquiteturais do *middleware*.

O EXEHDA é um *middleware* adaptativo ao contexto e baseado em serviços, que visa criar e gerenciar um ambiente ubíquo, bem como, promover a execução, sob esse ambiente, de aplicações largamente distribuídas como as introduzidas pelo cenário computacional da Internet das Coisas. Seu objetivo é permitir que as aplicações possam obter informações de seus contextos de interesse e reagir às variações que acontecem nos mesmos (LOPES et al., 2013).

4.1 Aspectos Funcionais e Arquiteturais

O EXEHDA tem como objetivo a definição de uma arquitetura para um ambiente de execução, destinado às aplicações nas quais as condições de contexto são monitoradas de forma proativa e o suporte à execução deve permitir que, tanto a aplicação como ele próprio, utilizem essas informações na gerência da adaptação de seus aspectos funcionais e não-funcionais. Entende-se por adaptação funcional, aquela que implica na modificação do código sendo executado. Por sua vez, adaptação não funcional é aquela que atua sobre a gerência dos parâmetros operacionais que regem a execução distribuída. Também a premissa siga-me das aplicações ubíquas deverá ser suportada, garantindo a execução da aplicação do usuário em qualquer tempo, lugar e dispositivo (LOPES, 2016).

Os principais requisitos que o EXEHDA atende são:

- gerenciar, de modo independente, tanto aspectos não funcionais como funcionais das aplicações distribuídas;
- dar suporte à adaptação dinâmica de aplicações;
- disponibilizar mecanismos para obter e tratar informações sensoriadas, inferindo ciência de contexto e situação;
- empregar informações de contexto na tomada de decisões;
- decidir as ações adaptativas de forma colaborativa com a aplicação;

- disponibilizar a semântica *siga-me*, permitindo ao usuário acompanhar as aplicações e acessar dados a partir de qualquer lugar.

A arquitetura de software do *middleware* EXEHDA, caracterizada na Figura 4.1, visa fornecer uma solução integrada para construir e executar aplicativos distribuídos em uma perspectiva de elevada escalabilidade quanto ao total de dispositivos interoperando.

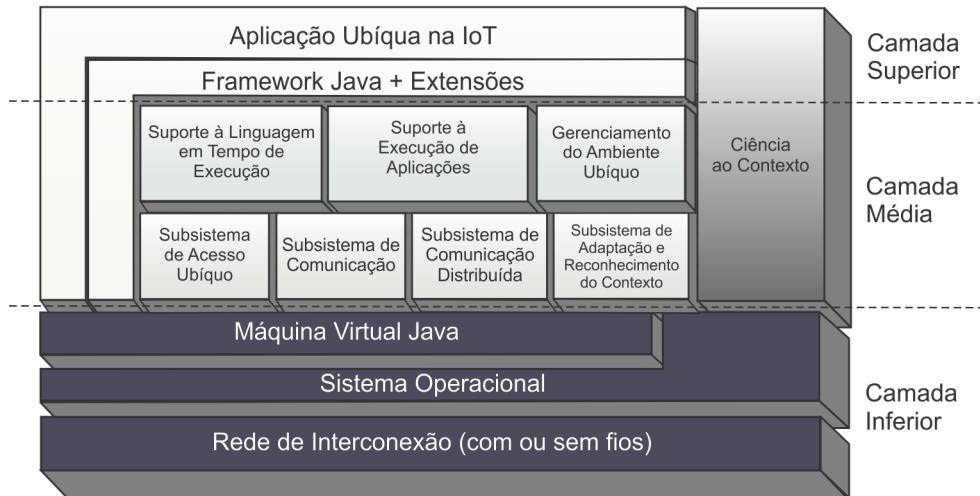


Figura 4.1 – Arquitetura de Software do *Middleware* EXEHDA

Fonte: Adaptada de (LOPES et al., 2014)

A arquitetura do *middleware* EXEHDA é dividida em uma organização lógica de três camadas. Cada uma destas camadas tem um nível específico de abstração:

- camada de aplicação (superior);
- camada de suporte e ambiente de execução (média);
- camada dos sistemas básicos (inferior).

A camada superior corresponde às abstrações que o designer do aplicativo fornece para facilitar o desenvolvimento de um aplicativo adaptativo que explore informações contextuais. Isso é obtido, principalmente, pelo fornecimento de um *framework* Java. Também temos nessa e na próxima camada, a representação da Ciência de Contexto. A razão para isso, é ressaltar sua importância na arquitetura, destacando sua presença na concepção dos componentes das aplicações suportadas pelo EXEHDA.

Na camada intermediária estão os mecanismos de apoio para a implementação de estratégias de computação distribuída em uma perspectiva ubíqua. Essa camada possui dois níveis, o primeiro nível consiste nos módulos de serviço do aplicativo e o segundo nível é formado pelos serviços básicos do EXEHDA. Esses serviços básicos possibilitam recursos necessários

para o nível superior e abrangem vários aspectos, como acesso ubíquo, comunicação, execução distribuída, reconhecimento de contexto e adaptação.

Finalmente, a camada inferior da arquitetura é composta de linguagens nativas e sistemas que integram o ambiente físico de execução. Por razões de portabilidade, nessa camada, a plataforma para implementação é uma *Java Virtual Machine* em suas diferentes abordagens. A arquitetura pressupõe a existência de uma rede para suportar a execução de componentes e serviços em escala global (LOPES et al., 2014; LOPES, 2016).

EXEHDA tem como requisito permanecer operacional durante os períodos de desconexão planejada. Para dar suporte a esse recurso, os serviços são divididos em duas partes, uma instância de nodo e uma instância de celular. O primeiro é o local para cada dispositivo, enquanto o último, executa no nodo base. Assim, o dispositivo local poderá estar operacional durante o desligamento planejado, considerando que a instância do nodo do serviço deve renunciar, temporariamente, ao acesso dos recursos que estão na rede. Por outro lado, a instância celular do serviço, em execução no nodo base da célula, atua como um ponto de referência para serviços que exigem procedimentos de coordenação distribuídos, inter-nodos ou inter-células (LOPES et al., 2014).

4.2 Ambiente Ubíquo na IoT provido pelo *middleware* EXEHDA

O ambiente computacional ubíquo gerenciado pelo *middleware* EXEHDA para uso em aplicações da IoT tem a sua organização, conforme a Figura 4.2.

Considerando a perspectiva de promover a computação de aplicações na IoT, atendendo a critérios de distributividade, escalabilidade, heterogeneidade, mobilidade e adaptabilidade ao contexto, este ambiente é constituído por células de execução. Deste modo os dispositivos computacionais são distribuídos entre as células, sendo cada célula constituída dos seguintes componentes:

- EXEHDAbase, o elemento central da célula, sendo responsável por todo serviços e constituindo referência para os demais elementos;
- EXEHDAano que corresponde aos dispositivos computacionais responsáveis pela execução das aplicações;
- EXEHDAano móvel, um subcaso do anterior, que corresponde aos dispositivos tipicamente móveis que podem se deslocar entre as células do ambiente ubíquo, como notebooks, tablets ou smartphones;

- EXEHDAbronda, responsável por fazer a interoperação entre os serviços do *middleware* e os diversos tipos de gateways;
- EXEHDAgateway, que consiste no elemento responsável por setorizar pontos de coleta e/ou atuação distribuídos, disponíveis no meio físico, realizando a interação destes com os outros componentes do *middleware*.

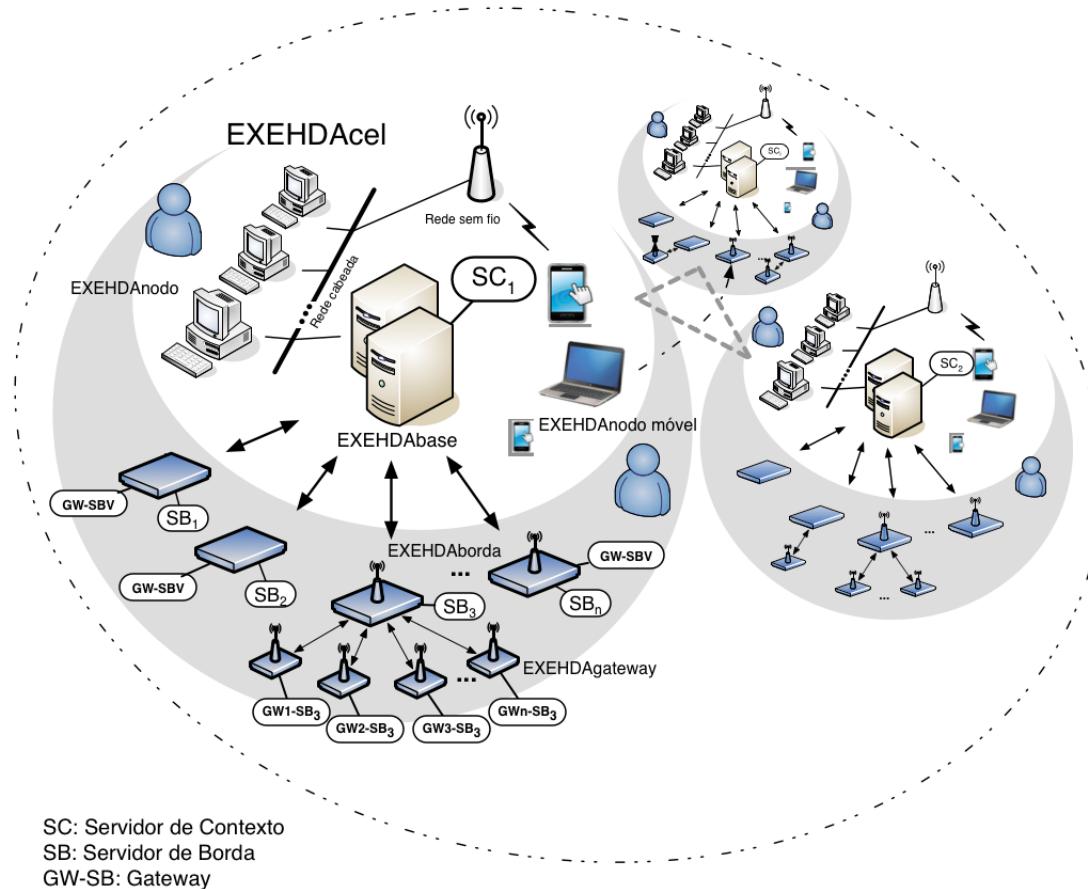


Figura 4.2 – Ambiente na IoT Provido pelo *Middleware* EXEHDA
Fonte: Adaptado de (SOUZA et al., 2019)

Para provimento de Ciência de Contexto no ambiente ubíquo, o EXEHDA se vale de dois tipos principais de servidores da arquitetura do EXEHDA: Servidor de Borda e Servidor de Contexto (vide Figura 4.2). O Servidor de Borda se destina a gerenciar a interação com o meio físico através de *Gateways*, sendo instanciado em um equipamento do tipo EXEHDAbronda. O Servidor de Contexto, por sua vez, é alocado no EXEHDAbase e atua no armazenamento e no processamento das informações contextuais, integrando dados históricos e dados provenientes de diferentes Servidores de Borda distribuídos no ambiente .

A premissa é de que os sensores e/ou atuadores sejam integrados ao Servidor de Borda somente através de *Gateways*. Os *Gateways* são utilizados, então, para tratar os diversos tipos de protocolos físicos inerentes a dispositivos de sensoriamento e/ou atuação, bem como garantir que dispositivos com capacidade restrita, tanto computacional como energética, possam se comunicar com o Servidor de Borda via TCP/IP.

Os *Gateways* possuem hardware e capacidades computacionais limitadas, sendo dedicados para tratar tecnologias específicas, como redes de sensores sem fio, por exemplo, fazendo a conversão de protocolos e o gerenciamento dos dispositivos.

O *middleware* além de ser uma solução para o acesso ubíquo na perspectiva da IoT, tem como finalidade prover uma arquitetura de comunicação e execução distribuída de processos. Sua arquitetura fornece mecanismos para concepção de procedimentos para obter e tratar informações de contexto e realizar seu processamento a partir de regras construídas para o cenário de aplicação considerado.

4.3 Considerações Finais do Capítulo

Neste capítulo são revisados aspectos centrais do *middleware* EXEHDA. Os aspectos descritos foram selecionados considerando o significado dos mesmos para a natureza do *Framework* iMercur. O EXEHDA oferece uma arquitetura de execução e comunicação entre processos. Sua arquitetura dispõe de mecanismos para concepção de procedimentos para obter e tratar informações de contexto e realizar seu processamento a partir de regras contextuais, oferecendo uma solução para o acesso ubíquo na perspectiva da IoT.

5 IMERCUR: CONCEPÇÃO E TECNOLOGIAS

Neste capítulo é apresentada a concepção do *Framework* iMercur, bem como as diferentes tecnologias que foram selecionadas para o seu desenvolvimento e operação, as quais foram selecionadas considerando sua adoção pela comunidade técnico-científica, bem como sua disponibilização enquanto software *open source* (vide seção 5.7).

Neste sentido, são discutidas as diferentes unidades e módulos constituintes do iMercur, sendo caracterizadas as formas como estas diferentes unidades interoperam com a finalidade de prover as funcionalidades desejadas para emprego do iMercur diretamente pela comunidade usuária.

5.1 Visão geral

A arquitetura de software concebida no G3PD, nos trabalhos já realizados para atendimento das necessidades do SANEP, é formada por quatro unidades, conforme pode ser visto na sua arquitetura de software apresentada na Figura 5.1.



Figura 5.1 – Framework iMercur na Arquitetura de Software Concebida pelo G3PD para o SANEP
 Fonte: Adaptada de Londero (2021)

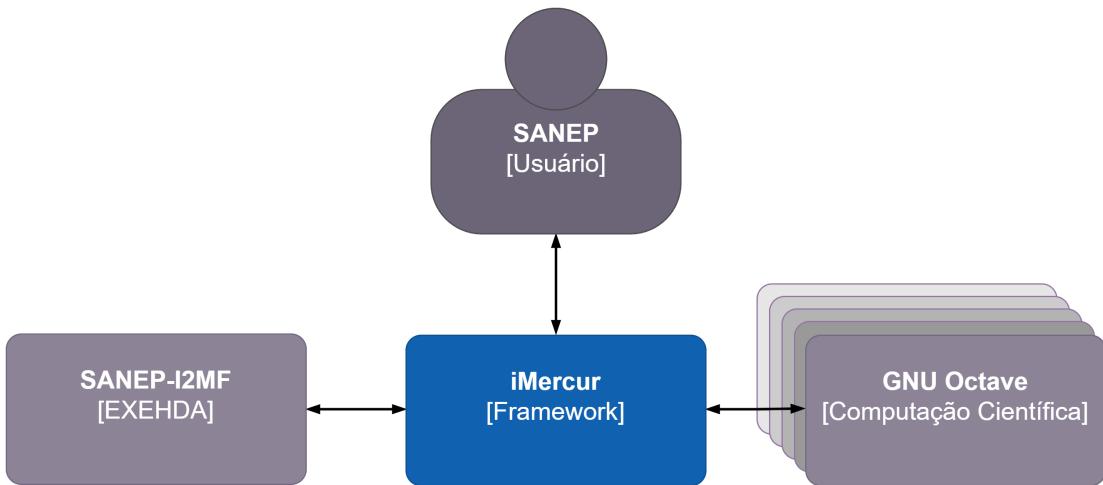


Figura 5.2 – Visão Geral do *Framework* iMercur

(Diagrama C4 de Nível 1)

Fonte: Elaborado pelo autor

A principal contribuição do *Framework* iMercur consiste uma ampla reorganização da Unidade de Atendimento ao Usuário, com destaque a nova estratégia para a Unidade de Inferência de Situação, potencializando a participação do usuário na especificação de regras, para inferência de contextos e situações, com diferentes níveis de complexidade.

Para atingir esta principal contribuição, o iMercur¹ explora a interoperabilidade com recursos de processamento de diferentes naturezas conforme a visão exibida na Figura 5.2. Nesta Figura, além do GNU Octave, enquanto plataforma que foi selecionada para ser empregada nesta Dissertação, fica caracterizada a possibilidade de integração com outras plataformas de computação científica, considerando a sua conveniência em função da natureza das demandas da comunidade usuária.

Os diagramas das figuras que descrevem a concepção do *Framework* iMercur seguem o Modelo C4², que é uma abordagem para diagramar arquiteturas de software. O modelo C4 é apresentado, com os seus níveis de abstração, na seção 5.7.

5.2 Arquitetura de Software Proposta

A arquitetura de software do *Framework* iMercur é apresentada na Figura 5.3, incluindo suas quatro unidades distintas concebidas para o provimento das suas funcionalidades. Cada unidade do iMercur é composta por módulos com funcionalidades específicas e suas respectivas

¹Mercúrio, emissário portador das mensagens de Júpiter aos outros deuses.

²<https://c4model.com>

interfaces com os módulos das unidades subjacentes e com o usuário final. Deste modo, a Figura 5.3 apresenta as unidades concebidas para composição do iMercur, caracterizando a comunicação entre as mesmas e com os sistemas externos, bem como com o usuário final.

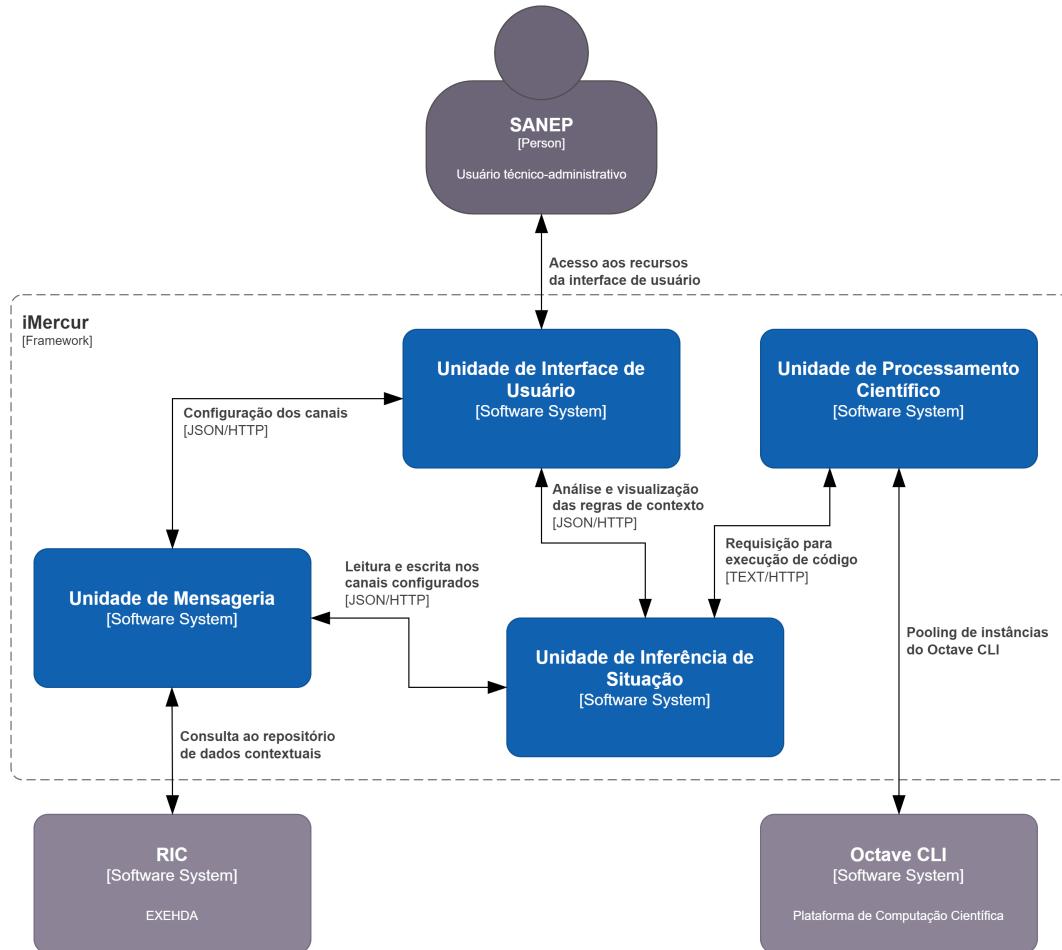


Figura 5.3 – Visão Geral das Unidades do *Framework* iMercur

(Diagrama C4 de Nível 1)

Fonte: Elaborado pelo autor

O usuário final, por meio da Unidade de Interface de Usuário, tem acesso a recursos que permitem a configuração dos canais de dados de contexto, o gerenciamento de regras para análises contextuais e o acesso a diferentes alternativas para visualização gráfica das informações.

A Unidade de Mensageria, por sua vez, carrega os dados do contexto de interesse a partir do Repositório de Informações Contextuais (RIC) nos canais configurados pelo usuário final, disponibilizando estes dados para a Unidade de Inferência de Situação.

A partir dos dados contextuais carregados dos canais e das regras de contexto definidas pelo usuário, a Unidade de Inferência de Situação envia essas informações para a Unidade de Processamento Científico para o tratamento dessas regras pela Plataforma de Computação Científica selecionada para uso. Os resultados são disponibilizados ao usuário final por meio da

Unidade de Interface de Usuário. Na continuidade, juntamente com a descrição das funcionalidades dessas unidades, estão as especificações de cada um dos módulos que as compõem.

5.3 Unidade de Atendimento ao Usuário

A Unidade de Atendimento ao Usuário do iMercur permite o acesso a todas as grandes sensoriadas nos diferentes artefatos espalhados pela cidade, possibilitando diferentes alternativas tanto para a visualização das diferentes informações obtidas do meio, assim como, os resultados das diferentes análises realizadas.

Além da escolha do contexto para obtenção dos dados através do sensoriamento remoto, também há possibilidade da seleção dos sensores a partir do contexto escolhido. Como grandes sensoriadas destacam-se: nível de água, concentração de cloro, temperatura ambiente, temperatura da água, umidade relativa do ar, entre outros.

O Módulo de Interface de Usuário comprehende as diferentes funções oferecidas pelo iMercur e se destina a permitir ao usuário o registro de especificações de configuração de regras e canais, bem como a visualização textual e gráfica das diferentes informações manipuladas. Este módulo é responsável pelo gerenciamento das regras contextuais cadastradas pelos usuários e a apresentação dos resultados decorrentes do processamento das mesmas. Na Interface de Usuário, o *front-end* é responsável pela parte visual que os usuários irão acessar e interagir por meio de seus dispositivos *on-line*.

O *front-end* do iMercur realiza chamadas HTTP a APIs REST, que são as interfaces de programação por onde as solicitações da Unidade de Atendimento ao Usuário são atendidas. As telas de interface de usuário serão descritas e detalhadas nas próximas seções conforme a necessidade de utilização da interface em cada módulo.

5.4 Unidade de Inferência de Situação

Esta unidade tem suas funcionalidades organizadas em três componentes arquiteturais, como mostra a Figura 5.4. O RIGIM (Repositório de Informações Gerenciais do iMercur) consiste em um Banco de Dados e foi concebido com a finalidade de armazenar os dados de configurações necessárias às operações da arquitetura do iMercur, são estes: dados de configuração de canais; dados de configuração de regras; e informações de usuários. Os dados de configurações são persistidos para garantir a repetibilidade das execuções das regras.

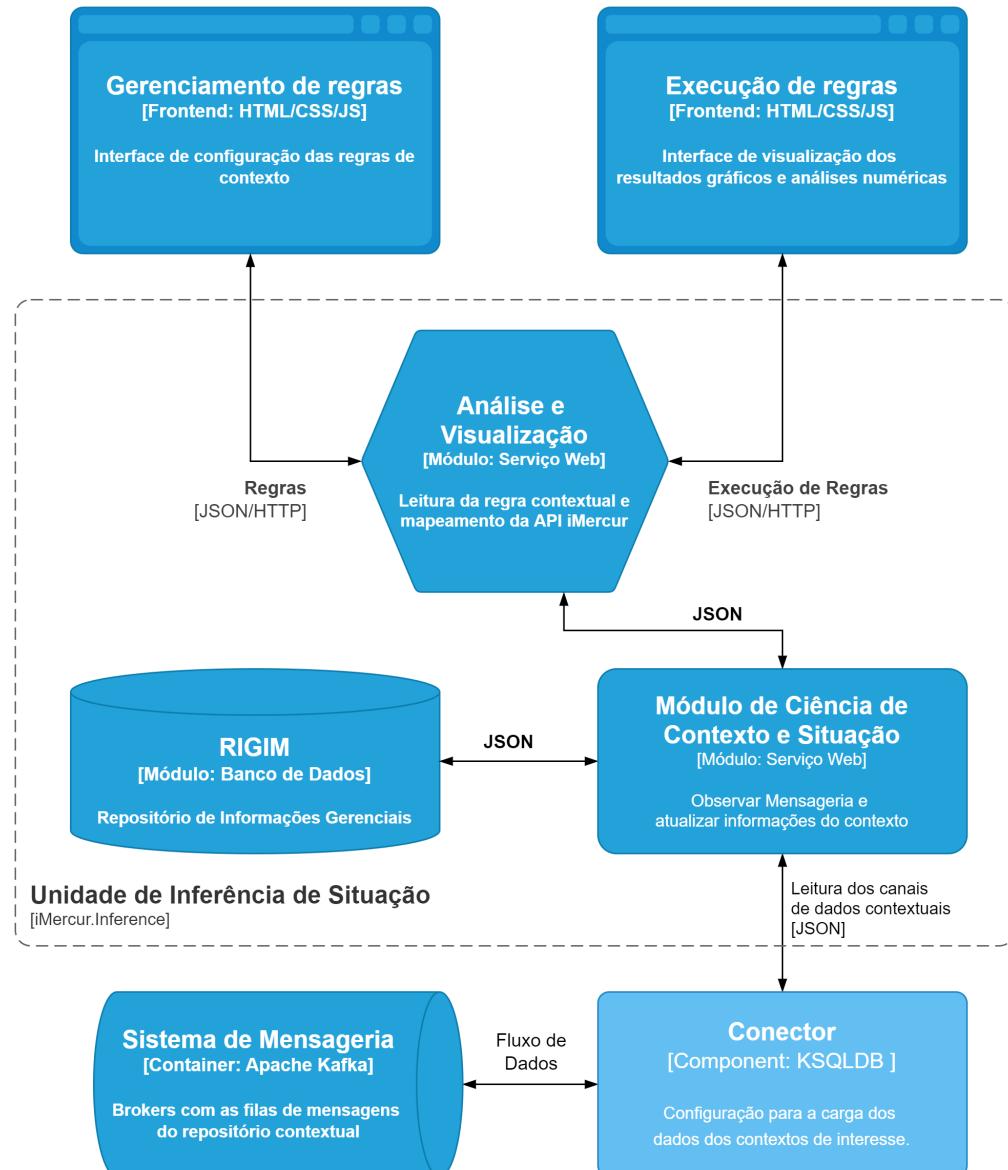


Figura 5.4 – Unidade de Inferência de Situação

Fonte: Elaborado pelo autor

O serviço Web de Análise e Visualização é responsável pelo gerenciamento das regras contextuais, pelo mapeamento de chamadas à API de leitura de canais que o *Framework* iMercur implementa e pelas solicitações para a execução das regras. Os métodos HTTP REST que o serviço Web implementa são apresentados na Tabela 5.1.

A identificação (ruleContextId) de uma regra é feita por uma estrutura GUID³, implementação de tipo do framework .NET na linguagem C#, a qual representa um identificador único global, sendo um valor de 16 bytes gerado randomicamente. Para criar uma regra é enviado no corpo da requisição HTTP POST a estrutura representada pelo JSON da Figura 5.5.

Tabela 5.1 – Métodos da API REST para a Configuração de Regras

Método	Descrição
POST /Rule/Create	Cria uma nova regra
GET /Rule/{ruleContextId}	Obtém informações de uma regra
GET /Rule/Execute/{ruleContextId}	Executa uma regra
GET /Rule	Obtém todas as regras
PUT /Rule	Atualiza as configurações de uma regra
DELETE /Rule/{ruleContextId}	Exclui uma regra

```

1   {
2     "ruleName": string,
3     "ruleDescription": string ,
4     "rule": string ,
5     "recurrence": boolean,
6     "recurrenceTime": integer
7 }
```

Figura 5.5 – JSON de Criação de uma Regra

Fonte: Elaborado pelo autor

De modo análogo, para atualizar uma regra é enviado junto ao corpo da requisição HTTP PUT o JSON da Figura 5.6.

```

1   {
2     "ruleContextId": string (GUID)
3     "ruleName": string,
4     "ruleDescription": string ,
5     "rule": string ,
6     "recurrence": boolean,
7     "recurrenceTime": integer
8 }
```

Figura 5.6 – JSON de Atualização de uma Regra

Fonte: Elaborado pelo autor

As propriedades utilizadas para criar/atualizar regras são as seguintes:

- ruleContextId – identificador único da regra;
- ruleName – nome da regra;
- ruleDescription – descrição da regra;
- rule – regra propriamente dita (o código Octave);

³<https://learn.microsoft.com/pt-br/dotnet/api/system.guid>

- `recurrence` – campo indicativo que a regra será executada repetidamente;
- `recurrenceTime` – tempo entre execuções de uma regra marcada como recorrente.

Este serviço Web, ao receber uma requisição HTTP para executar uma regra, realiza a leitura das informações provenientes dos canais de fluxos de dados, carregando estas informações nas variáveis da regra.

Dessa forma, para que os dados provenientes da leitura de um canal possam ser carregados por uma regra foi desenvolvida uma API com métodos de leitura de canais, interpretada pelo *Framework* iMercur, que se integra ao código da regra e tem como retorno uma matriz de dados numéricos carregados a partir da leitura do canal configurado.

No momento da execução da regra, ocorre um mapeamento dos métodos da API de leitura, onde os mesmos são substituídos pela matriz de dados numéricos retornada. A sintaxe e descrição dos métodos são mostradas na Figura 5.7.

```

1  data = iMercurRead(string channelId, integer time);
2
3  data = iMercurRead(string channelId, string start, string end);

```

Figura 5.7 – API do iMercur para Trocas de Dados

Fonte: Elaborado pelo autor

- `string channelId` – este parâmetro é o identificador do canal que os dados contextuais são carregados, consiste no nome do canal, sendo um identificador único criado no cadastro de canal;
- `integer time` – parâmetro que especifica o tempo anterior, em minutos, que será analisado, do momento de execução da regra;
- `string start`: parâmetro que especifica a data inicial que será analisada, sendo representada no formato `yyyy-MM-ddTHH:mm:ss`;
- `string end`: parâmetro que especifica a data final que será analisada, sendo representada no formato `yyyy-MM-ddTHH:mm:ss`.

O Módulo de Ciência de Contexto e Situação é um serviço baseado em regras, de forma a tratar eventos gerados a partir de mudanças nos estados dos contextos de interesse. Podem ser combinadas informações de diferentes sensores. O gatilho para o disparo de qualquer regra de processamento contextual é a própria chegada do dado contextual na Unidade de Inferência de Situação, para isso as regras precisam estar configuradas para execução recorrente.

Com o objetivo de deduzir situações referentes aos artefatos sendo sensoriados, a Unidade de Inferência de Situação é detalhada no diagrama de sequência da Figura 5.8, mostrando a troca de mensagens entre os componentes.

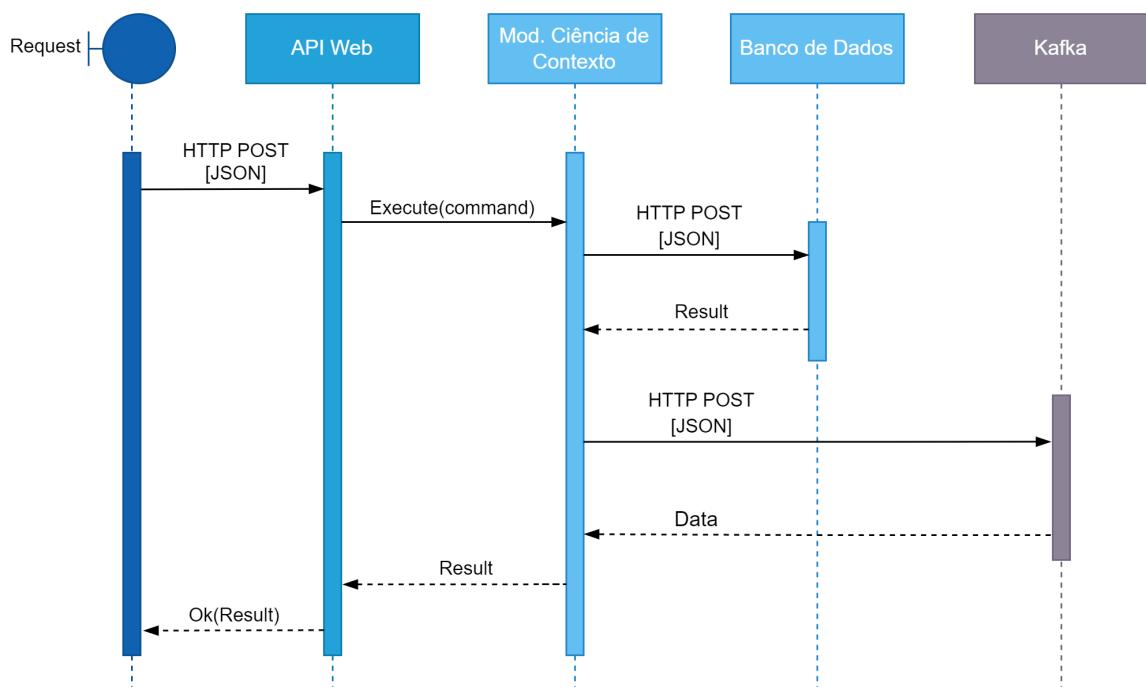


Figura 5.8 – Diagrama de Sequência da Unidade de Inferência e Situação
Fonte: Elaborado pelo autor

Eventos de captura de contexto, cuja ocorrência é de natureza periódica, ou produzidos por estados pré-definidos para os diferentes sensores envolvidos, podem disparar regras. Estas regras podem ser de diferentes naturezas, baseadas em especificação pelo usuário, ou explorarem aprendizado de máquina, ou até mesmo combinarem diferentes estratégias para modelagem das variáveis contextuais e a inferência decorrente das mesmas. Desta inferência são identificadas situações, as quais podem disparar diferentes ações.

As regras de contexto são construídas a partir dos mecanismos de seleção dos dados contextuais, os quais serão escolhidos com suporte das funcionalidades propostas pelo iMercur, acessíveis a partir do Módulo de Interface do Usuário, na Unidade de Atendimento ao Usuário. A Figura 5.9 apresenta a tela de cadastro de regras.

iMercur Regras Canais Sensores admin@imercur.com Logout

Criar regra

Nome
Regra 1

Descrição
Média da temperatura da última hora

Regra

```
period = 60;
temperature = iMercurRead(Canal1, period);
average = mean(temperature);
```

Recorrência

Período
2

Criar

[Voltar para lista](#)

Figura 5.9 – Tela de Cadastro de Regras

Fonte: Elaborado pelo autor

Nesta tela, são campos obrigatórios:

- Nome – representa o nome da regra;
- Descrição – uma breve descrição da regra;
- Regra – o código da regra escrita para a plataforma de computação que irá executar, neste caso o Octave.
- Recorrência – caixa de checagem que marca a regra como recorrente;
- Período – tempo em minutos para a execução recorrente da regra.

A tela da Figura 5.10 mostra a lista de regras criadas. Nesta tela estão listadas o nome, a descrição e a data da última execução da regra. Juntamente com cada registro estão as opções para executar, editar, ver detalhes e deletar a regra.

Regras

[Criar nova](#)

Nome	Descrição	Executado em	
Regra 1	Média da temperatura da última hora	30/11/2022 21:19:14	Executar Editar Detalhes Deletar
Regra 2	Gráfico da função da curva de demanda no dia	30/11/2022 21:19:01	Executar Editar Detalhes Deletar

Figura 5.10 – Tela de Listagem de Regras

Fonte: Elaborado pelo autor

A Figura 5.12 exibe a tela de execução com o resultado da execução de uma regra.

Executar regra

Nome	Regra 1
Descrição	Média da temperatura da última hora
Resultado	average = 26.6000

Executado 30/11/2022 21:19:14
em

[Voltar para lista](#)

Figura 5.11 – Tela de Execução de Regras

Fonte: Elaborado pelo autor

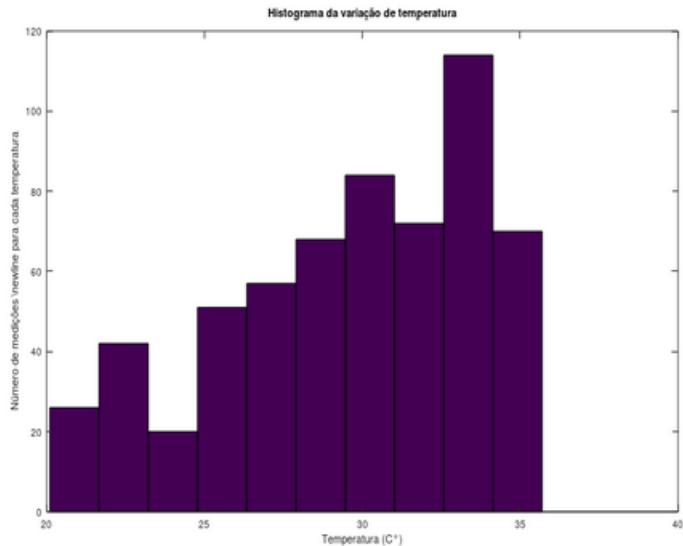
A Figura 5.12 exibe a tela de execução com o resultado da execução de uma regra que plota um gráfico.

Executar regra

Nome Regra 2

Descrição Histograma da variação de temperatura

Resultado



Executado 30/11/2022 21:19:01
em

[Voltar para lista](#)

Figura 5.12 – Tela de Execução de Regras com Gráfico
Fonte: Elaborado pelo autor

5.5 Unidade de Mensageria

A Unidade de Mensageria é responsável pela integração entre as bases de dados contextuais (RICs) e o *Framework* iMercur, como mostra a Figura 5.13.

Esta unidade é composta por uma API Web responsável pelas chamadas à plataforma do Apache Kafka, sendo utilizados como principais recursos para este trabalho: o próprio Apache Kafka, que é o *broker* gerenciador das filas de mensagens, chamadas de tópicos; o *framework* Kafka Connect que funciona como um serviço produtor/consumidor entre a fonte de dados contextuais e os tópicos do *broker*; e o *framework* KsqlDB que é um banco de dados desenvolvido especificamente para aplicações que utilizam processamento de fluxos de dados. O Kafka Connect e o KsqlDB são apresentados em mais detalhes na seção 5.7.

O serviço de Web API oferece uma interface *web* para o gerenciamento dos canais pelos quais os dados de contexto são carregados. Cada canal é uma fila de mensagens que

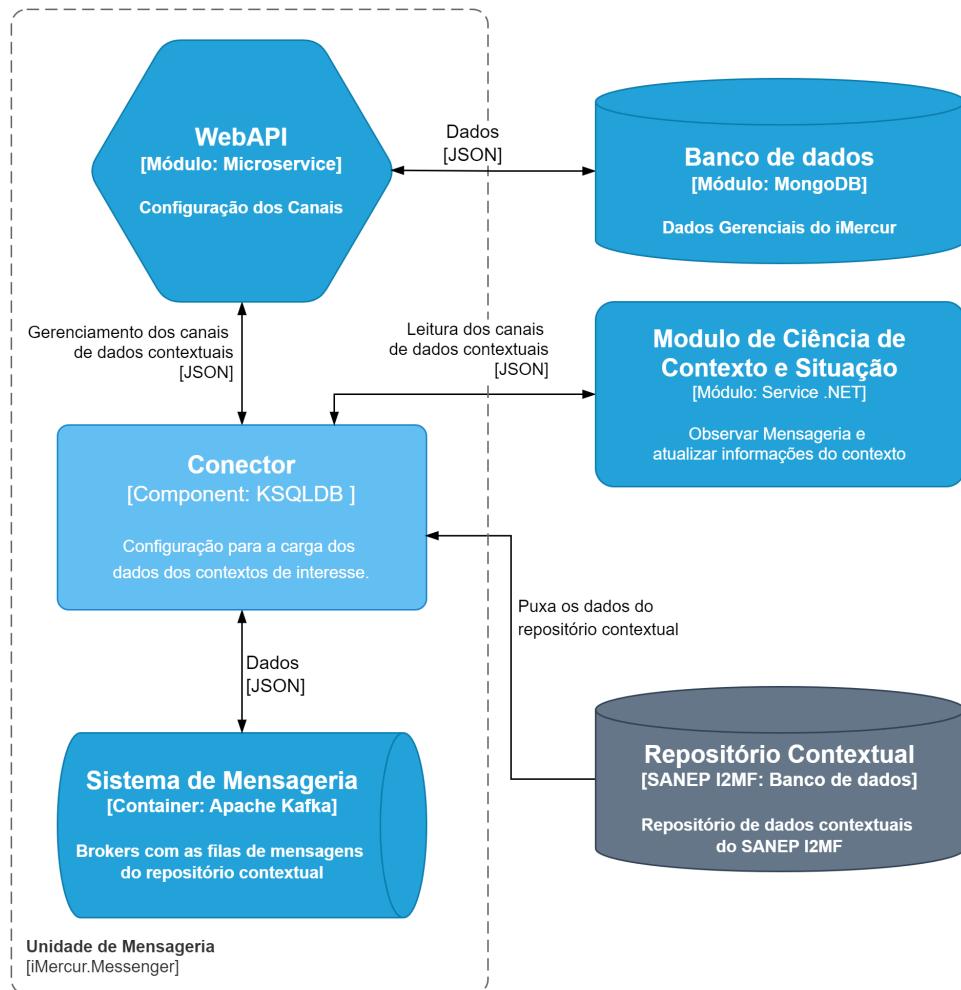


Figura 5.13 – Unidade de Mensageria

Fonte: Elaborado pelo autor

internamente é chamada de tópico pelo Kafka. Quando um canal é configurado pelo usuário, os dados do contexto de interesse a ser processado, são carregados em tempo real do repositório de dados contextuais para um determinado tópico. A Tabela 5.2 mostra os métodos HTTP da interface de gerenciamento de canais.

Tabela 5.2 – Métodos da API REST para a Configuração dos Canais

Método	Descrição
GET /Channel/id	Obtém informações do canal
POST /Channel	Cria um novo canal
GET /Channel	Obtém todos os canais
PUT /Channel	Atualiza as configurações do canal
DELETE /Channel/id	Exclui o canal, parando suas tarefas

As mensagens de um tópico são consumidas pelo Módulo de Ciência de Contexto e Situação, da Unidade de Inferência de Situação, de acordo com o contexto de interesse que a

regra de processamento contextual solicita. Para este trabalho, os dados contextuais são extraídos de um banco de dados relacional PostgreSQL. No entanto, a utilização do *Framework* Kafka Connect torna flexível o uso de diferentes fontes de dados, possibilitando conectar de forma concomitante diversos repositórios de dados contextuais. Este acesso pode ser feito por diferentes protocolos, por exemplo MQTT, FTP, SFTP, entre outros.

Nome	Descrição	Ativo	
Canal1	Dados de unidade e temperatura.	<input checked="" type="checkbox"/>	Editar Detalhes Apagar
Canal2	Dados de temperatura	<input checked="" type="checkbox"/>	Editar Detalhes Apagar

Figura 5.14 – Tela de Listagem de Canais

Fonte: Elaborado pelo autor

A Figura 5.14 apresenta a interface de usuário com a listagem dos canais configurados. Esta interface faz uma solicitação GET /Channel para o serviço Web API que lista os canais configurados.

Para a criação de um canal são necessários três passos: (i) criar um conector para acessar o repositório de dados contextuais; (ii) criar um fluxo de dados das informações que estão sendo atualizadas no repositório de dados contextuais; e (iii) registrar esta configuração no banco de dados do iMercur. A Figura 5.15 mostra a tela de cadastro de um canal no iMercur.

iMercur Regras Canais Sensores admin@imercur.com Logout

Criar Canal

Nome

Descrição

Conexão

Usuário

Senha

Consulta

Ativo

Criar

Figura 5.15 – Tela de Criação de um Canal

Fonte: Elaborado pelo autor

5.6 Unidade de Processamento Científico

A Unidade de Processamento Científico é composta por dois módulos: (1) um Módulo de Integração, que incorpora uma API Web e (2) o Módulo de Interoperação, que interage com o Octave CLI (*Command Line Interface*):

- O Módulo de Integração implementa um método de requisição POST que lê a informação do corpo da requisição, enviada pela Unidade de Inferência de Situação;
- O Módulo de Interoperação, é o responsável por interoperar com a Plataforma de Computação Científica e pelo retorno do resultado do processamento da informação ao módulo que encaminhou mensagem.

O diagrama da Figura 5.16 representa a ligação entre estes componentes que compõem a unidade.

Um método de requisição POST aceita os dados anexados no corpo da mensagem. Estes dados são tratados pelo Módulo de Integração como sendo do tipo texto e representa o código-fonte que será executado pelo Octave CLI. Ao chegar uma requisição, o Módulo de Integração

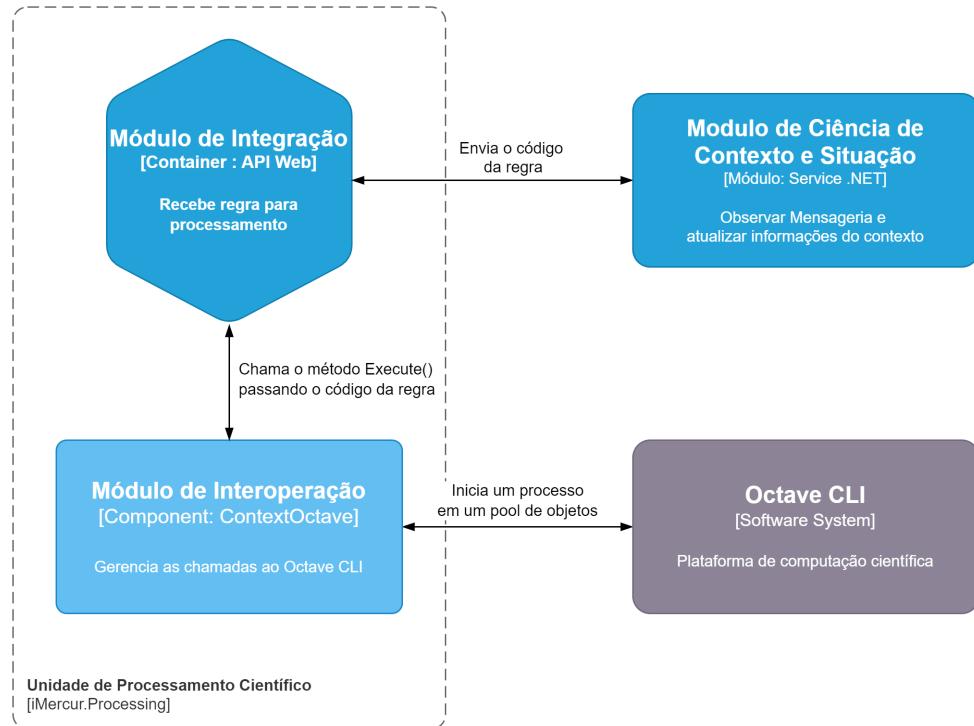


Figura 5.16 – Diagrama dos Módulos da Unidade de Processamento Científico

Fonte: Elaborado pelo autor

faz uma chamada ao método `Execute()` do Módulo de Interoperação, passando como parâmetro o código-fonte.

O Módulo de Interoperação implementa o padrão de projeto criacional *Object Pool*, o qual aloca um conjunto de objetos reutilizáveis prontos para o uso. Cada um desses objetos, quando solicitados para a execução, disparam um novo processo no sistema operacional do computador hospedeiro, o qual irá inicializar uma nova instância do Octave CLI.

Quando o resultado é retornado para o Módulo de Integração, a execução do processo que instancia o Octave CLI é finalizada e o objeto reutilizável é liberado para o uso por outra instância, caso necessário.

O Módulo de Integração retorna o resultado do processamento com o código HTTP 200 OK, que é a resposta de status de sucesso indicando que a requisição foi bem-sucedida para a Unidade de Inferência de Situação, a qual enviou a requisição. De outro modo, um código de status de resposta HTTP 400 Bad Request é retornado, indicando que houve um erro ao processar a requisição, neste caso, a mensagem do erro também é retornada. O protocolo HTTP é apresentado na seção 5.7 em mais detalhes. O diagrama de sequência da Figura 5.17 descreve a sequência de passos para o processamento do código-fonte pelo Octave.

O número de objetos é quem determina o número de instâncias do Octave em execução. Para a implementação aqui descrita, o número de objetos é configurado de acordo com a

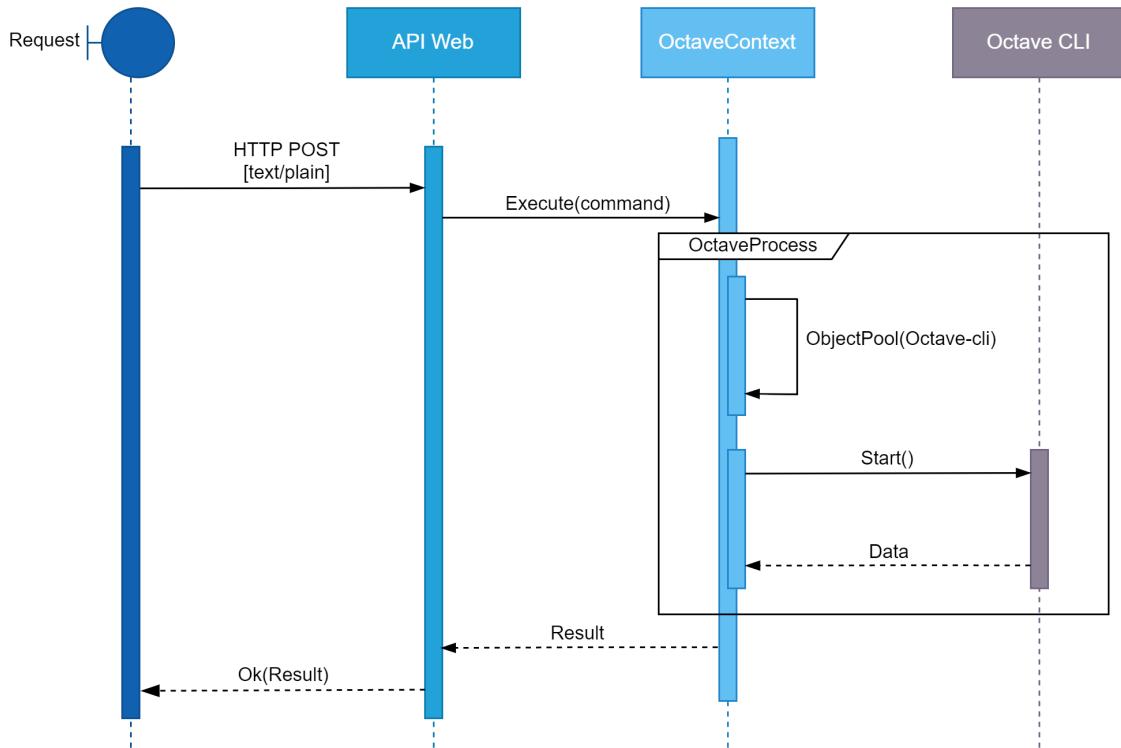


Figura 5.17 – Diagrama de Sequência da Unidade de Processamento Científico

Fonte: Elaborado pelo autor

quantidade de processadores do computador hospedeiro que executa o Octave. Esta abordagem exige que a Unidade de Processamento Científico execute no mesmo sistema operacional que está instalada a plataforma de computação científica.

Um caso típico de execução na Unidade de Processamento Científico é representado na Figura 5.18. Na implementação do Módulo de Integração foi incluída a biblioteca Swagger⁴ que implementa a especificação OpenAPI⁵. Esta especificação define uma descrição de interface padrão para APIs Web.

5.7 Tecnologias Selecionadas Para o Desenvolvimento

Esta seção resume a seleção de tecnologias empregadas no iMercur, a qual foi realizada considerando as tendências observadas na literatura da área e nos trabalhos relacionados elencados no capítulo 3. A seguir são apresentadas as principais tecnologias de *software* utilizadas na concepção do *Framework*.

⁴<https://swagger.io>

⁵<https://www.openapis.org>

POST /Execute

Request body **text/plain**

```
temperature = [10 23 30 25];      %dados carregados da última hora (chamada padrão de leitura)
average = mean(temperature)
```

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5150/Execute' \
  -H 'accept: */*' \
  -H 'Content-Type: text/plain' \
  -d 'temperature = [10 23 30 25];      %dados carregados da última hora (chamada padrão de leitura)
average = mean(temperature)'
```

Request URL

```
http://localhost:5150/Execute
```

Server response

Code	Details
200	Response body <code>average = 22</code>
	Response headers <code>content-type: text/plain; charset=utf-8</code> <code>date: Mon,17 Oct 2022 00:32:38 GMT</code> <code>server: Kestrel</code> <code>transfer-encoding: chunked</code>

Responses

Code	Description
200	Success

Figura 5.18 – Exploração da Unidade de Processamento Científico
Fonte: Elaborado pelo autor

Diagramas no Modelo C4

O Modelo C4⁶ foi criado como uma forma de ajudar as equipes de desenvolvimento de software a descrever e comunicar a arquitetura de software. Permite a construção de mapas de código, em vários níveis de detalhes, da mesma forma que é utilizado no Google Maps para

⁶<https://c4model.com>

aumentar e diminuir o zoom de uma área de interesse.

O modelo C4 é uma abordagem de “abstração” para diagramar a arquitetura de software, baseada em abstrações que refletem como arquitetos e desenvolvedores de software pensam e constroem software. O pequeno conjunto de abstrações e tipos de diagramas torna o modelo C4 fácil de aprender e usar. No modelo não é necessário utilizar todos os quatro níveis do diagrama, pode-se utilizar apenas aqueles que agregam valor como os diagramas *System Context* e *Container*. Para criar esses mapas de código, primeiro é necessário definir um conjunto comum de abstrações para criar uma linguagem para descrever a estrutura estática de um sistema de software. O modelo C4 considera as estruturas estáticas de um sistema de software em termos de contêineres, componentes e código. O Modelo pode ser descrito a partir dos seguintes tipos do modelo C4:

- usuário, representa um usuário interno do seu sistema de software, por exemplo, atores, papéis e pessoas;
- sistema de Software, um sistema de software é o mais alto nível de abstração e descreve o sistema como um todo. Isso inclui o sistema de software que você está modelando e os outros sistemas de software dos quais seu sistema de software depende, como unidades e módulos;
- contêiner, representa um aplicativo ou um armazenamento de dados. Um contêiner é algo que precisa estar em execução para que o sistema geral de software funcione;
- componente, é um agrupamento de funcionalidades relacionadas encapsuladas por trás de uma interface bem definida. Como uma coleção de classes de implementação por trás de uma interface. Aspectos como como esses componentes são empacotados (por exemplo, um componente versus muitos componentes por arquivo JAR, DLL, biblioteca compartilhada etc.) são uma preocupação separada e ortogonal;
- usuário externo, representa o usuário externo, quem irá operar os recursos de interface de usuário;
- sistema externo, representa os sistemas externos que vão interagir com o sistema proposto.

Um ponto importante a ser observado é que todos os componentes em um contêiner normalmente são executados no mesmo processo. No modelo C4, os componentes não são



Figura 5.19 – Legenda de Cores do Modelo C4

Fonte: Elaborado pelo autor

unidades implantáveis separadamente. Na Figura 5.19 são descritas as cores de acordo com os tipos do Modelo C4.

O Modelo C4 pode ser dividido nos níveis descritos a seguir, não sendo necessária a utilização de todos para a construção do projeto:

- nível 1, diagrama de contexto do sistema, permite observar o quadro geral do sistema como uma caixa no centro, cercada por seus usuários e outros sistemas com os quais interage;
- nível 2, diagrama de contêiner, mostra as principais opções de tecnologia e como os contêineres se comunicam entre si. É um diagrama simples e focado em tecnologia de alto nível;
- nível 3, diagrama de componentes, mostra como um contêiner é composto por vários outros componentes, o que são cada um desses componentes, suas responsabilidades e os detalhes de tecnologia/implementação;
- nível 4, código, é um nível de detalhe opcional e geralmente está disponível sob demanda em ferramentas. Idealmente, esse diagrama seria gerado automaticamente usando ferramentas (por exemplo, uma ferramenta de modelagem UML) e mostra apenas os atributos e métodos que permitem montar o projeto.

Hypertext Transfer Protocol - HTTP

O HTTP⁷ é um protocolo da camada de aplicação no modelo de conjunto de protocolos da Internet para sistemas de informação hipermídia distribuídos e colaborativos. O HTTP é a base da comunicação de dados para a *World Wide Web* (WWW), no qual documentos de hipertexto incluem hiperlinks para outros recursos que o usuário pode acessar facilmente, por exemplo, com um clique do mouse ou tocando na tela em um navegador da web.

JavaScript Object Notation - JSON

Para que as aplicações possam conversar entre si é utilizado um formato de dados padronizado, dentre os quais destaca-se o JSON⁸.

Consiste em um formato leve para trocas de dados especificado por Douglas Crockford no ano 2000. Para seres humanos é fácil de ler e escrever e às máquinas é fácil de interpretar e gerar.

Está baseado em um subconjunto da linguagem de programação JavaScript. JSON é em formato texto e completamente independente da linguagem de programação, pois usa convenções que são familiares às linguagens mais utilizadas . Ou seja, usa convenções familiares aos programadores da família C de linguagens, incluindo C, C++, C#, Java, JavaScript, Perl, Python e muitos outros.

Tais propriedades tornam o JSON uma linguagem ideal de intercâmbio de dados. Ele está constituído em duas estruturas:

- uma coleção de pares nome/valor, como um objeto, registro, estrutura, dicionário, tabela *hash*, lista de chaves ou matriz associativa;
- uma lista ordenada de valores, como uma matriz, vetor, lista ou sequência.

Essas são estruturas de dados universais. Praticamente todas as linguagens de programação modernas as suportam de uma forma ou de outra. Faz sentido que um formato de dados intercambiável com linguagens de programação também seja baseado nessas estruturas. No JSON elas podem assumir as formas de objeto, matriz, valor, sequência e/ou número.

⁷<https://www.ietf.org/rfc/rfc2616.txt>

⁸<https://www.json.org>

Linguagens utilizadas no ambiente Web

Os métodos pelos quais os computadores se comunicam, através do uso de linguagens de marcação e pacotes multimídia, são conhecidos como tecnologia da web. A seguir, são listadas as tecnologias utilizadas em ambiente web do iMercur.

- HTML: o acrônimo HTML⁹ vem do inglês e significa *Hypertext Markup Language* ou, em português, Linguagem de Marcação de Hipertexto. É uma linguagem de marcação utilizada na construção de páginas na Web. Documentos HTML podem ser interpretados por navegadores. Criada por Tim Berners-Lee (físico britânico), que também criou outros protocolos associados como o HTTP, a tecnologia é fruto da junção entre os padrões HyTime e SGML.
- JavaScript: é uma linguagem de programação interpretada de alto nível, caracterizada também, como dinâmica, fracamente tipada, *prototype-based* e multi-paradigma. Ideializada no meio da década de 90, mais precisamente em 1996, pelo programador Brendan Eich. Ele concebeu a linguagem quando ainda trabalhava na *Netscape Communications Corporation* e foi originalmente utilizada para funcionar no navegador *Netscape Navigator* com o objetivo de facilitar processos dentro de páginas web, tornando a programação de animações e alertas muito mais simples.
- C#: o C Sharp¹⁰ é uma linguagem de programação orientada a objeto e fortemente tipada que permite o desenvolvimento muitos tipos de aplicativos. Programas escritos em C# são executados no framework .NET, que é um *framework* composto por um sistema de execução virtual chamado CLR (*Common Language Runtime*) e um conjunto de bibliotecas de classes. O CLR é a implementação da Microsoft da CLI (*Common Language Infrastructure*), um padrão internacional. O código-fonte escrito em C# é compilado em uma IL (linguagem intermediária) que está em conformidade com a especificação da CLI. Quando o programa C# é executado, o *assembly* é carregado no CLR que executará a compilação JIT (*Just-In-Time*) para converter o código de IL em instruções nativas da máquina. O CLR também oferece serviços de coleta automática de lixo, tratamento de exceções, gerenciamento de recursos, entre outros.

⁹<https://www.w3.org>

¹⁰<https://learn.microsoft.com/pt-br/dotnet/csharp/>

Banco de Dados PostgreSQL

PostgreSQL¹¹ é um Sistema Gerenciador de Banco de Dados Objeto Relacional (SGBD) gratuito e de código aberto. Ele é um projeto *open-source*, coordenado pelo *PostgreSQL Global Development Group*, no qual as atividades do grupo são patrocinadas por diversas organizações de todo o mundo.

Possui mais de 30 anos de desenvolvimento ativo. Ganhou uma forte reputação de confiabilidade, integridade de dados e correção, robustez de recursos, extensibilidade e dedicação da comunidade de código aberto por trás do software, para fornecer consistentemente soluções inovadoras e de alto desempenho. Também é executado nos principais sistemas operacionais da atualidade.

O PostgreSQL possui muitos recursos destinados a ajudar os desenvolvedores a criar aplicativos e, os administradores a proteger a integridade dos dados e a criar ambientes tolerantes a falhas, além de ajudá-los a gerenciá-los, independentemente do tamanho do conjunto desses dados.

Também é altamente extensível, por exemplo, você pode definir seus próprios tipos de dados, criar funções personalizadas e até escrever código de diferentes linguagens de programação sem recompilar seu banco de dados.

O PostgreSQL oferece os recursos necessários para realizar a replicação entre servidores, trazendo a possibilidade de manter mais de um servidor com as informações armazenadas dos artefatos empregados no SANEP, tornando assim o Projeto como um todo mais escalável, frente a eventual necessidade de inclusão de novas demandas de sensoriamento e/ou atuação.

Serviço Apache Kafka

Kafka Shree et al. (2017) é um serviço de *log* distribuído, separável, redundante e persistente desenvolvido em linguagem Scala e baseado no protocolo TCP. No Kafka, as mensagens são classificadas em diferentes tópicos, cada tópico pode ser dividido em várias partições que são distribuídas e armazenadas em diferentes *brokers*.

Existem vários produtores, *brokers*, consumidores e um Zookeeper (FU; ZHANG; YU, 2020). O Kafka emprega uma arquitetura ponto a ponto em vez de uma arquitetura mestre-escravo, em que todos os *brokers* mantêm o mesmo status. Os produtores usam o modo *Push* para publicar mensagens para o *brokers*, e todos os consumidores em um grupo trabalham juntos

¹¹<http://postgresql.org>

para consumir todas as partições de um Tópico inscrito no modo *Pull*. Cada partição é uma fila de mensagens sequencial e imutável que pode ser adicionada continuamente.

As mensagens em uma partição são atribuídas a um número de sequência, denominado *offset*, que é único em cada partição. Em circunstâncias normais, o *offset* aumenta linearmente à medida que o consumidor consome a mensagem. Mas o deslocamento real é controlado pelo consumidor, que pode redefinir o deslocamento para reler a mensagem. Cada partição só pode ser digerida por um único consumidor em um grupo de consumidores e as mensagens em uma partição têm a garantia de serem ordenadas. Os produtores não precisam ser gerenciados pelo gerenciador de recursos de *cluster Zookeeper*. Isso porque esses produtores são transitórios e podem ser fechados a qualquer momento sem coordenação. Mas os consumidores e os *brokers* precisam ser gerenciados pelo Zookeeper para atingir o balanceamento de carga.

A alta disponibilidade e confiabilidade do Kafka é obtida por meio do mecanismo de *backup* (ou seja, cada partição tem várias replicações), e as réplicas são armazenadas em diferentes *brokers* e são sincronizadas. Há uma réplica líder e uma réplica seguidora múltipla, e todas as cargas de gravação / leitura são direcionadas à réplica líder. Quando o trabalhador onde a réplica do líder está localizada falhar, um novo líder será eleito entre os seguidores para continuar o serviço.

Além disso, o Kafka oferece suporte a mensagens transacionais e garantia de entrega. Ele oferece suporte à garantia de entrega de pelo menos uma vez por padrão e permite a garantia de no máximo uma vez configurando o produtor para enviar de forma assíncrona. A garantia exata também pode ser implementada, mas requer cooperação com o sistema de armazenamento de destino.

Framework Kafka Connect

Kafka Connect¹² é uma ferramenta para *streaming* de dados entre o Apache Kafka e outros sistemas de dados, o que permite definição rápida de conectores que movem grandes conjuntos de dados para o Kafka. O Kafka Connect pode acessar bancos de dados e coletar dados e disponibilizar em tópicos em um *broker* do servidor Kafka para processamento de fluxo. Também permite envio do *broker* para outros sistemas em diferentes protocolos de comunicação.

O Kafka Connect é um componente gratuito e de código aberto do Apache Kafka que funciona como um conjunto de conectores centralizados para a integração de forma simples

¹²<https://docs.confluent.io/platform/current/connect>

entre diferentes bancos de dados, armazenamentos de valores-chave, índices de pesquisa e sistemas de arquivos.

Framework KsqlDB

O ksqlDB¹³ é um banco de dados para aplicações de processamentos de fluxos sobre o Apache Kafka. Por meio de visualizações materializadas em tabelas, dos dados represados nos tópicos do Apache Kafka, os mesmos podem ser consultados como uma tabela de banco de dados por meio de consultas utilizando a linguagem SQL.

O serviço KsqlDB expõe um mecanismo de configuração por meio de uma API REST onde parâmetros predefinidos são passados para o serviço gerenciar fluxos de dados e acessar o *framework* Kafka Connect para configurar um conector, somente permitindo o tipo de conteúdo no formato JSON em requisições e respostas. A Tabela 5.3 apresenta os métodos HTTP disponibilizados pela API REST do serviço KsqlDB utilizados para configurar os canais do *Framework* iMercur.

Tabela 5.3 – Métodos da API REST para a Configuração dos Streams de Dados

Método	Descrição
POST /ksql	Criar <i>streams</i> de dados dos tópicos e conectores de fontes de dados externas
POST /query	Executa consultas em <i>streams</i>

A Figura 5.20 apresenta o JSON de configuração com os parâmetros para criar um conector do tipo *source*. Esta configuração conecta em um banco de dados PostgreSQL e carrega os dados especificados pela consulta SQL do parâmetro *Query* no tópico especificado pelo parâmetro *topic.prefix*.

A Figura 5.21 mostra o JSON de configuração de um fluxo de dados para a criação de um canal.

¹³<https://ksqldb.io>

```

1 POST /ksql HTTP/1.1
2 Accept: application/vnd.ksql.v1+json
3 Content-Type: application/vnd.ksql.v1+json
4
5 {
6     "ksql": "CREATE SOURCE CONNECTOR imercurdadoscontextuaisconnector WITH (
7         'connector.class' = 'io.confluent.connect.jdbc.JdbcSourceConnector',
8         'dialect.name' = 'PostgreSQLDatabaseDialect',
9         'validate.non.null' = 'false',
10        'timestamp.initial' = '0',
11        'timestamp.column.name' = 'datareg',
12        'mode' = 'timestamp',
13        'offset.storage.file.filename' = '/tmp/connect.offsets',
14        'tasks.max' = '1',
15        'query' = 'SELECT contextId, umidade, temperatura, datareg FROM
16            dadoscontextuais',
17        'table.types' = 'table',
18        'topic.prefix' = 'imercur.dadoscontextuais',
19        'name' = 'imercurdadoscontextuaisconnector',
20        'numeric.mapping' = 'best_fit',
21        'connection.url' = 'jdbc:postgresql://postgres:5432/postgres',
22        'connection.user' = 'postgres',
23        'connection.password' = 'postgres',
24        'errors.deadletterqueue.topic.name' = 'error.imercur.dadoscontextuais'
25    );"
}

```

Figura 5.20 – JSON de Configuração de um Conecotor
Fonte: Elaborado pelo autor

```

1 POST /ksql HTTP/1.1
2 Accept: application/vnd.ksql.v1+json
3 Content-Type: application/vnd.ksql.v1+json
4
5 {
6     "ksql": "CREATE STREAM dadoscontextuais (contextId INT, umidade DOUBLE,
7             temperatura DOUBLE, datareg TIMESTAMP) WITH
8             (kafka_topic='imercur.dadoscontextuais', value_format='json', partitions=1);"
9 }

```

Figura 5.21 – JSON de Configuração de um Fluxo de Dados
Fonte: Elaborado pelo autor

Plataforma Octave

O Octave¹⁴ foi escrito por John W. Eaton, destinada principalmente a cálculos numéricos. Ele fornece uma interface de linha de comando conveniente para resolver problemas

¹⁴<https://www.gnu.org>

lineares e não lineares numericamente e para realizar outros experimentos numéricos usando uma linguagem que é compatível com o Matlab. Também pode ser usado como uma linguagem orientada a lote.

O Octave possui ferramentas abrangentes para resolver problemas comuns de álgebra linear numérica, encontrar as raízes de equações não lineares, integrar funções ordinárias, manipular polinômios e integrar equações diferenciais ordinárias e diferenciais algébricas. É facilmente extensível e personalizável por meio de funções definidas pelo usuário escritas na própria linguagem do Octave ou usando módulos carregados dinamicamente escritos em C++, C, Fortran ou outras linguagens. Faz parte do projeto GNU, é um software livre sob os termos da licença GPL.

5.8 Considerações Finais do Capítulo

Neste capítulo é apresentada a contribuição central desta dissertação a plataforma de software em desenvolvimento no âmbito do convênio UCPel-SANEP. Sendo caracterizada a sua organização enquanto um *framework* que permite a integração com plataformas de computação científica, locais ou remotas, para o processamento de contextos complexos.

O iMercur por meio da Internet, oferece facilidades *on-line* para criação e manutenção de regras que possam ser manipulados diretamente pelo usuário final, que no caso do iMercur é a comunidade técnico-administrativa do SANEP, que apresenta forte alinhamento com a perspectiva crescente de emprego da IoT nas propostas computacionais modernas de operação distribuída.

6 IMERCUR: AVALIAÇÃO

Neste capítulo está registrada a avaliação do *framework* iMercur, tendo por base o método TAM (*Technology Acceptance Model*). A experiência do usuário com um sistema computacional pode ser avaliada de forma explícita, com o emprego de entrevistas e questionários. Essa estratégia pode refletir diretamente a usabilidade das aplicações, indicando a capacidade da arquitetura em atender os requisitos dos usuários, bem como pode possibilitar a verificação do correto funcionamento de seus módulos (KNAPPMEYER et al., 2013).

Modelo TAM

O *Technology Acceptance Model* (TAM) foi proposto por (DAVIS; BAGOZZI; WARSHAW, 1989) para criar modelos de aceitação em tecnologia da informação.

Teve origem em um contrato da *International Business Machines* (IBM) com o *Massachusetts Institute of Technology* (MIT), nos anos 1980, com o objetivo de avaliar o potencial de mercado para novos produtos da marca e possibilitar uma explicação dos determinantes da utilização de computadores.

Como afirmado pelos seus criadores, o modelo TAM tem a vantagem de ser específico para tecnologia da informação e tem uma forte base teórica, além do amplo apoio empírico, para sua aplicação.

O propósito essencial do modelo TAM, especificamente formatado para usuários de sistemas de informação, é prover uma base para mapear o impacto de fatores externos sobre aqueles internos ao indivíduo, como as crenças, atitudes e intenções de uso. Ele foi formulado com o objetivo de medir esses impactos, por meio da avaliação de algumas variáveis fundamentais, sugeridas por pesquisas anteriores, que tratam da aceitação de computadores de modo cognitivo e afetivo.

O TAM é uma teoria de sistemas de informação que modela como os usuários aceitam e usam uma tecnologia. O modelo sugere que, quando os usuários são apresentados a uma nova tecnologia, vários fatores influenciam sua decisão sobre como e quando eles a usarão, notadamente:

- Utilidade Percebida (*Perceived Usefulness - PU*) - É o grau em que uma pessoa acredita que, ao usar um determinado sistema, aumentaria seu desempenho no trabalho.
- Facilidade de Uso Percebida (*Perceived Ease of Use - PEOU*) - É o grau em que uma

pessoa acredita, que ao usar um determinado sistema, estaria livre de esforço (DAVIS; BAGOZZI; WARSHAW, 1989).

A Utilidade Percebida pode ser definida como uma probabilidade subjetiva percebida pelo usuário, de que determinada tecnologia possa melhorar a performance em relação ao objeto de uso, geralmente, um sistema de informação (KARAHANNA; STRAUB; CHERVANY, 1999). Moore and Benbasat (1991) definem a utilidade percebida como uma vantagem relativa, ou seja, o grau em que uma inovação tecnológica é percebida, como superior, em comparação à tecnologia antiga que está sendo substituída.

Já a Facilidade de Uso Percebida se refere às expectativas do indivíduo, em termos de esforço físico ou mental, para o uso de determinado sistema ou tecnologia. A Figura 6.1 sugere que os indivíduos usarão uma determinada tecnologia se acreditarem que esse uso fornecerá resultados positivos, focalizando-se na facilidade de uso percebida (*Perceived Ease of Use*) e na utilidade percebida (*Perceived Usefulness*).

De acordo com o modelo, o uso dos sistemas de informação seria determinado pela intenção de uso que o indivíduo apresenta. Essa, por sua vez, seria determinada em conjunto pela atitude de uso do indivíduo com relação ao uso real do sistema e pela utilidade percebida, cada uma exercendo um peso relativo.

Essa relação, entre atitude e intenção, sugere que as pessoas formam intenções para desempenhar ações para as quais tenham um sentimento positivo.

Já a relação entre utilidade percebida e intenção de uso, é baseada no entendimento de que, dentro de um contexto organizacional, as pessoas formam intenções com relação a comportamentos que elas acreditam que aumentarão seu desempenho no trabalho.

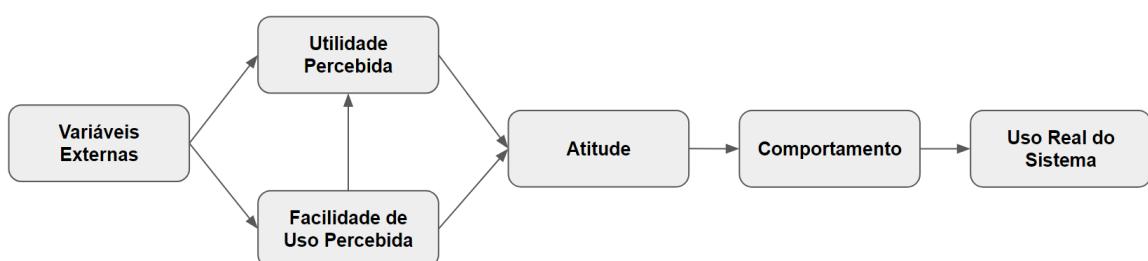


Figura 6.1 – Visão Geral do Modelo TAM
Fonte: Adaptada de (DAVIS; BAGOZZI; WARSHAW, 1989).

Aplicação do modelo TAM no iMercur

A coleta de dados, em uma pesquisa que segue a proposta do modelo TAM, pode ser realizada através da aplicação de instrumentos contendo afirmações fechadas, com as opções de resposta utilizando a escala Likert (SOUTH et al., 2022), considerando uma escala de mensuração com cinco categorias, variando de "Discordo totalmente" a "Concordo totalmente". Ao responderem a um instrumento baseado nesta escala, os respondentes especificam seu nível de concordância com a afirmação.

Deste modo, a escala Likert irá medir as atitudes e o grau de conformidade do respondente com uma afirmação. Ao contrário de responder apenas "sim" ou "não", o pesquisado mostra mais especificamente o quanto ele concorda ou discorda de uma atitude ou ação, ou o quanto ele está satisfeito ou insatisfeito. Nessa perspectiva, o modelo TAM foca no porquê dos usuários aceitarem ou rejeitarem uma tecnologia e como melhorar a sua aceitação.

A consistência de um instrumento de coleta de dados refere-se ao grau com que os itens do mesmo estão correlacionados entre si e com o resultado geral da pesquisa, o que representa uma mensuração da confiabilidade do instrumento utilizado. Um dos procedimentos estatísticos mais utilizados para mensuração da consistência é o coeficiente, apresentado por Lee J. Cronbach em 1951 (HORA; MONTEIRO; ARICA, 2010), conforme a Equação 6.1. Neste sentido, um instrumento de sondagem é considerado confiável quando o alfa de Cronbach apresentar um valor acima de 0,7.

$$\alpha = \frac{k}{k-1} \left[1 - \frac{\sum V_i}{V_t} \right] \quad (6.1)$$

Sendo:

- K = número de itens do formulário;
- Vi = variância de cada item;
- Vt = variância do somatório dos totais de cada participante.

Para o cálculo do Alfa de Cronbach, normalmente, os itens do questionário são transformados de uma escala nominal para uma numérica (HORA; MONTEIRO; ARICA, 2010), segundo a tabela 6.1.

Nesta avaliação, foi feita a apresentação e exploração das funcionalidades do *Framework iMercur* junto aos profissionais da área de Tecnologia da Informação de uma empresa de grande

Tabela 6.1 – Valores Numéricos para a Escala Likert

Item	Valor
Discordo totalmente	0
Discordo parcialmente	0,25
Indiferente	0,50
Concordo parcialmente	0,75
Concordo totalmente	1

porte. Num primeiro momento, se torna oportuna a apresentação do *Framework* para um grupo de profissionais do mercado de Tecnologia da Informação com bom nível de especialização. O principal critério para esta abordagem, é a potencial contribuição decorrente das observações realizadas, seja quanto aos mecanismos utilizados na arquitetura do *Framework* desenvolvido, como também, dos recursos da interface de usuário.

O grupo que participou do instrumento de avaliação foi composto por doze colaboradores no total: seis Engenheiros de Software, dois de Engenheiros de Qualidade, dois de Arquitetos de Software e dois da área de Gestão dos times de desenvolvimento de produtos. Os colaboradores atendem as áreas da empresa denominadas Inteligência Comercial e Produtos Digitais.

Efetivadas as melhorias no iMercur decorrentes das contribuições deste grupo com expertise em Tecnologia de Informação, um novo Instrumento de Avaliação TAM será aplicado junto ao corpo técnico-administrativo do SANEP.

O instrumento de coleta de dados considerou todos os níveis de afirmação da Escala Likert. Os entrevistados especificam seu nível de concordância com uma afirmação proposta em um item (assertiva atitudinal), mediante um critério que pode ser objetivo ou subjetivo. Assim, se mede o nível de concordância ou não concordância à afirmação proposta.

Tabela 6.2 – Afirmativas do Instrumento de Avaliação TAM

Construto	Afirmativa
Facilidade de uso percebida	1 - As funcionalidades de gerência do iMercur são fáceis de entender. 2 - Os diferentes parâmetros a serem definidos nas funcionalidades do iMercur são claros e objetivos. 3 - Com pouco esforço consigo definir regras contextuais no iMercur explorando o Octave.
Utilidade Percebida	4 - As funcionalidades oferecidas pelo iMercur são relevantes para o SANEP. 5 - O iMercur provê informações que podem auxiliar na gerência distribuída dos artefatos do SANEP. 6 - O iMercur pode minimizar riscos na operação dos artefatos do SANEP.

Considerando as respostas obtidas no instrumento de coleta de dados e convertendo os itens da Escala Likert para os valores numéricos, sugeridos na tabela 6.1, foram calculadas as variâncias individuais de cada item, bem como a soma da pontuação de cada um deles, que foi utilizada para o cálculo da variância total. Desta forma, através da Equação 6.1 foi obtido o valor do Alfa de Cronbach.

A análise dos dados coletados buscou a identificação do nível de aprovação das funcionalidades do iMercur, quanto a facilidade de uso e a utilidade percebida pelos participantes da pesquisa. Para esta análise foram considerados os resultados obtidos em cada item da Escala Likert.

O Instrumento respondido, apresentado na Tabela 6.2, foi aplicado no mês Dezembro de 2022, em uma reunião de 1h30m de duração, na qual foram abordados o Modelo TAM e as funcionalidades do *Framework* iMercur. Este instrumento, em consonância com o Método TAM, teve por foco a apreciação das funcionalidades do iMercur, quanto a facilidade de uso e a utilidade percebida pelos participantes da pesquisa.

Considerando as respostas obtidas no questionário e convertendo os itens da Escala Likert para os valores numéricos, sugeridos na Tabela 6.1, foi gerada a tabela apresentada na Tabela 6.2. Foram calculadas as variâncias individuais de cada afirmação, bem como a variância dos totais da pontuação de cada usuário, que foram utilizadas para o cálculo da variância total. Os resultados obtidos são bastante promissores, apontando ótimos escores tanto para a facilidade de uso, como para a Utilidade Percebida do iMercur.

Considerando os dados tabulados da Tabela 6.2, para o cálculo do Alfa de Cronbach, segundo a Equação 6.1, foram calculados: (i) o somatório das variâncias individuais dos itens ($\sum V_i = 0,03$); e, (ii) a variâncias dos totais decorrentes das pontuações de cada um dos respondentes ($V_t = 0,077690972$). Utilizando o número de afirmações ($k = 6$), como mostra a Equação 6.2, foi obtido o valor do Alfa de Cronbach igual a 0,72.

$$\alpha = \frac{6}{6 - 1} \left[1 - \frac{0,03}{0,077690972} \right] \quad (6.2)$$

Segundo a literatura, os valores aceitáveis do Alfa de Cronbach variam de 0,70 a 0,95 (BLAND; ALTMAN, 1997) (DEVELLIS, 2016). Um baixo valor de alfa pode ser devido a um baixo número de perguntas, baixa inter-relação entre itens ou construções heterogêneas. Se alfa for muito alto, pode sugerir que alguns itens sejam redundantes, neste sentido é recomendado um valor alfa máximo de 0,90 (STREINER, 2003).

Usuários	Afirmações						Totais
	1	2	3	4	5	6	
1	1,00	1,00	1,00	1,00	1,00	1,00	6,00
2	1,00	1,00	1,00	1,00	1,00	1,00	6,00
3	0,75	1,00	0,75	1,00	1,00	0,75	5,25
4	1,00	1,00	1,00	1,00	1,00	1,00	6,00
5	1,00	1,00	1,00	1,00	1,00	1,00	6,00
6	1,00	1,00	1,00	1,00	1,00	1,00	6,00
7	0,75	1,00	0,75	1,00	1,00	0,75	5,25
8	1,00	1,00	1,00	1,00	1,00	1,00	6,00
9	1,00	1,00	1,00	0,75	1,00	1,00	5,75
10	1,00	1,00	1,00	1,00	1,00	1,00	6,00
11	1,00	1,00	1,00	1,00	1,00	1,00	6,00
12	1,00	1,00	1,00	1,00	1,00	1,00	6,00
Variância	0,0086806	0	0,0086806	0,00477431	0	0,0086806	0,077691
Média	0,95833	1,00	0,95833	0,97917	1,00	0,95833	
Soma das Variâncias das Afirmações							0,031
Média dos Totais dos Usuários							5,85
Variâncias dos Totais dos Usuários							0,077691
Alpha de Cronbach							0,72

Figura 6.2 – Análise Descritiva das Respostas do Instrumento de Avaliação TAM

Fonte: elaborado pelo autor

6.1 Considerações Finais do Capítulo

Este capítulo apresenta o instrumento de avaliação empregando o método TAM para mensurar a percepção do usuário diante dos recursos oferecidos pelo *Framework* iMercur. As afirmações que integram o instrumento têm alinhamento com as funcionalidades providas pelo *Framework*.

Baseado no valor obtido para o Alfa de Cronbach igual a 0,72 pode-se considerar que a avaliação feita pelo grupo de profissionais da empresa de Tecnologia da Informação, de acordo com o método proposto, possui confiabilidade desejada e com isso se mostra um instrumento eficaz ao que se propõe. Nesse sentido, feita a apresentação e exploração das funcionalidades, os resultados obtidos se mostram promissores quanto à utilidade e facilidade de uso do *Framework* junto ao usuário final.

7 CONSIDERAÇÕES FINAIS

Este capítulo registra as principais conclusões decorrentes do trabalho desenvolvido nesta Dissertação de Mestrado, registra a publicação realizada bem como apresenta a perspectiva de continuidade da pesquisa com a indicação dos trabalhos futuros entendidos como prioritários.

7.1 Principais Conclusões

A revisão de literatura na área aponta que a IoT, em decorrência da oferta de tecnologias a custo cada vez menores e em regimes de confiabilidade crescentes, apresenta soluções para o mundo real, empregando a combinação de sensores e atuadores, capazes de fornecer e receber informações digitalizadas e colocá-las em produção empregando arquiteturas bidirecionais para trocas de dados, e deverá ser cada vez mais adotada em diferentes soluções distribuídas de monitoramento e automação.

Reforçando esta tendência, também se observa um constante crescimento do número de dispositivos conectados na Internet, com a estimativa de que já estamos próximos de ter 50 bilhões de artefatos conectados. Com este nível de escalabilidade, a oferta de serviços tende a aumentar, gerando um crescimento recorrente da área em termos de soluções de *software* que operem de forma autônoma, explorando Ciência de Contexto e/ou Situação na sua gerência.

Por sua vez, no que tange à interoperabilidade entre os servidores que compõem o Serviço para Ciência de Situação, o iMercur contempla uma abordagem compatível com a expectativa de operação necessária ao SANEP, utilizando protocolos padrões da Internet para as comunicações e plataformas *open-source* para concepção de sua arquitetura de *software*. Estes aspectos potencializam a possibilidade de que o iMercur possa ser expandido e/ou integrado às novas plataformas a serem desenvolvidas na continuidade das pesquisas envolvendo o SANEP, com uma boa perspectiva de que possam ser mantidos os padrões de manutenibilidade e escalabilidade da solução oferecida ao SANEP como um todo.

O *Framework* compartilha as informações sensoriadas de maneira visual, tanto para o público em geral, quanto para os colaboradores do SANEP possui uma interface de usuário para ser de uso intuitivo, facilitando a manipulação por parte de usuários com diferentes perfis de interesse.

O iMercur, além de informações textuais, possibilita o emprego de gráficos para caracterizar o comportamento dos diferentes sensores previstos. Outrossim, o iMercur disponibiliza

um ambiente para o usuário final, que propicia a criação de regras e sua associação aos sensores que as mesmas irão tratar. Estas regras podem ser disparadas de forma assíncrona, a qualquer momento, pelos usuários, ou a partir da coleta de informações provenientes de sensores conectados às mesmas.

As contribuições decorrentes da utilização do iMercur no ambiente para o qual está sendo idealizado, foram confirmadas com o uso do método TAM, o qual permitiu comprovar a facilidade de uso do *framework*, ou seja, garantindo que as expectativas das facilidades previstas fossem validadas, caracterizando assim a aceitação do iMercur perante seu público de interesse.

7.2 Publicações

Nesta seção estão registradas as publicações como primeiro autor relacionadas aos tópicos de pesquisa da abordagem.

Apresentações e Publicações

- iMercur Framework: Exploring End User Contextual Processing Platforms in SANEP Artifact Situation Awareness. IEEE America Latina. 2022;
- Avaliação de Situações no SANEP-I²MF Explorando Múltiplos Processadores Contextuais. Salão Universitário da UCPel. 2021;
- Framework iMercur: Uma Contribuição à Ciência de Contexto no SANEP-I²MF. Semana Tecnologia 2020 - Desafios e Oportunidades. Universidade Católica de Pelotas.

Publicações Previstas:

- SBAI - Simpósio Brasileiro de Automação Inteligente
- SEMISH - Seminário Integrado de Software e Hardware
- SBrT - Simpósio Brasileiro de Telecomunicações e Processamento de Sinais

7.3 Trabalhos Futuros

Dentre as possíveis alternativas para a continuidade da pesquisa desenvolvida para concepção da abordagem, destacam-se as frentes de trabalho elencadas a seguir:

- explorar o emprego de serviços em nuvem para disponibilização do iMercur. A expectativa com isto é potencializar os recursos de escalabilidade e disponibilidade para os usuários finais;
- estender a conectividade do iMercur para outros protocolos de comunicação com a finalidade de flexibilizar a criação de canais para diferentes repositórios contextuais;
- integrar o *Framework* iMercur com outras plataformas de computação científica, que possam atender demandas específicas de processamento contextual;
- implementar uma aplicação *front-end* mais robusta, garantindo que os novos recursos incorporados ao *Framework* viabilizem a usabilidade do usuário final.

REFERÊNCIAS

- ABADI, M. L. et al. Predictive classification of water consumption time series using non-homogeneous markov models. In: **2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)**. [S.l.: s.n.], 2017. p. 323–331.
- ADAMS, M. J.; TENNEY, Y. J.; PEW, R. W. Situation awareness and the cognitive management of complex systems. In: **Situational Awareness**. [S.l.]: Routledge, 2017. p. 43–62.
- AL-FUQAHA, A. et al. Internet of things: A survey on enabling technologies, protocols, and applications. **IEEE Communications Surveys Tutorials**, v. 17, n. 4, p. 2347–2376, 2015.
- ALEGRE, U.; AUGUSTO, J. C.; CLARK, T. Engineering context-aware systems and applications: A survey. **The Journal of Systems and Software**, Elsevier Inc., v. 117, p. 55–83, 2016.
- ALMEIDA, R. B. et al. Um estudo sobre consciência de situação. **Cadernos de Informática**, Porto Alegre-RS, v. 7, n. 1, p. 44–67, 2013.
- ALVI, M. S. Q. et al. Dynamic behavioural modeling, simulation and analysis of household water consumption in an urban area: a hybrid approach. In: **2018 Winter Simulation Conference (WSC)**. [S.l.: s.n.], 2018. p. 2411–2422.
- ANEEL. **No Dia Mundial da Água, Aneel publica infográfico sobre hidrelétricas no Brasil - Sala DE Imprensa - ANEEL**. 2021. Available from Internet: <<https://bit.ly/3cPPeYs>>.
- ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer Networks**, p. 2787–2805, 10 2010.
- BIBRI, S. E. **The Human Face of Ambient Intelligence: Cognitive, Emotional, Affective, Behavioral and Conversational Aspects**. 1st. ed. [S.l.]: Atlantis Publishing Corporation, 2015. ISBN 9462391297.
- BLAND, J. M.; ALTMAN, D. G. Statistics notes: Cronbach's alpha. **Bmj**, British Medical Journal Publishing Group, v. 314, n. 7080, p. 572, 1997.
- CAPT, T. et al. Urban water demand: Statistical optimization approach to modeling daily demand. **Journal of Water Resources Planning and Management**, v. 147, n. 2, p. 04020105, 2021.
- CLOSS, L. Sac: Situation-aware care:“um modelo de monitoramento de pacientes utilizando ciência de situação”. Universidade do Vale do Rio dos Sinos, 2017.
- DAVIS, F. D.; BAGOZZI, R. P.; WARSHAW, P. R. User acceptance of computer technology: a comparison of two theoretical models. **Management science**, INFORMS, v. 35, n. 8, p. 982–1003, 1989.
- DEVELLIS, R. F. **Scale development: Theory and applications**. [S.l.]: Sage publications, 2016.
- DEY, A. K. Understanding and using context. Springer-Verlag, Berlin, Heidelberg, v. 5, n. 1, p. 4–7, jan. 2001. ISSN 1617-4909.

- DURSO, F. T.; GRONLUND, S. D. Situation awareness. **Handbook of applied cognition**, p. 283–314, 1999.
- ENDSLEY, M. R. Situation awareness misconceptions and misunderstandings. **Journal of Cognitive Engineering and Decision Making**, Sage Publications Sage CA: Los Angeles, CA, v. 9, n. 1, p. 4–32, 2015.
- FU, G.; ZHANG, Y.; YU, G. A fair comparison of message queuing systems. **IEEE Access**, PP, p. 1–1, 12 2020.
- GANDODHAR, P. S.; CHAware, S. M. Context aware computing systems: A survey. In: **2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on**. [S.l.: s.n.], 2018. p. 605–608.
- HEJAZI, H. et al. Survey of platforms for massive iot. In: **2018 IEEE International Conference on Future IoT Technologies (Future IoT)**. [S.l.: s.n.], 2018. p. 1–8.
- HOHPE, G.; WOOLF, B. **Enterprise integration patterns: Designing, building, and deploying messaging solutions**. [S.l.]: Addison-Wesley Professional, 2004.
- HORA, H. R. M. da; MONTEIRO, G. T. R.; ARICA, J. Confiabilidade em questionários para qualidade: um estudo com o coeficiente alfa de cronbach. **Produto & Produção**, v. 11, n. 2, p. 85–103, 2010.
- HU, S. et al. An innovative hourly water demand forecasting preprocessing framework with local outlier correction and adaptive decomposition techniques. **Water**, v. 13, n. 5, 2021.
- KARAHANNA, E.; STRAUB, D. W.; CHERVANY, N. L. Information technology adoption across time: a cross-sectional comparison of pre-adoption and post-adoption beliefs. **MIS quarterly**, JSTOR, p. 183–213, 1999.
- KARAMAN, Ç. Ç.; YALIMAN, S.; OTO, S. A. Event detection from social media: 5w1h analysis on big data. In: **IEEE. 2017 25th Signal Processing and Communications Applications Conference (SIU)**. [S.l.], 2017. p. 1–4.
- KARCHOUD, R. et al. All for one and one for all: Dynamic injection of situations in a generic context-aware application. **Procedia Computer Science**, Amsterdam, The Netherlands, v. 113, p. 17 – 24, 2017. ISSN 1877-0509. The 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) / Affiliated Workshops.
- KNAPPMEYER, M. et al. Survey of Context Provisioning Middleware. **IEEE Communications Surveys & Tutorials**, v. 15, n. 3, p. 1492–1519, jan 2013.
- KRUMM, J. et al. **Ubiquitous Computing Fundamentals**. New York: Chapman {&} Hall/CRC, 2010.
- LAZIDIS, A.; TSAKOS, K.; PETRAKIS, E. G. Publish–subscribe approaches for the iot and the cloud: Functional and performance evaluation of open-source systems. **Internet of Things**, v. 19, p. 100538, 2022. ISSN 2542-6605. Available from Internet: <<https://www.sciencedirect.com/science/article/pii/S2542660522000403>>.

- LEE, J. D.; KIRLIK, A.; DAINOFF, M. J. **The Oxford handbook of cognitive engineering.** [S.l.]: Oxford University Press, 2013.
- LI, X. et al. Context aware middleware architectures: survey and challenges. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 15, n. 8, p. 20570–20607, 2015.
- LONDERO, J. M. SANEP-I²MF: Uma abordagem na IoT explorando ciência de situação direcionada aos artefatos de distribuição de água potável do SANEP. 2021.
- LOPES, J. et al. Uma abordagem autonômica baseada em regras para consciência de situação na computação ubíqua. 2013.
- LOPES, J. L. et al. A middleware architecture for dynamic adaptation in ubiquitous computing. **Journal of Universal Computer Science**, v. 20, n. 9, p. 1327–1351, 2014. ISSN 0948695X.
- LOPES, J. L. B. Uma arquitetura para provimento de ciência de situação direcionada às aplicações ubíquas na infraestrutura da internet das coisas. 2016.
- MACHADO, R. S. et al. State of the art in hybrid strategies for context reasoning: A systematic literature review. **Information and Software Technology**, Elsevier, 2019.
- MARQUES, G.; GARCIA, N.; POMBO, N. A survey on iot: Architectures, elements, applications, qos, platforms and security concepts. In: _____. [S.l.: s.n.], 2017. v. 22. ISBN 978-3-319-45143-5.
- MIRANZI, M. A. S. et al. Compreendendo a história da saúde pública de 1870-1990. **Saúde Coletiva**, 2010. ISSN 1806-3365. Available from Internet: <<https://www.redalyc.org/articulo.oa?id=84213511007>>.
- MOORE, G. C.; BENBASAT, I. Development of an instrument to measure the perceptions of adopting an information technology innovation. **Information systems research**, INFORMS, v. 2, n. 3, p. 192–222, 1991.
- NIE, X. et al. Big data analytics and iot in operation safety management in under water management. **Computer Communications**, v. 154, p. 188–196, 2020. ISSN 0140-3664. Available from Internet: <<https://www.sciencedirect.com/science/article/pii/S0140366419320560>>.
- NUNES, L.; DIAZ, R. A evolução do saneamento básico na história e o debate de sua privatização no brasil. **Revista de Direito da Faculdade Guanambi**, v. 7, n. 02, p. e292, dez. 2020. Available from Internet: <<http://revistas.faculdadeguanambi.edu.br/index.php/Revistadedireito/article/view/292>>.
- PERERA, C. **Sensing as a Service for Internet of Things: A roadmap.** [S.l.]: Leanpub Publishers, 2017.
- PERERA, C. et al. Context aware computing for the internet of things: A survey. **IEEE Communications Surveys and Tutorials**, v. 16, n. 1, p. 414–454, jan 2014.
- QIU, T. et al. How can heterogeneous internet of things build our future: A survey. **IEEE Communications Surveys & Tutorials**, IEEE, v. 20, n. 3, p. 2011–2027, 2018.

- RAYES, A.; SAMER, S.** **Internet of Things From Hype to Reality: The Road to Digitization.** 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2016. ISBN 3319448587.
- RIBEIRO, J. W.; ROOKE, J. M. S.** Saneamento básico e sua relação com o meio ambiente e a saúde pública. **Monografia de Especialização em Análise Ambiental, Universidade Federal de Juiz de Fora, Minas Gerais, Brasil.** 36p, 2010.
- ROUTH, K.; PAL, T.** A survey on technological, business and societal aspects of internet of things by q3, 2017. In: **2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU).** [S.l.: s.n.], 2018. p. 1–4.
- SHREE, R. et al.** Kafka: The modern platform for data management and analysis in big data domain. In: **2017 2nd International Conference on Telecommunication and Networks (TEL-NET).** [S.l.: s.n.], 2017. p. 1–5.
- SMITH, K.; HANCOCK, P. A.** Situation awareness is adaptive, externally directed consciousness. **Human factors**, SAGE Publications Sage CA: Los Angeles, CA, v. 37, n. 1, p. 137–148, 1995.
- SOUTH, L. et al.** Effective use of likert scales in visualization evaluations: a systematic review. In: **WILEY ONLINE LIBRARY. Computer Graphics Forum.** [S.l.], 2022. v. 41, n. 3, p. 43–55.
- SOUZA, R. S. de et al.** Continuous monitoring seed testing equipments using internet of things. **Computers and Electronics in Agriculture**, v. 158, p. 122 – 132, 2019. ISSN 0168-1699. Available from Internet: <<http://www.sciencedirect.com/science/article/pii/S0168169917309158>>.
- STREINER, D. L.** Starting at the beginning: an introduction to coefficient alpha and internal consistency. **Journal of personality assessment**, Taylor & Francis, v. 80, n. 1, p. 99–103, 2003.
- TEMDEE, P.; PRASAD, R.** **Context-Aware Communication and Computing: Applications for Smart Environment.** Switzerland: Springer International Publishing, 2018. (Springer Series in Wireless Technology, 1).
- WU, H.** Research proposal: Reliability evaluation of the apache kafka streaming system. In: **2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW).** [S.l.: s.n.], 2019. p. 112–113.