UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIA

MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

ÍTALO LINHARES DE ARAÚJO

ÁGAPE: FRAMEWORK PARA DETECÇÃO DE QUEDAS DE IDOSOS E SUAS CAUSAS COM DISPOSITIVOS VESTÍVEIS

FORTALEZA

2022

ÍTALO LINHARES DE ARAÚJO

ÁGAPE: FRAMEWORK PARA DETECÇÃO DE QUEDAS DE IDOSOS E SUAS CAUSAS COM DISPOSITIVOS VESTÍVEIS

Tese apresentada ao Mestrado e Doutorado em Ciência da Computação do Centro de Ciência da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Engenharia de Software.

Orientadora: Profa. Dra. Rossana Maria de Castro Andrade.

Coorientador: Prof. Dr. Paulo Armando Cavalcante Aguilar.

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A692Á Araújo, Ítalo Linhares de.
Ágape: Framework para Detecção de Quedas de Idosos e Suas Causas com Dispositivos Vestíveis / Ítalo Linhares de Araújo. – 2022.
143 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2022.
Orientação: Profa. Dra. Rossana Maria de Castro Andrade.
Coorientação: Prof. Dr. Paulo Armando Cavalcante Aguilar.

1. Arcabouço. 2. Internet das Coisas de Saúde. 3. Detecção de Quedas. I. Título.

CDD 005

ÍTALO LINHARES DE ARAÚJO

ÁGAPE: FRAMEWORK PARA DETECÇÃO DE QUEDAS DE IDOSOS E SUAS CAUSAS COM DISPOSITIVOS VESTÍVEIS

Tese apresentada ao Mestrado e Doutorado em Ciência da Computação do Centro de Ciência da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Engenharia de Software.

Aprovada em: 26/04/2022

BANCA EXAMINADORA

Profa. Dra. Rossana Maria de Castro Andrade (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Armando Cavalcante Aguilar (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Miguel Franklin de Castro
Universidade Federal do Ceará (UFC)

Prof. Dr. Danielo Gonçalves Gomes
Universidade Federal do Ceará (UFC)

Profa. Dra. Janaína Fonseca Victor Coutinho
Universidade Federal do Ceará (UFC)

“Ainda que eu tenha o dom de profecia, saiba todos os mistérios e todo o conhecimento e tenha uma fé capaz de mover montanhas, se não tiver amor, nada serei.”

(Paulo de Tarso)

RESUMO

A Internet das Coisas (IoT, do inglês *Internet of Things*) permite a comunicação de objetos do dia-a-dia com a Internet e, com isso, mais serviços úteis são providos aos usuários. Esses serviços podem estar inseridos em vários domínios, como na saúde (IoHT, do inglês, *Internet of Health Things*), no qual dispositivos inteligentes podem monitorar os pacientes a fim de coletar dados e identificar situações anormais, indicando a ocorrência de situações de emergência. Uma das situações que pode ser monitorada é a queda de idosos, um problema mundial que pode ter consequências graves, inclusive óbito. Diante desse cenário, é importante detectar uma queda mais rapidamente para que as sequelas sejam minimizadas. Contudo, detectar apenas a queda não é suficiente, pois elas podem ter origem em problemas de saúde que também precisam ser tratados, como problemas de pressão arterial e diabetes, sendo chamados de causas intrínsecas, e que podem ser monitorados por dispositivos vestíveis de IoT. Sendo assim, uma solução com múltiplos dispositivos que detecta eventos de quedas e possíveis fatores causadores pode auxiliar a minimizar as sequelas. Dentro desse escopo, foram encontrados trabalhos na literatura que visavam a detecção de quedas, porém sem relacionar com outros aspectos da saúde do idoso que podem resultar nas quedas. Além disso, verificou-se a ausência de trabalhos com reuso de software para facilitar o desenvolvimento de aplicações para detecção dessas quedas considerando multidispositivos e que também relacionassem a queda com a sua provável causa intrínseca. Então, este trabalho propõe um *framework*, artefato de reuso que contém códigos comuns a aplicações de um domínio, chamado Ágape, cujo objetivo é facilitar o desenvolvimento de aplicações IoHT para detecção de quedas com foco em dispositivos IoT vestíveis, relacionando as quedas com suas possíveis causas. O Ágape foi avaliado em relação ao tempo necessário para configurá-lo, o tempo necessário para desenvolver uma aplicação com o seu suporte, além de uma avaliação baseada no Technology Acceptance Model (TAM), a fim de identificar a percepção de utilidade e de facilidade de uso do Ágape pelos participantes. Os resultados indicam que o *framework* reduz o tempo de desenvolvimento de aplicações para detecção de quedas e suas possíveis causas em 98,09%. Além disso, os participantes indicaram que conseguiram perceber a utilidade do Ágape, bem como acharam fácil utilizá-lo.

Palavras-chave: arcabouço; internet das coisas de saúde; detecção de quedas.

ABSTRACT

The Internet of Things (IoT) allows everyday objects to communicate with the Internet and, with that, more useful services are provided to users. These services can be embedded in various domains, such as health with Internet of Health Things (IoHT), in which smart devices can monitor patients in order to collect data and identify abnormal situations, indicating the occurrence of emergency situations. One of the situations that can be monitored is an elderly fall, a worldwide problem that can have serious consequences, including death. Given this scenario, it is important to detect a fall more quickly so that the sequelae are minimized. However, just detecting the fall is not enough, as they can be originated from health problems that also need to be treated, such as blood pressure problems and diabetes, called intrinsic causes, and that can be monitored by IoT wearable devices. Therefore, a multi-device solution that detects fall events and possible causative factors can help minimize sequelae. Within this scope, studies were found in the literature that aimed to detect falls, but without relating to other aspects of the health of the elderly that can result in falls. In addition, there was a lack of works with software reuse to facilitate the development of applications for the detection of falls that consider multiple devices and that also relate the fall to its probable intrinsic cause. So, this work proposes a framework, reuse artifact that contains codes common to applications in a domain, called Agape, whose objective is to facilitate the development of IoHT applications for fall detection focusing on wearable IoT devices, relating falls with their possible causes. Agape was evaluated in terms of the time required to configure it, the time required to develop an application with its support, in addition to an evaluation based on the Technology Acceptance Model (TAM), in order to identify the perception of usefulness and ease of use of Agape by the participants. The results indicate that the framework reduces the development time of applications to detect falls and their possible causes by 98.09%. In addition, participants indicated that they were able to perceive the usefulness of Ágape as well as found it easy to use.

Keywords: framework; internet of health things; fall detection.

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IoT	Internet of Things
IoHT	Internet of Health Things
API	Application Programming Interface
TAM	Technology Acceptance Model
TIC	Tecnologias de Informação e Comunicação
AM	Aprendizagem de Máquina
MedDRA	Medical Dictionary for Regulatory Activities Terminology
SSN	Semantic, Sensor, Network
WIoT	Wearable Internet of Things
KNN	K-nearest neighbors
SVM	Support Vector Machine
HEMOCE	Centro de Hematologia e Hemoterapia do Ceara
AAL	Ambient Assisted Living

SUMÁRIO

1 INTRODUÇÃO

Esta tese apresenta um *framework* para o desenvolvimento de soluções para detecção de quedas e identificação das suas possíveis causas utilizando como base dispositivos vestíveis da Internet das Coisas (IoT, do inglês *Internet of Things*). O *framework* possui uma arquitetura bem definida para facilitar a sua instanciação e um modelo de dados orientado a objeto, que fornece suporte para o tratamento de dados distintos.

Este capítulo apresenta a contextualização do trabalho na Seção 1.1; na Seção 1.2, é descrita a motivação deste trabalho; a Seção 1.3 contém a hipótese e as questões de pesquisa que guiam esta tese; a Seção 1.4 descreve o objetivo e as principais contribuições; a metodologia deste trabalho é explorada na Seção 1.5; e a Seção 1.6 define a estrutura desta tese.

1.1 Contextualização

A Internet das Coisas (IoT, do inglês *Internet of Things*) permite a comunicação entre objetos do dia-a-dia (GUBBI *et al.*, 2013), podendo ser dispositivos físicos ou virtuais conectados com a Internet (NOURA *et al.*, 2019). Com isso, serviços mais robustos são providos aos usuários, decorrente do fato dos objetos coletarem dados dos usuários e do ambiente e compartilharem entre si ou enviarem para uma nuvem a fim de processar tais dados.

A IoT nasceu da união de várias áreas da Computação, como Redes de Sensores Sem Fio, Computação Ubíqua, Computação em Nuvem e *Big Data* (AL-FUQAHA *et al.*, 2015; ANDRADE *et al.*, 2017a) e, por suas características, pode contribuir em diferentes setores, abrangendo desde cenários de agricultura até saúde, passando também por economia de energia (TSAI *et al.*, 2021; NORD *et al.*, 2019; GUBBI *et al.*, 2013).

A área da saúde com IoT, foco deste trabalho, é denominada de *Internet of Health Things (IoHT)* (ALSALIBI *et al.*, 2021) e é uma das mais atrativas dentre aquelas possibilitadas pela IoT, pois pode aprimorar muitas aplicações, como atividades físicas, monitoramento remoto, cuidado de idosos, entre outros. Isso é possibilitado pelo uso dos dispositivos inteligentes, os quais tem tido seu uso aumentado para monitorar continuamente os usuários de qualquer lugar e a qualquer momento (USAK *et al.*, 2020). Esse monitoramento contínuo pode ser feito com o apoio de dispositivos vestíveis, como o APPLE WATCH (2022), o qual contém, dentre muitos sensores e funcionalidades, um sensor de eletrocardiograma com a capacidade de identificar comportamentos anormais do coração e sugere, quando necessário, uma consulta médica.

Dentro dos cenários de monitoramento contínuo do usuário, a IoHT também pode auxiliar na detecção de quedas, uma vez que é possível identificar as ações executadas por ele a partir de sensores presentes nos dispositivos vestíveis, como o acelerômetro. Para entender como a IoHT pode auxiliar nesse domínio, é importante compreender que a queda é um evento, no qual uma pessoa vai de encontro ao solo ou a um local mais baixo do que se encontra, consciente ou inconscientemente, com lesão ou não (FREITAS *et al.*, 2016). A detecção de quedas é um ponto importante para ser monitorado, pois as quedas são a segunda maior causa de lesões não intencionais e podem levar até a morte, segundo a OMS (2021). Ainda de acordo com a OMS (2021), o risco de queda aumenta com o passar da idade, sendo maior nas pessoas idosas.

Assim, a detecção de quedas se mostra ainda mais relevante dado a expectativa de vida está aumentando e a população mundial será de 25% de idosos em todos os continentes, exceto na África, em 2050 (Nações Unidas, 2017). No cenário brasileiro, segundo as projeções do Instituto Brasileiro de Geografia e Estatística (IBGE), atualmente 14% da população possui 60 anos (idade que define pessoas idosas no Brasil) ou mais, e, em 2050, essa parcela da população compreenderá 28,45% da população (IBGE, 2018). Diante desses dados, percebe-se o crescimento dessa faixa da população tanto no Brasil como no mundo, o que requer mais cuidados com a saúde. Além disso, o risco é maior nos idosos e, de acordo com Ang *et al.* (2020), 30% das pessoas que caem uma vez, caem novamente, requisitando um cuidado maior, especialmente com os idosos.

Diante dos dados apontados, é importante desenvolver soluções que contribuam para detectar o instante em que uma queda ocorre, buscando prover um atendimento mais rápido e minimizando as sequelas. Contudo, a queda pode ocorrer devido a diversas causas, que podem ser classificadas em fatores intrínsecos e extrínsecos (OMS, 2007; NGUYEN *et al.*, 2018). Os fatores extrínsecos são relacionados ao ambiente em que a pessoa se encontra, como pisos escorregadios e a existência de corrimão, enquanto os intrínsecos são relacionados a problemas de saúde, histórico de quedas e medicamentos, entre outros. Desses últimos, os problemas de saúde podem ser monitorados através de dados fisiológicos, os quais podem ser coletados por dispositivos vestíveis, provendo serviços mais precisos e permitindo relacionar uma queda com a possível causa.

Para auxiliar no monitoramento do usuário, buscando identificar uma queda ou suas causas, existem soluções variadas. Para identificar a queda, podem ser utilizados sensores de movimentos presentes em dispositivos vestíveis, como o Samsung Watch 4 (SAMSUNG, 2022)

e o Huawei Watch GT 3 (HUAWEI, 2022). Para dados fisiológicos, existem aqueles para medir a glicose como o Performa Connect (ACCU-CHEK, 2022) e o K'Watch (K'WATCH, 2022), os batimentos cardíacos como em Polar (2022), a pressão arterial em OMRON (2022), Medical (2022), entre outros. Com os dados coletados por tais dispositivos, em associação com os de movimento, é possível associá-los para identificar uma queda com as possíveis causas, em uma situação de anomalia nos dados fisiológicos, o que contribui para prover informações precisas para as equipes de saúde responsáveis pelo atendimento do usuário.

Nesse contexto em que muitos dispositivos vestíveis podem ser utilizados para monitorar o usuário, é importante desenvolver soluções que facilitem o desenvolvimento de aplicações para detecção de quedas e suas causas e que lidem com os dados diferentes gerados pelos dispositivos vestíveis. Assim, uma das formas de facilitar o desenvolvimento é utilizando soluções de reuso, como *frameworks*, arquitetura ou padrões. Os *frameworks*, por exemplo, pode aumentar a qualidade, desempenho, confiança e interoperabilidade (FAYAD; SCHMIDT, 1997).

1.2 Motivação

Diante do exposto na Seção 1.1, é importante notar que a IoHT pode contribuir em várias áreas, dentre elas, a detecção de quedas. Isso ocorre, pois dispositivos inteligentes podem monitorar os usuários e, a partir dos dados coletados por eles, é possível identificar situações incomuns e alertar quando elas ocorrerem. Porém, a IoHT não se limita à detecção de quedas, ela permite explorar outros dados para facilitar a identificação das possíveis causas e permitir que as equipes de emergência forneçam um atendimento mais adequado e preciso aos usuários, reduzindo, assim, as sequelas.

Como visto na Seção 1.1, dois tipos de fatores podem estar envolvidos em uma queda, o intrínseco e o extrínseco. Como os intrínsecos são fatores relacionados a problemas de saúde, como problemas de pressão, diabetes, entre outros (BUKSMAN *et al.*, 2008), é possível monitorá-los com o apoio de dispositivos vestíveis e identificar situações anômalas nos dados fisiológicos dos usuários.

Ao mesmo tempo em que a evolução dos dispositivos da IoHT permite o monitoramento de vários aspectos fisiológicos dos usuários, é notada a presença da heterogeneidade dos dispositivos, umas das características da IoT de acordo com Razzaque *et al.* (2016). Tal característica é muito importante, pois amplia o espectro de monitoramento, porém insere novos desafios a serem tratados pelos desenvolvedores de aplicações IoHT. Esses desafios estão relacio-

nados às diferentes formas de coletar os dados, de organizá-los, processá-los e de interpretar os resultados.

Nesse contexto, ilustra-se a diferença dos dados com o exemplo de batimentos cardíacos, como feito pelo dispositivo Polar (2022) e da glicemia, como coletado em K'watch (2022). Caso seja necessário um monitoramento contínuo, os batimentos cardíacos devem ser medidos a cada segundo, enquanto a glicemia pode ser medida em horas. Segundo a *American Diabetes Association* (ADA) (ADA, 2022), a glicemia deve ser monitorada de 6 a 10 vezes durante um dia, podendo variar de acordo com as necessidades de cada pessoa.

Assim, percebe-se que cada dispositivo possui sua própria forma de tratar os dados, tornando mais difícil o desenvolvimento de uma solução que englobe vários dispositivos. Isso é demonstrado com diversos trabalhos que monitoram condições clínicas dos pacientes, sejam eles vestíveis (HUSSAIN *et al.*, 2019; SALEH; JEANNÈS, 2019; BOUTELLAA *et al.*, 2019) ou inseridos no ambiente (MOKHTARI *et al.*, 2018; PALIPANA *et al.*, 2018; BHANDARI *et al.*, 2017), como sensor de presença ou de pressão no solo. Por outro lado, outros trabalhos utilizam dispositivos diferentes, contudo, não apresentam a possível causa intrínseca da queda, perdendo a oportunidade de prover mais dados para equipe médica e, com isso, tornando o atendimento médico impreciso, uma vez que a equipe médica desconhece as possíveis causas intrínsecas que resultaram na queda. Isso se mostra importante, pois há iniciativas na literatura que buscam identificar as causas relacionadas à movimentação, como um tropeço, escorregão ou desmaio (SASAKI *et al.*, 2022; AZIZ *et al.*, 2014; AZIZ; ROBINOVITCH, 2011), sem considerar as causas intrínsecas.

Assim, uma solução composta por vários dispositivos distintos, também chamado aqui de multidispositivos, pode ser mais precisa tanto para a detecção de quedas como relatado em Nizam e Jamil (2020) e Gjoreski *et al.* (2020), quanto para as causas, devido à associação dos dados distintos coletados. Nesse cenário, existem iniciativas que visam associar um tipo de monitoramento, como o de batimentos cardíacos, com a detecção de quedas (HORTA *et al.*, 2015). Entretanto, as soluções não costumam ser genéricas, ficando restrita a um tipo de dispositivo de monitoramento, além de ser preciso agregar mais informações relevantes para o contexto visando o provimento de serviços mais corretos. Outro aspecto que dificulta a utilização é que muitos dos dispositivos não são de uso natural para o usuário, por exemplo, o uso de smartphones no tórax do usuário como em Piva *et al.* (2014) ou uma solução com com vários fios monitorando o coração como em Horta *et al.* (2015).

No contexto do desenvolvimento de soluções IoHT com foco em quedas, também foi observada a ausência de trabalhos com reuso de software (NGUYEN *et al.*, 2018) tendo como fonte de dados múltiplos dispositivos, de forma a reduzir o tempo de desenvolvimento e que seja de fácil adaptação para permitir a adição de novos dispositivos inteligentes (e.g., smartwatch, glicosímetro, monitor cardíaco) ou alterações nas informações dos dispositivos já existentes para compor a solução (e.g., um smartphone passa a ter um novo sensor).

Quando observa-se artefatos de reuso de forma geral, existem os *frameworks*, os quais são definidos como um trecho de código semi-acabado e pronto para usar (PREE, 1995). Eles correspondem a códigos de softwares predefinidos que permitem a reutilização do mesmo em aplicações de um determinado domínio, visando reduzir o tempo e o esforço necessário para o desenvolvimento de um software, além de ajudar a aumentar a qualidade (FAYAD; SCHMIDT, 1997).

Quando associados à IoHT, que possui requisitos críticos, como tempo de resposta curto e processamento de dados diferentes oriundo de múltiplos dispositivos diferentes, um *framework* permite o uso de funcionalidades já desenvolvidas e estáveis e com isso minimiza esses requisitos críticos. Além disso, o uso de *frameworks* também facilita o desenvolvimento de aplicações com a redução do tempo de desenvolvimento.

É também importante ressaltar que os artefatos de reuso, como o *framework*, são adaptáveis, o que atende a característica de adaptação da IoT (RAZZAQUE *et al.*, 2016). Essa adaptação deve ocorrer em tempo de execução para lidar com a ausência ou presença de um ou mais dispositivos, o que pode ocorrer, por exemplo, pela descarga da bateria. A partir daí, a solução deve associar os dispositivos com os tipos de dados e gerenciar os algoritmos a serem executados.

No contexto de múltiplos dispositivos vestíveis IoHT, que monitoram características diferentes do usuário, as quais podem ser relacionadas à queda propriamente dita ou às causas, surge o problema: *Como desenvolver aplicações IoHT que utilizam multidispositivos vestíveis para detectar quedas e suas possíveis causas, de modo que o tempo de desenvolvimento dessas aplicações seja reduzido e que os desenvolvedores tenham a percepção de facilidade e de utilidade dessa solução?*

1.3 Hipótese e Questões de Pesquisa

Considerando o problema previamente descrito, define-se a hipótese de pesquisa como segue:

Em um ambiente IoHT, o uso de um *framework* para o desenvolvimento de aplicações que coletam dados de múltiplos dispositivos vestíveis para monitorar o usuário com o objetivo de detectar quedas e possíveis causas reduz o tempo de desenvolvimento dessas aplicações e os desenvolvedores têm a percepção de que o *framework* é útil e fácil em comparação com o desenvolvimento sem essa solução.

Diante da análise da hipótese descrita, são derivadas questões de pesquisa que contribuem para a sua comprovação, as quais são descritas a seguir:

- **Questão de Pesquisa 1:** *Quais os sensores mais utilizados para detecção de quedas que estão presentes em dispositivos vestíveis da IoT?*
  - Justificativa: Diante da variedade de dispositivos e sensores que monitoram o usuário, quais os sensores mais utilizados, uma vez que os algoritmos e os resultados obtidos podem variar conforme o sensor utilizado. Assim, a resposta para essa pergunta pode indicar algoritmos consolidados para serem inseridos no *framework*, facilitando o desenvolvimento de uma nova solução de detecção de quedas e as possíveis causas.
- **Questão de Pesquisa 2:** *Quais funcionalidades comuns são capturadas pelos frameworks de saúde existentes?*
  - Justificativa: Existem *frameworks* que são utilizados para o desenvolvimento de aplicações IoT como demonstrado em (ARAÚJO, 2018) e cada um contribui para o desenvolvimento focando em funcionalidades comuns de um domínio. Essa questão visa descobrir se já existem *frameworks* para detecção de quedas e suas causas.
- **Questão de Pesquisa 3:** *Como integrar os dados fisiológicos do usuário monitorados pelos dispositivos vestíveis com a informação de uma queda para identificar a causa da mesma?*
  - Justificativa: Uma vez que muitos dados são coletados, dentre eles dados fisiológicos, além de dados da movimentação do corpo, que podem indicar uma possível queda, e possuem formatos diferentes, como relacionar esses tipos de informação?

1.4 Objetivo e Resultados Esperados

Este trabalho tem como objetivo desenvolver um *framework* que combine dados de múltiplos dispositivos da IoT, sendo o foco em dispositivos vestíveis, de modo a detectar quedas com maior precisão e associar com fatores que podem ocasionar a queda. O *framework* deve fazer uma análise temporal dos dados considerando o instante em que foram detectadas situações anômalas de quedas e a partir de então, detectar as possíveis causas. Para apoiar tal identificação, deve ser especificado um modelo de dados para lidar com dados distintos.

Em resumo, os principais resultados esperados deste trabalho são:

- *Framework:* visando alcançar o objetivo de contribuir para a redução do tempo de desenvolvimento das aplicações do domínio descrito, um *framework* é desenvolvido contendo algoritmos de processamento de dados e fazendo a associação entre as quedas e as possíveis causas.
- *Modelo de Dados Orientado a Objeto:* associar dados distintos requer modelos de dados para tratar e lidar com eles, reduzindo a complexidade, assim, nesta tese, é especificado um modelo de dados orientado a objetos para dar suporte ao framework com o objetivo de lidar com os diferentes tipos de dados oriundos dos dispositivos vestíveis.

1.5 Metodologia

Para atingir os objetivos apresentados na Seção 1.4, a metodologia seguida nesta tese é apresentada na Figura 1 e descrita a seguir. Ela é dividida em três etapas, sendo a primeira a concepção, a segunda a execução e a terceira a avaliação.

Primeiramente, é feito o planejamento da *revisão da literatura*, sendo essa revisão da literatura *baseada* nas etapas e atividades de uma revisão sistemática[1] (RS) (WOHLIN *et al.*, 2012; KITCHENHAM *et al.*, 2009). Contudo, ressalta-se que não foi feita uma RS e sim a execução dos passos de uma RS até a obtenção dos artigos para leitura. Os elementos que guiaram a revisão de literatura deste trabalho são descritos a seguir.

Na concepção, houve a definição dos objetos de estudo deste trabalho, que consistiu de *frameworks* de suporte ao desenvolvimento que buscassem auxiliar o desenvolvimento de aplicações no domínio de *IoHT*, uma vez que utiliza-se sensores para monitorar os usuários, especialmente de dispositivos vestíveis. Logo, a saída dessa etapa corresponde ao tema *framework*

[1] O conceito de revisão sistemática utilizado é o da Engenharia de Software Experimental.

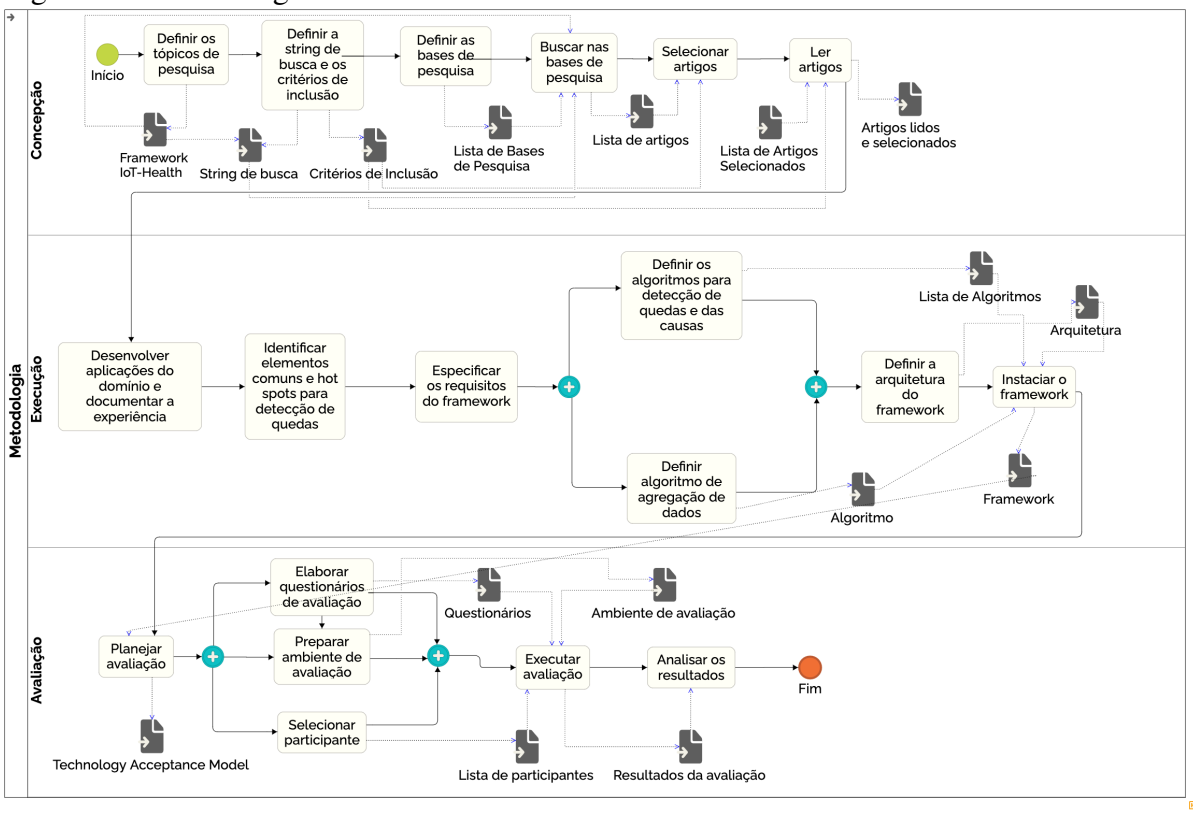
IoHT.

Em seguida, foram definidos os critérios de aceitação e a *string* de busca, a qual contém as palavras-chave utilizadas para encontrar os artigos. A lista de critérios buscou verificar se os artigos possuíam mais de 4 páginas, se foram escritos em inglês e se apresentavam soluções para detecção de quedas e utilizavam multidispositivos vestíveis da IoT.

Quanto à *string* de busca, não foram utilizados descritos, mas ela foi composta por palavras como *framework* e termos similares como *tool* e Application Programming Interface (API). Associado a elas, foram adicionados termos relativos ao domínio pesquisado (*health*). Por fim, foram adicionadas palavras representativas para a IoT e que pudessem retornar algum resultado relevante como "*context-aware*" e "*ubiquitous computing*". Dessa forma, a *string* ficou definida:

TITLE-ABS-KEY ((framework OR tool OR api) AND (healthcare OR fall OR e-health) AND (iot OR "internet of things"OR context-aware OR "ubiquitous computing"))*

Figura 1 – Metodologia utilizada neste trabalho



Fonte: O Autor.

Em geral, foi feito um filtro pelos títulos e resumos dos artigos para identificar aqueles que atendem aos critérios, para, em seguida, lê-los completamente. Ao final, foram encontrados

os artigos que formam a base para este trabalho e passou-se a desenvolver o *framework*.

O desenvolvimento do *framework* foi baseado nos conceitos presentes em Pree (1995) e na adaptação das metodologias especificadas em Yang *et al.* (1998), Wilson e Wilson (1993) e Stanojević *et al.* (2011) e foi composta por três fases. A primeira consistiu na definição do *framework*, que teve sua origem no desenvolvimento de aplicações de detecção de quedas e que gerou a experiência para o autor com o desenvolvimento de aplicações para detecção de quedas de idosos (ALMEIDA *et al.*, 2016; ARAÚJO *et al.*, 2018; LINHARES *et al.*, 2020) e que iniciou em 2016. Além disso, essa experiência foi documentada e permitiu a geração dos elementos comuns e os *hot spots*, que são os elementos que mudam de acordo com a aplicação desenvolvida. Essa fase ocorre enquanto a extração desses elementos não for satisfatória.

A fase de definição do *framework* gera insumos para a fase de desenvolvimento do *framework*, que consiste da especificação de requisitos, em que também é feita a definição dos dos algoritmos de processamento do dados recebidos dos sensores, o algoritmo para agregar os resultados e para tomar a decisão. Em seguida, é feito o projeto do *framework*, que gera a arquitetura e diagramas. Na sequência é feita a instanciação do *framework*, o qual é testado para assegurar que os requisitos foram corretamente implementados. Esse processo ocorre enquanto todos os requisitos não foram documentados e implementados.

Por último, foi realizado o planejamento da avaliação, sendo a mesma um estudo empírico em ambiente sintético (ZELKOWITZ *et al.*, 2003) com o desenvolvimento de uma aplicação usando o Ágape. Além disso, também é avaliado o tempo de desenvolvimento do Ágape em comparação com o tempo de desenvolvimento de uma aplicação de detecção de quedas, sendo ela o WatchAlert (ALMEIDA *et al.*, 2016). Além do tempo, para apoiar o estudo de caso, foi feita uma avaliação qualitativa baseada no Technology Acceptance Model (TAM) (DAVIS, 1989), o qual permite avaliar a aceitação de tecnologias quanto à duas características: percepção de utilidade e percepção de facilidade. Ademais, foi coletado o tempo de configuração para usar o Ágape, uma vez que um tempo superior para configurar, pode levar ao não do uso do *framework*. Na sequência, três atividades ocorrem em paralelo, sendo elas: definição dos participantes, a preparação do ambiente e a construção dos questionários de avaliação.

Por fim, a fase de avaliação possui mais duas atividades. A primeira delas é a execução propriamente dita do estudo de caso realizado a partir das definições anteriores. Como saída, há os resultados obtidos após a execução da avaliação pelos participantes. Essa avaliação é feita posteriormente, sendo a atividade dessa fase. Na avaliação dos resultados é possível

comprovar se o *framework* atende aos requisitos para o qual foi proposto.

1.6 Estrutura da Tese

Os capítulos da tese estão organizados conforme a descrição a seguir:

- **Capítulo 2:** apresenta a fundamentação teórica necessária para a o entendimento do trabalho aqui proposto;
- **Capítulo 3:** são descritos os trabalhos relacionados ao que é proposto nesta tese;
- **Capítulo 4:** solução composta por um *framework* e um modelo de dados;
- **Capítulo 5:** sobre a avaliação experimental a ser realizada de modo a validar o trabalho proposto;
- **Capítulo 6:** conclui a tese, sumarizando os tópicos apresentados e as publicações.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica, sendo uma visão geral dos tópicos relacionados à esta tese sendo eles Internet das Coisas, detecção de quedas, modelo de dados e artefatos de reuso.

A Internet das Coisas é o meio pelo qual a proposta deve ser implementada, uma vez que ela permite a utilização de multidispositivos inteligentes para monitorar o paciente, especialmente pela utilização de dispositivos vestíveis. Já a detecção de quedas é um problema grave e com sérias consequências, especialmente em idosos, e que podem ser minimizadas pela utilização dos dispositivos da IoT. Esses dispositivos geram dados distintos e para amenizar as diferenças entre eles, um modelo de dados pode ser utilizado. Por fim, os artefatos de reuso, os quais podem ser utilizados para facilitar o desenvolvimento de uma aplicação, são apresentados.

Dessa forma, o capítulo está dividido da seguinte maneira: na seção 2.1 são apresentados os conceitos básicos de IoT, assim como a IoT pode auxiliar na área da saúde; na seção 2.2 são descritas as formas de detecção de uma queda além da possibilidade de associá-la com as causas, contendo um exemplo da literatura; na seção 2.4 são descritos os tipos de modelos de dados existentes; a seção 2.5 contém o conceito de *frameworks*, processos de desenvolvimento deles e exemplos da IoT; e, por fim, a seção 2.6 conclui o capítulo.

2.1 Internet das Coisas

2.1.1 Conceitos

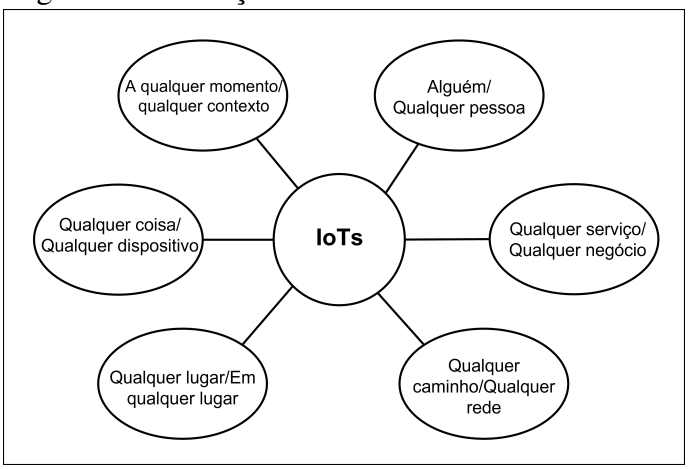
A Internet das Coisas (IoT, do inglês *Internet of Things*) é um paradigma da Computação, o qual evoluiu da união de diversas áreas como Computação Ubíqua, Redes de Computadores, Dispositivos Embarcados, Redes de Sensores Sem fio (ANDRADE *et al.*, 2017a; AL-FUQAHA *et al.*, 2015) e dos avanços nos sensores e atuadores (RAZZAQUE *et al.*, 2016) e que permite que objetos ao nosso redor possam se conectar à Internet (NOURA *et al.*, 2019; GUBBI *et al.*, 2013; VASSEUR; DUNKELS, 2010). Além disso, tais objetos podem ser providos de inteligência, o que contribui para que eles possam ver, ouvir, pensar e desempenhar funções de maneira conjunta (AL-FUQAHA *et al.*, 2015).

Nesse sentido, a IoT é composta por vários dispositivos conectados à Internet que pode coletar, compartilhar e analisar uma grande quantidade de dados (IMRAN *et al.*, 2021) ou

como apresentado em Mishra e Pandya (2021) ela caracteriza pela coleta e transferência de dados sem intervenção humana. Diante dessa inteligência, tais dispositivos facilitam e contribuem para o provimento de novos serviços customizados e centrados no usuário (CHATTERJEE; ARMENTANO, 2015).

Outra visão da IoT descreve que ela é um conceito que remete a um conjunto conectado de qualquer pessoa, qualquer coisa, a qualquer hora, qualquer lugar, qualquer serviço e através de qualquer rede (HUANG *et al.*, 2021; ISLAM *et al.*, 2015; SUNDMAEKER *et al.*, 2010), sendo tais conceitos relacionados conforme apresentado na Figura 2. Isso pode levar a criação de ambientes inteligentes, sejam eles casas, prédios, ambientes de monitoramento de usuários ou pode ser aplicado em cenários automotivos, de telecomunicações, agricultura, segurança, infraestrutura urbana, serviços de emergência (SUNDMAEKER *et al.*, 2010; GUBBI *et al.*, 2013; AL-FUQAHA *et al.*, 2015; ISLAM *et al.*, 2015).

Figura 2 – Definição de IoT



Fonte: Adaptado de (SUNDMAEKER *et al.*, 2010)

Nesses ambientes, o usuário pode interagir com diversos objetos inteligentes, os quais permitem capturar dados de uso (RAHMANI *et al.*, 2022), bem como dados sobre o usuário visando o aprimoramento de serviços. Por exemplo, no cenário de uma casa inteligente o usuário pode interagir com a televisão, a qual aprende os hábitos dele e pode ligar automaticamente no dia e horário de um programa que o usuário goste. Além disso, pode interagir com o ar-condicionado de modo que ele ajuste a temperatura baseado no perfil do todas as pessoas presentes na casa como a aplicação AutomaGREat Andrade *et al.* (2017a). Ou ainda, se o usuário estiver saindo do trabalho e se deslocando até a sua casa, as luzes e o ar-condicionado podem ser ligados para que ele chegue e encontre o ambiente conforme as suas preferências.

Existem outros dois conceitos similares quanto ao papel da IoT, que é colocada

como uma infraestrutura, sendo eles encontrados em International Telecommunication Union (2012) e ISO/IEC (2014). O primeiro diz que a IoT é uma infraestrutura que permite serviços avançados com a interconexão de objetos baseada nas tecnologias de informação e comunicação interoperáveis. No segundo, a IoT é uma infraestrutura de objetos, pessoas, sistemas e recursos de informação interconectados juntos com serviços inteligentes para processar informações físicas e virtuais e reagir. Além disso, cabe destacar que o conceito encontrado em ISO/IEC (2014) é mais abrangente, envolvendo pessoas e sistemas além de objetos sendo mais próximo dos conceitos descritos anteriormente.

Cabe destacar os objetos inteligentes como ponto de convergência entre as definições, de modo que ele possui um papel importante para a implantação da IoT no dia-a-dia das pessoas. Cada um desses objetos ajuda na composição dos serviços, pois dados de localização coletadas pelo relógio inteligente podem ser associados com a movimentação do usuário detectada por um colar e se ele estiver em casa a interação com os objetos podem descrever os hábitos e rotina do usuário permitindo detectar comportamentos anômalos do usuário.

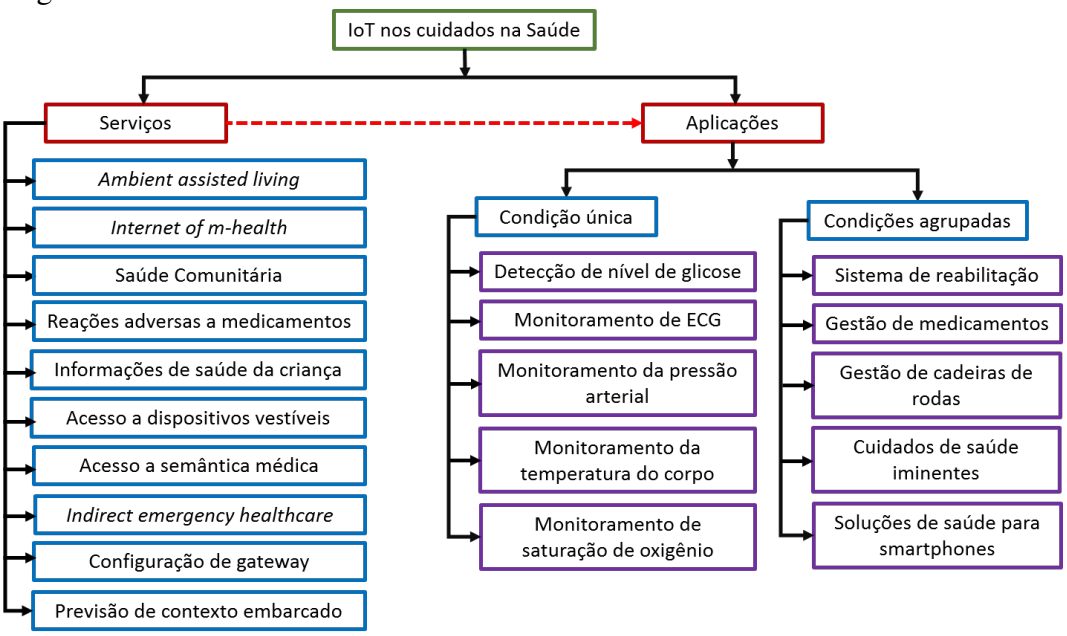
2.1.2 Internet of Health Things

Diante do exposto na Seção 2.1.1, a IoT pode atuar em diversas áreas, entre elas a de saúde (MISHRA; PANDYA, 2021; GUBBI *et al.*, 2013; CHATTERJEE; ARMENTANO, 2015), visando prover uma melhoria na qualidade de vida das pessoas, especialmente em pessoas idosas. Nesse contexto, vários serviços podem ser fornecidos para os usuários, e, para isso, podem ser consideradas informações coletadas a partir de condições únicas ou a partir da associação de várias informações. A demonstração de tipos de serviços e aplicações de IoHT são apresentados na Figura 3.

O primeiro serviço citado é o *Ambient Assisted Living (AAL)*, que é o uso de Tecnologias de Informação e Comunicação (TIC) na vida diária de uma pessoa e ambientes de trabalho possibilitando-a permanecer ativa, socialmente conectada e viva de forma independente na velhice (MONEKOSSO *et al.*, 2015). Além disso, o AAL pode ajudar idosos durante várias fases do processo de envelhecimento contribuindo para manter os idosos conectados, saudáveis, ativos e felizes (CICIRELLI *et al.*, 2021). Esse tipo de ambiente cobre uma grande variedade de tópicos que vão desde reconhecimento de atividades (MONEKOSSO *et al.*, 2015) até envio de alertas de emergências (IPIÑA *et al.*, 2009).

O segundo serviço descrito na Figura 3 prescreve o uso da computação móvel,

Figura 3 – IoT nos cuidados da saúde



Fonte: Adaptado de (ISLAM *et al.*, 2015)

sensores médicos e tecnologias de comunicação para serviços médicos (ISTEPANIAN *et al.*, 2004). Esse tipo de serviço tem sido analisado para, por exemplo, identificar formas não invasivas de medir o nível da glicemia do paciente (MONEKOSSO *et al.*, 2015), ou batimentos cardíacos, nível de oxigênio no sangue, entre outros. Ainda no aspecto individual de cada paciente, soluções IoT podem ser utilizadas na identificação de reações adversas ao uso de medicamentos, tanto pela não adaptação a um medicamento, como alergias, ou não surtindo o efeito desejado, quanto pela combinação de mais de um deles, podendo acarretar em outros problemas de saúde (ISLAM *et al.*, 2015).

Quanto às tecnologias de comunicação citados anteriormente, podem existir serviços como a telemedicina, que engloba teleconsulta, telemonitoramento e reabilitação assistida por computador. A telemedicina ganhou mais relevância, tornando-se necessária com o surgimento da pandemia do novo coronavírus, com a COVID-19 (Haghi Kashani *et al.*, 2021). Além disso, ao observar sob a ótica de hospitais, a telemedicina reduz os custos de cuidados médicos, além de aumentar a capacidade de diagnósticos (PRIYADARSHINI *et al.*, 2022).

Outro serviço lida com o monitoramento da saúde comunitária, cujo conceito advém da possibilidade de estabelecer uma rede de cobertura de uma área de uma comunidade local, como redes locais em torno de um hospital ou uma área rural (ISLAM *et al.*, 2015). Isso permite o mapeamento de problemas e questões sanitárias, podendo atuar de maneira mais efetiva sobre determinada área. Além disso, podem ser utilizadas para monitorar determinada parcela

da população como as crianças visando observar problemas emocionais, comportamentais ou mentais deles mesmos ou dos seus familiares (ISLAM *et al.*, 2015).

Quanto ao compartilhamento de informações médicas, ontologias tem sido utilizadas para representar o conhecimento em sistemas de saúde e também em IoT, como em Duncan *et al.* (2015), Kumar *et al.* (2015) e Ali *et al.* (2018). Assim, uma ontologia pode contribuir com outro serviço que é o cuidado de saúde de emergência indireto. Esse serviço é descrito como situações de emergência em que a saúde das pessoas está associada, como acidentes de trânsito ou problemas ambientais.

Os dois últimos serviços estão associados com a configuração de *gateways* e predição de contexto embarcado (ISLAM *et al.*, 2015). O primeiro é um serviço arquitetural, cuja função é conectar os nós da IoT, sendo esses nós os equipamentos médicos. O segundo é um serviço a ser fornecido por um framework de modo que facilite para os desenvolvedores a criação de aplicações sensíveis ao contexto voltadas para os cuidados com saúde, as quais também podem ser de IoT.

Quanto às aplicações, elas são classificadas, conforme visto na Figura 3, em condições únicas e condições agrupadas. No caso das condições únicas, são aplicações que coletam dados relativos às condições do paciente, como medição da glicemia ou da pressão arterial ou ainda informações de um eletrocardiograma (ECG) e são usadas para identificar uma condição única.

No caso das condições agrupadas, tem-se as condições de reabilitação do paciente, visando uma melhoria na qualidade de vida e na restauração das habilidades funcionais de pessoas com alguma limitação física. Outros tipos de aplicações que consideram as condições agrupadas podem ser citadas, como o gerenciamento de medicamentos, gerenciamento de cadeira de rodas, sistemas de saúde utilizados em *smartphones*, entre outros. Em todas as situações, o foco é conhecer como estão os cuidados com o paciente, e, no caso da cadeira de rodas, também serve para automatizar totalmente as suas funções para pessoas com limitações.

Ainda quanto as condições únicas ou agrupadas, tem-se a detecção de quedas como uma condição única, pois para isso, utiliza-se dados de sensores de movimentação do corpo, como descrito na Seção 2.2. Em contrapartida, uma condição agrupada refere-se à detecção de quedas e suas possíveis causas, o que requisita o uso de vários sensores diferentes para realizar tal ação.

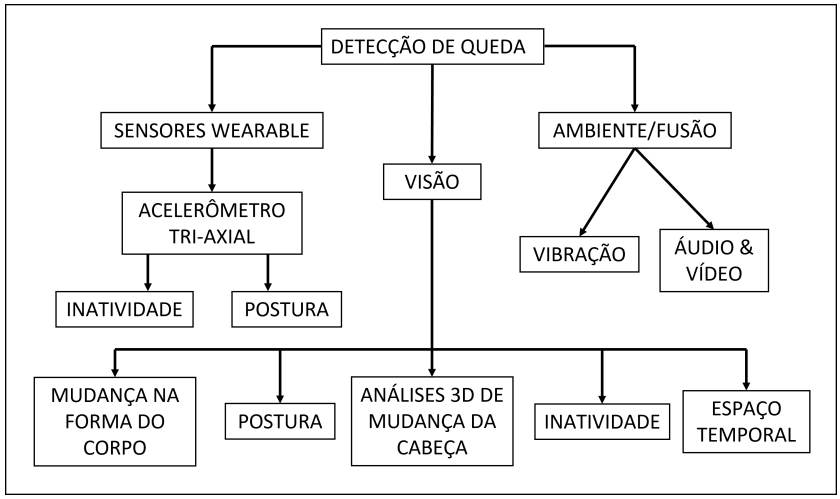
2.2 Tecnologias para Detecção de Quedas

Quedas são a segunda principal causa de mortes acidentais segundo a OMS (2021), sendo necessárias iniciativas para auxiliar a prevenir ou identificar mais rapidamente quando uma delas ocorre de modo a diminuir as sequelas (ARAÚJO *et al.*, 2018). Assim, uma queda pode ser identificada com o apoio da tecnologia, especialmente a IoT, devido ao uso de diversos dispositivos. Em Tanwar *et al.* (2022), Raeve *et al.* (2022), Ahamed *et al.* (2019) e Mubashir *et al.* (2013), tais tecnologias são sumarizadas em três categorias a partir do método e dos dispositivos utilizados para identificar uma queda:

1. **Dispositivos vestíveis:** permitem detectar a movimentação do corpo do usuário;
2. **Visão Computacional:** processa imagens de vídeo para identificar uma queda; e
3. **Ambiente/Fusão de dados:** mescla vários dados para identificar uma queda.

A Figura 4 contém a organização dos métodos de classificação seguindo a descrição anterior, a qual classificação em três tipos. Os dispositivos vestíveis, embora estejam descritos na Figura 4 apenas com o acelerômetro como também encontrado em ARAÚJO *et al.* (2018), podem utilizar outros sensores, como o giroscópio (ALMEIDA *et al.*, 2016; HSIEH *et al.*, 2014). O objetivo é aprimorar as métricas, como a sensibilidade – quantidade de acertos de situações de queda – dos algoritmos, o que reduz o número de situações de falsos negativos. A forma mais comum dos sensores citados corresponde a presença de três eixos (X, Y e Z) para coletar os dados do corpo do usuário, permitindo identificar quando ele passa da postura em pé ou sentado, por exemplo, para o chão. Mais detalhes são apresentados nas subseções seguintes.

Figura 4 – Formas de detecção de quedas



Fonte: Adaptado de (MUBASHIR *et al.*, 2013)

2.2.1 Dispositivos vestíveis

Ao utilizar um dispositivo vestível, é possível utilizar diversos sensores para detectar uma queda, sendo o mais utilizado o acelerômetro. Para entender melhor como os sensores de dispositivos vestíveis podem ser utilizados são apresentados nas Figuras 5 e 6 dados de um acelerômetro, coletados a partir de um relógio inteligente indicando o comportamento característico de uma pessoa que sofreu uma queda. Essa forma de identificação de uma queda é encontrada de modo similar em ARAÚJO *et al.* (2018), Khojasteh *et al.* (2018), Hsieh *et al.* (2014) e Almeida *et al.* (2016). Para o uso dos dados é preciso converter os valores dos três eixos em um valor único e os algoritmos utilizam a equação 2.1. Destaca-se que para o giroscópio o princípio é similar ao do acelerômetro.

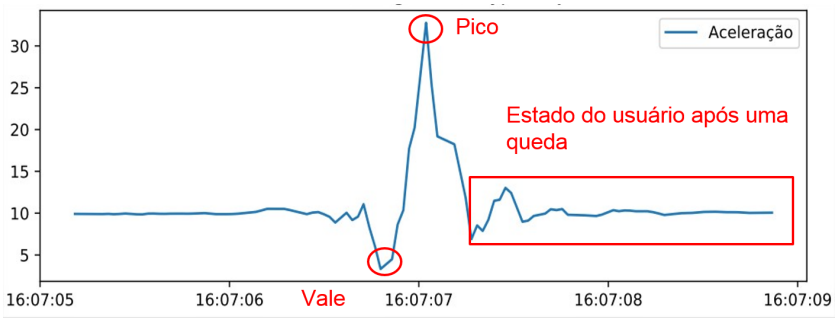
$$VA(t_i) = \sqrt{{A_x}^2(t_i) + {A_y}^2(t_i) + {A_z}^2(t_i)} \tag{2.1}$$

Tomando como base o acelerômetro, existem três situações específicas que permitem a identificação de uma queda. A primeira delas é uma queda livre e a segunda caracteriza um impacto do usuário com o solo. Essas duas são importantes, pois representam o comportamento de uma queda, uma vez que esse comportamento não aparece, uma queda é logo descartada. Contudo, para confirmar se uma é queda ou outra ação como bater palmas ou pular, é preciso analisar o comportamento final dos dados. Caso seja um comportamento de estabilidade, sem movimentação, uma queda pode ser caracterizada. Caso contrário, pode ser outra atividade, não sendo, assim, uma queda.

No gráfico da Figura 5, as três situações são abordadas. O eixo vertical corresponde à aceleração da gravidade, enquanto o horizontal refere-se ao tempo. A primeira situação, a de queda livre é representada pelo ponto mais abaixo no gráfico e identificado como vale. Percebe-se que o valor obtido durante a queda livre é próximo a $0m/s^2$. Em contrapartida, quando o usuário tem um contato com o solo, chamado de pico no gráfico, o valor da aceleração aumenta bruscamente em menos de 1s, chegando a ultrapassar a $30m/s^2$, o que corresponde a 3x a aceleração da gravidade da Terra, cujo valor é $10m/s^2$.

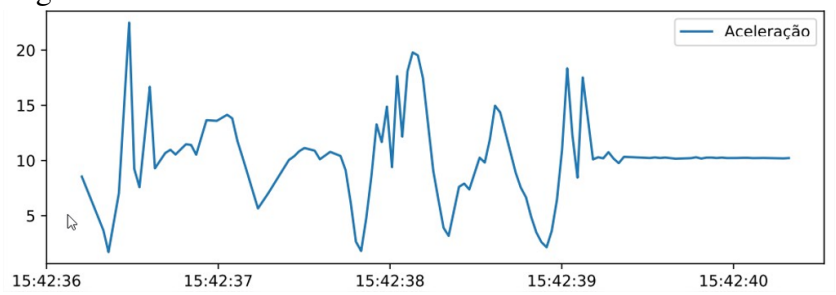
Por fim, a última análise busca identificar se o usuário encontra-se inativo, parcialmente ativo ou ativo (ARAÚJO *et al.*, 2018; ALMEIDA *et al.*, 2016; HSIEH *et al.*, 2014). Se o paciente está inativo, ele está parado, podendo estar inconsciente. Caso a indicação seja de

Figura 5 – Gráfico de uma queda com dados de acelerômetro



Fonte: O autor.

Figura 6 – Gráfico de uma atividade com dados de acelerômetro



Fonte: O autor.

parcialmente ativo, ele está com pouca movimentação. E, se o paciente está ativo, ele pode estar desempenhando outra atividade, como bater palmas, pular ou uma atividade física.

Em termos de valores, quando o usuário encontra-se no estado inativo ou parcialmente ativo, o valor da aceleração da gravida durante os próximos 1,5s devem corresponder ao valor da gravidade da Terra, $10m/s^2$. Observa-se que nos primeiros instantes, ainda há uma variação, contudo há uma estabilização do quadro do usuário. Em contrapartida, quando o usuário está ativo, o gráfico continua oscilando, como apresentado na Figura 6. Nele, é possível ver um comportamento de queda livre seguido por uma aceleração repentina para um valor maior que $20m/s^2$. Após a análise do último valor, percebe-se mais facilmente que não é uma queda.

Ainda sobre os dados, eles podem ser submetidos a algoritmos de *thresholds* (RAEVE *et al.*, 2022; RUENIN *et al.*, 2022; ARAÚJO *et al.*, 2018) ou algoritmos de Aprendizagem de Máquina (AM) (NORMAN, 2022) como em De *et al.* (2022), Raeve *et al.* (2022), Pinto *et al.* (2022), Andrade *et al.* (2021) e Linhares *et al.* (2020). Quanto aos primeiros, eles analisam se os valores encontrados ultrapassam determinados limites, sejam superiores (e.g., identificação de um pico) ou inferiores (e.g., identificação de um vale). Esses limites podem ser definidos para cada contexto e é preciso utilizar dados de quedas e atividades diárias como base.

Quanto aos algoritmos de Aprendizagem de Máquina, o mais comum é a utilização de algoritmos supervisionados, nos quais são indicados quais dados são de queda ou atividade

des diárias dos usuários. Assim, os algoritmos podem aprender o comportamento e quando analisarem novos dados, eles podem classificar em queda ou não.

2.2.2 Visão Computacional

Quando trata-se de visão computacional, ela corresponde ao processamento de imagens a fim de detectar uma queda (WU *et al.*, 2019), permitindo a identificação de múltiplos eventos simultaneamente sendo menos invasiva (MUBASHIR *et al.*, 2013) quando se utiliza câmera de profundidade (TANWAR *et al.*, 2022) no que diz respeito a ser utilizado pelo paciente. Contudo, o uso das câmeras diminui a privacidade dos pacientes (RAEVE *et al.*, 2022; ARAÚJO *et al.*, 2018), limitando os ambientes em que eles podem ser monitorados, como o banheiro. Em contrapartida, o uso de câmeras permite um monitoramento contínuo de forma passiva e a análise de mais de uma característica (TANWAR *et al.*, 2022). Esse monitoramento passivo ocorre devido ao fato de não ser precisa nenhuma interação do usuário para que ele possa ser ativado.

Analisando o que pode ser realizado a partir do processamento de imagens, em Mubashir *et al.* (2013) os autores descrevem que podem ser detectadas cinco situações, as quais podem se complementar. São elas: i) Mudança da posição do corpo; ii) Inatividade; iii) Análise da mudança da cabeça; iv) Postura; e v) Espaço-Temporal. O primeiro item, a mudança da posição do corpo, é caracterizado pela identificação de mudanças posturais ou se houve alguma mudança na cabeça utilizando quadros consecutivos de uma imagem, permitindo, assim, a identificação do momento de uma queda. Em um momento, o usuário pode estar em uma posição vertical enquanto no instante seguinte ele já se encontra em uma posição horizontal.

Ainda analisando os quadros da imagem, é possível perceber se o usuário se mostrou inativo após um evento de queda identificado pelo processamento de imagens, o que sugere a necessidade de uma assistência mais rápida para o usuário. Cabe destacar que todos os pontos citados anteriormente são dependentes do quinto item, pois todos os dados tem como base análises do usuário durante o tempo.

Para realizar o processamento das imagens, há a utilização de algoritmos extensivos executando em um computador (RAEVE *et al.*, 2022), pois é preciso lidar com grandes volumes de dados, uma vez que cada quadro pode ter diversos pixels a serem processados e é preciso mais de um quadro para identificar uma queda. Além disso, a área de cobertura das câmeras é limitada (RAEVE *et al.*, 2022), o que sugere o uso de pelo menos duas câmeras para monitorar

um usuário em cada ambiente. Os autores acrescentam que o custo do uso de câmeras é mais alto.

Por fim, não é possível identificar as possíveis causas da queda, dado que muitos fatores não podem ser observados pelas câmeras. Por exemplo, dentre os fatores extrínsecos, corrimãos e tapetes antiderrapantes não costumam ser observados pelas câmeras. E dentre os fatores intrínsecos, muitos estão relacionados às condições fisiológicas, o que também não pode ser detectado pelas imagens.

2.2.3 Ambiente/Fusão de Dados

No ambiente onde o usuário é monitorado, podem ser inseridos diversos sensores de modo que os dados coletados por eles são integrados para detectar uma queda. Exemplos de sensores que podem ser embarcados no ambiente podem ser de áudio, vídeo ou de vibração (RAEVE *et al.*, 2022; MUBASHIR *et al.*, 2013) ou mais precisamente sensores de pressão e infravermelho (WANG *et al.*, 2022). Esses sensores podem ser utilizados em conjunto visando melhores resultados na precisão da detecção de quedas.

No caso do áudio e vídeo, eles podem ser integrados para, quando a análise do vídeo identificar uma possível situação de queda, questionar o usuário se ele está bem ou se precisando de assistência médica (MUBASHIR *et al.*, 2013). Caso a resposta seja afirmativa para o auxílio ou ele não responder, o sistema pode enviar uma mensagem de alerta para os familiares. Caso contrário, o sistema continua monitorando o paciente.

No caso dos sensores de vibração, eles são alocados no solo para identificar as vibrações que ocorrem, as quais podem ser do usuário pisando no solo. Entretanto, se a vibração identificada for mais intensa, ela pode caracterizar uma queda do usuário. Além disso, é possível identificar onde o usuário se encontra, por onde ele passou, entre outras informações (MUBASHIR *et al.*, 2013). Os sensores também podem ser instalados em paredes, como um acelerômetro buscando identificar uma queda através de um algoritmo que executa em paralelo (RAEVE *et al.*, 2022) ou também na cama, entre outros lugares (TANWAR *et al.*, 2022).

Contudo, para desenvolver um sistema de detecção de quedas utilizando sensores é necessário atender condições específicas como um sensor de infravermelho que requisita um ambiente sem objetos que obstruam a captação (WANG *et al.*, 2022). No caso de sensores de pressão, é preciso inserí-los em uma grande área de captação (WANG *et al.*, 2022) para evitar que ocorra alguma imprecisão na detecção.

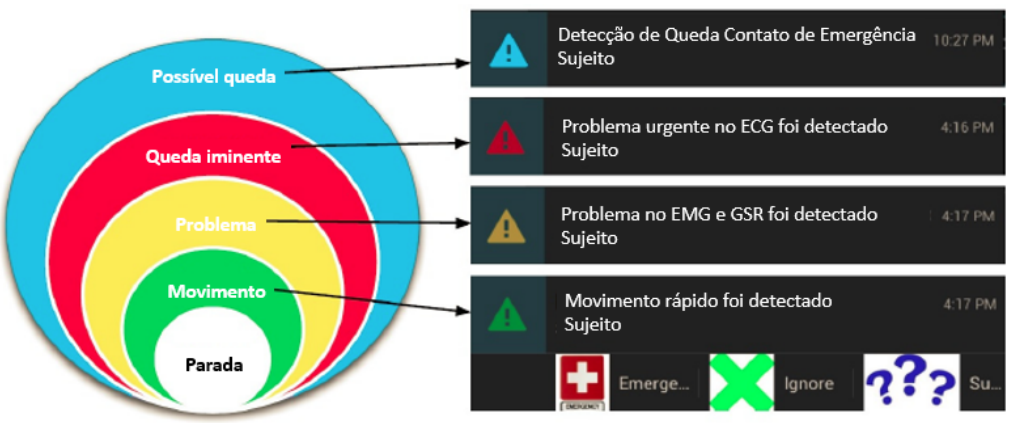
Ademais, há um custo médio a alto para a instalação dos sensores, além de não serem intrusivos para o usuário (TANWAR *et al.*, 2022), o que difere dos vestíveis e das câmeras, a depender do modelo. Outro aspecto que pesa contra esses sensores é o fato de serem complexos para configurar (TANWAR *et al.*, 2022). Assim como as câmeras, não é possível também detectar aspectos fisiológicos, impedindo a detecção de causas intrínsecas até das causas extrínsecas, uma vez que não é possível identificar elementos do ambiente.

Por fim, para ilustrar uma possível utilização dos três sensores atuando de forma integrada, tem-se que em um primeiro momento um sensor detecte uma vibração intensa no piso, o que pode acionar o sistema de detecção de quedas da câmera. Se a pessoa tiver caído, ela vai detectar uma mudança na posição do corpo do usuário, que estava vertical e passou para a horizontal. Ao final, é feita uma checagem via áudio com o intuito de confirmar com o usuário se ele está bem ou precisando de auxílio.

2.3 Associação de quedas com problemas de saúde

É importante conhecer como uma queda pode ser associada com outros problemas, dentre eles os problemas cardíacos e motores. Um exemplo dessa associação de causas e quedas é feito por um trabalho encontrado até agora na literatura, sendo o trabalho de Horta *et al.* (2015). O objetivo é detectar quedas e associar com problemas de saúde identificados a partir de um aparelho de eletrocardiograma (ECG) e de eletromiografia (EMF), sendo esse último utilizado para identificar problemas motores. A Figura 7 contém essa associação conforme apresentado pelos autores.

Figura 7 – Associação entre quedas e problemas cardíacos



Fonte: Adaptado de (HORTA *et al.*, 2015)

Os autores destacam que os usuários precisam saber bem o significado das cores e alertas para que entenda bem o que pode estar ocorrendo. Isto posto, o primeiro evento de alerta é o "Parado" (*Stopped*), o qual descreve que o usuário encontra-se em pé e sem um risco de queda. A partir desse evento inicial, foram elencados limites que caracterizavam cada um dos eventos posteriores.

O segundo alerta acontece quando está em "Movimento" (*Movement*) e significa que o sistema detectou um movimento rápido do usuário. Isso não significa que foi identificado algum risco de queda. Contudo, quando o alerta para o estado amarelo com o evento "Problema" (*Problem*) identificado, significa um sinal de perigo. Esse alerta é emitido quando o sistema detecta problemas musculares ou de equilíbrio relacionado com quedas, mas que pode ser imperceptível para o usuário. Ainda nesse cenário, que é o segundo alerta mais perigoso, o usuário deve manter a calma e tentar corrigir problemas fisiológicos ou emocionais.

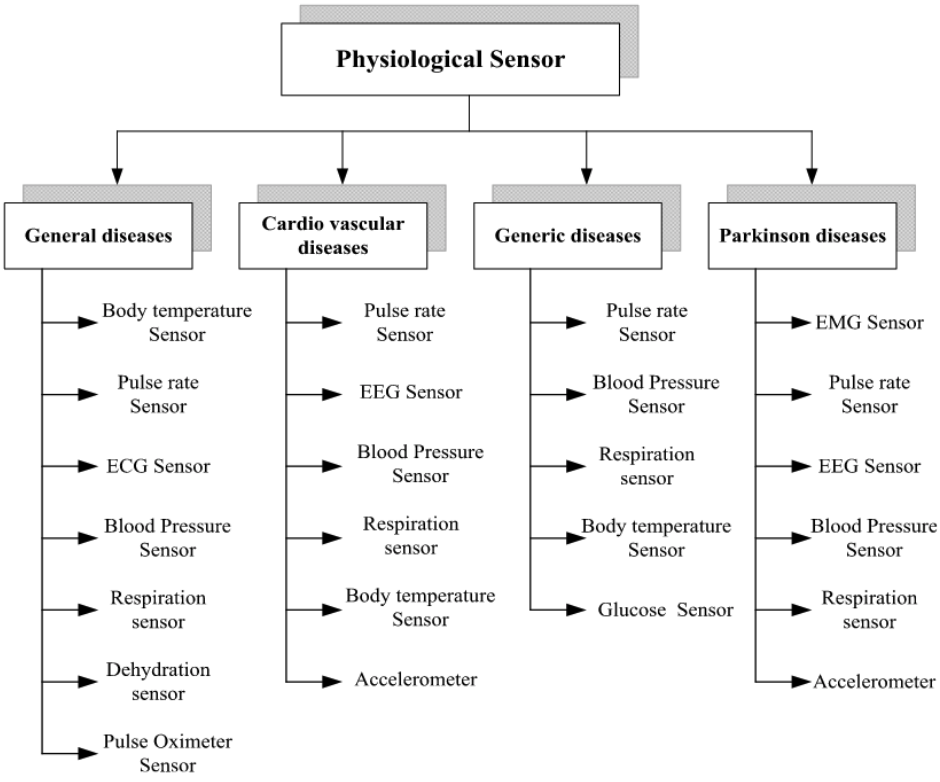
O mais perigoso alerta antes da queda é o vermelho, cujo evento é "Queda Eminente" (*Eminent Fall*), e o qual deve fazer com que o usuário procure assistência médica urgentemente, pois ele pode ter um problema não aparente no coração. Durante a análise é verifico se os batimentos cardíacos estão normais, rápidos ou lentos. Essa análise permite identificar um problema como arritmia cardíaca.

O trabalho de Horta *et al.* (2015) mostra um indicativo de que pode ser possível associar outras condições de saúde, como pressão e glicemia que são fatores intrínsecos ao paciente visando o aprimoramento da detecção de quedas, bem como das condições de saúde do paciente. Esses pontos podem ser aprimorados com o apoio de objetos inteligentes da IoT ou com sensores fisiológicos existentes como apresentado na Figura 8.

No trabalho de Sahu *et al.* (2021), é apresentada uma classificação para os sensores fisiológicos que podem ajudar a compor uma solução de IoHT. Quanto aos tipos, eles são de dez tipos distintos, os quais podem auxiliar a detectar problemas diferentes, como alguma enfermidade mais geral ou um problema cardíaco. No caso do sensor de temperatura corporal (*body temperature sensor*) é possível medir a temperatura buscando detectar problemas como hipotermia, febre, insolação, entre outros (BAKER *et al.*, 2017).

O sensor de frequência de pulso (*pulse rate sensor*) é um dos mais comuns meios de medir sinais vitais e é importante, pois auxilia na detecção de uma grande variedade de sintomas e emergências, como parada cardíaca, síncope do vaso vagal ou embolia pulmonar (BAKER *et al.*, 2017). Enquanto isso, o eletrocardiograma é geralmente usado para medir a atividade

Figura 8 – Sensores Fisiológicos



Fonte: (SAHU *et al.*, 2021)

elétrica do coração, o que contribui para avaliar a saúde do coração (KUMAR; Devi Gandhi, 2018), doenças cardiovasculares e falhas no coração (WOLGAST *et al.*, 2016). A hipertensão é a principal causa de riscos cardiovasculares e ataques cardíacos e é caracterizada pela pressão sanguínea. Para realizar essa medição, foi desenvolvido o sensor de pressão sanguínea, que ajuda a monitorar a saúde de pessoas com doenças cardiovasculares, Parkinson ou outras doenças.

Quanto à respiração, pode ser utilizado o sensor respiratório, o qual conta o número de vezes que uma pessoa respirou em um minuto (BAKER *et al.*, 2017). Essa medição contribui para associar quadros de asma, episódios de apneia, hiperventilação por ataques de pânico, entre outros. Relacionado a esse sensor, há o oxímetro, cujo papel é medir o nível de oxigênio no sangue e indica o funcionamento da respiração. Essa medida é bastante utilizada, por exemplo, em pessoas com COVID-19.

Outros dois sensores medem sinais elétricos, porém, um mede dos músculos, o eletromiograma, enquanto o outro mede do cérebro em diferentes atividades, e é chamado de eletroencefalograma (EEG). O primeiro mede os sinais dos músculos em momentos de contração e relaxamento, permitindo identificar doenças neuromusculares, por exemplo. Esse sensor foi utilizado no sistema descrito anteriormente. O EEG é mais utilizado para detectar anomalias no

sono, coma, encefalopatias, doença de Parkinson e até a saúde do cérebro.

O último sensor, uma vez que o acelerômetro já foi tratado na Seção 2.2.1, é o glicosímetro. O papel dele é medir o nível de glicose no sangue, permitindo identificar problemas relacionados à diabetes. Por exemplo, se a glicemia medida estiver baixa ou alta pode ocasionar desmaios.

Assim, percebe-se que com o avanço desses dispositivos, é possível medir diversas características fisiológicas, permitindo associar com uma queda, indicando a possível causa. Por exemplo, uma pessoa caiu e em um instante de tempo próximo a queda o usuário apresentou i) atividade cerebral irregular detectada por um EEG; ou ii) a glicemia estava bem baixa; ou iii) os batimentos cardíacos também baixos; nesses cenários, a pessoa pode ter caído por um desses problemas identificados, o que indica para a equipe médica qual outro problema deve ter que ser tratado, reduzindo, assim, o tempo de investigação da possível causa e evitando maiores sequelas.

2.4 Modelo de Dados

Muitos dados costumam ser heterogêneos, especialmente na IoT, devido à presença de diversos dispositivos distintos gerando dados, os quais indicam uma falta de padronização e que evita que se obtenha uma representação consistente do conhecimento inteiro (REDA *et al.*, 2018). Visando a resolução de tais problemas, um modelo de dados pode ser utilizado, contudo, é preciso observar as particularidades de cada um, a fim de escolher o que melhor atende as necessidades de cada aplicação.

Na literatura foram encontradas várias formas de representação de dados, sendo as principais chave-valor, linguagem de marcação, baseado em lógica, orientação a objeto e ontologias, como descrito em Kumar (2015). O modelo chave-valor é o modelo mais simples, sendo necessário um valor a ser registrado e uma chave, cujo papel é permitir o acesso ao valor armazenado (CATTELL, 2011). Esse tipo de modelo de dados é bastante utilizado em bancos de dados não relacionais (MARINESCU, 2022) também conhecidos como NoSQL (STONEBRAKER, 2010). Alguns exemplos desses banco de dados são Voldermort, Cassandra (POKLUDA; SUN, 2013), MongoDB (BRADSHAW *et al.*, 2019).

No caso do *Markup Schema* ou linguagens de marcação, elas consistem de elementos, os quais podem ser aninhados, e são definidos em forma de *tags* com marcadores de início e fim (BOLEY, 2003). Um exemplo é <*tag*>...</*tag*>, sendo o início a partir de <*tag*> e o fim com

</*tag*> e os valores representados e que podem ser utilizados o que estiver entre as *tags*.

Exemplos de linguagens de marcação são i) *Extensible Markup Language* (XML) (ALMEIDA, 2002); ii) *HyperText Markup Language*(HTML) (BOUVIER, 1995), a qual foi criada inicialmente para a acessar a Web e hoje encontra-se na sua quinta versão, o HTML5 (SILVA, 2019); iii) *extensible access control markup language*(XACML) (ANDERSON *et al.*, 2003) e que foi utilizado em trabalhos da literatura como o de Li *et al.* (2021).

Já a representação do conhecimento baseado em lógica utiliza operadores lógicos para especificar os dados e as relações entre eles (KUMAR, 2015) e são utilizadas quando deseja-se descrever alguma abstração de um domínio de interesse e é feita utilizando elementos (BAADER *et al.*, 2017), como no trabalho de Morin *et al.* (2009). Nele é definida uma representação do conhecimento baseada em lógica de primeira ordem voltada para a identificação de intrusão em redes. Um exemplo de expressão utilizada no trabalho de (MORIN *et al.*, 2009) é descrita em seguida, cujo objetivo é registrar o nó H na rede N e, para isso, precisa do endereço IP do nó H e da rede N.

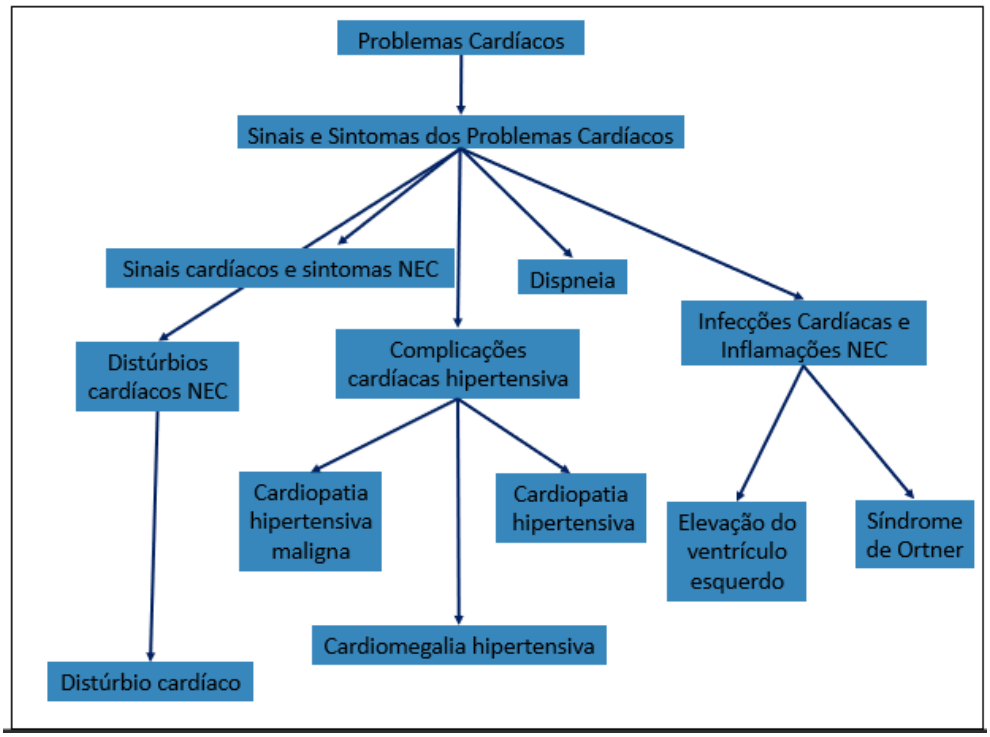
$$nodenet(H,N) \rightarrow nodeaddress(H,A_H) \wedge netaddress(N,A_N) \wedge matches(A_H,A_N)$$

A forma de representação de orientação a objeto descreve o uso de objetos para representar diversos tipos de contextos diferentes que encapsulam processamento e representação, além de permitir que os dados possam ser acessados utilizando interfaces bem definidas (KUMAR, 2015). Esse tipo de modelo de dados considera as características da orientação objeto, como generalização-especialização, agregação e associação, o que define uma estrutura hierárquica também nos dados (VALIANTE *et al.*, 2021). Exemplos desse tipo de banco de dados são o db4o (PATERSON *et al.*, 2006) e GemStone (BUTTERWORTH *et al.*, 1991).

A última maneira de representar dados é a ontologia. Ela especifica explicitamente um conceito (GRUBER, 1993) descrevendo-o formalmente juntamente com os relacionamentos entre os conceitos, que costumam ser de um domínio específico (Dou *et al.*, 2015), organizando-os de uma forma hierárquica (SHAHZAD *et al.*, 2021). Um exemplo de ontologia é a Medical Dictionary for Regulatory Activities Terminology (MedDRA) (MEDDRA, 2021), que é uma ontologia para problemas de saúde. Um trecho dela pode ser visualizado na Figura 9 com diretrizes sobre desordens cardíacas, como sinais e sintomas de cada uma tendo como exemplo complicações hipertensivas, inflamações e infecções cardíacas e dispneia.

No trabalho de Kumar (2015) é encontrada uma comparação entre os modelos de dados aqui apresentados e tal comparação foi retratada na Tabela 1. Os modelos de dados

Figura 9 – Ontologia MedDRA



Fonte: Adaptado de (MEDDRA, 2021)

descritos anteriormente foram classificado com base em quatro critérios que são explicitados aqui. O primeiro deles trata da simplicidade, cujo objetivo é avaliar as expressões utilizadas em cada modelo, analisando se são simples, fáceis de entender para serem usadas pelos desenvolvedores.

Analisando a simplicidade, o autor classificou que os modelos mais fáceis de entender são o de chave-valor, a linguagem de marcação por ter como base o XML e a ontologia. Os modelos baseado em lógica e orientado a objeto são mais difíceis, sendo o primeiro mais difícil pra implementar, uma vez que é preciso conhecimento de lógica de primeira ordem enquanto o segundo apresenta mais dificuldade para gerenciar os dados armazenados.

O segundo critério de avaliação trata da flexibilidade dos modelos. O foco é avaliar a facilidade do modelo evoluir, permitindo a inclusão de novas entidades e relacionamentos entre os dados. Assim, ao observar esse critério, o modelo chave-valor se mostra como o mais difícil de ser utilizado, uma vez que, segundo o autor, é difícil criar novos modelos a partir de outros já existentes. Enquanto que as linguagens de marcação são flexíveis permitindo, por exemplo, a conversão de XML para RDF.

Ainda segundo Kumar (2015), a forma baseada em lógica é flexível, pois pode ser facilmente implementada entre máquinas. O modelo orientado a objeto, por sua vez, também pode ser facilmente modificado permitindo a evolução ou alteração nos dados dependendo das necessidades. As ontologias também foram classificadas como flexíveis devido a facilidade de

reutilização de um conhecimento sobre um domínio previamente definido.

Tabela 1 – Comparativo entre os modelos de dados

Critério	**Modelos de Dados**				
	Chave-valor	**Linguagem de Marcação**	**Baseado em Lógica**	**Orientado a objeto**	**Ontologia**
Simplicidade	É simples utilizar	Implementação baseada em XML sendo, portanto, simples	Modelo bem desafiador para implementar	Dados de contexto como objetos são difíceis de gerenciar	É simples criar e manter dados contextuais usando ontologias
Flexibilidade	Difícil de criar modelos a partir de outros existentes	Suporta total flexibilidade para o RDF	Ele pode ser implementado entre máquinas	Modelo OO pode ser facilmente modificado	Conhecimento do domínio pode facilmente ser reusado
Generalidade	Suporta consulta de padrões	Tipos de informações de contexto são limitadas	Difícil construir nova lógica a partir de uma existente	Objetos são reusáveis	Novas ontologias podem facilmente ser criadas a partir de ontologias já existentes
Expressividade	Não suporta algoritmos complexos	Problemas de performance com grande volume de dados	Falta expressividade para apoiar a qualidade da informação	Consultar os objetos para obter resultados desejados é entediante	Suporta análise facilmente

Fonte: Adaptado de (KUMAR, 2015)

O terceiro critério de avaliação é a generalidade e descreve que um modelo de dados não deve ser restrito a um tipo de informação de contexto. Assim, permitir o uso de vários tipos de informações de contexto. Ela é melhor executada nos modelos orientado a objetos e nas ontologias. Isso ocorre devido ao fato dos objetos presentes nos primeiros serem reutilizáveis. Enquanto novas ontologias podem ser criadas a partir de outras já existentes de uma forma mais fácil. As demais possuem mais limitações, que estão relacionadas aos tipos de informações coletadas e até dificuldades para estender os modelos.

Já a expressividade está ligada à quantidade de detalhes que podem ser expressados por cada um dos modelos para representar cada informação utilizada. A expressividade é maior nas ontologias, pois permitem estabelecer relações entre os conceitos. Os demais modelos são mais limitados com fatores que englobam a falta de suporte a algoritmos complexos como o modelo de chave-valor, problemas de desempenho nas linguagens de marcação e a qualidade da informação pode ser restrita no modelo baseado em lógica.

2.5 *Frameworks*

Diante do contexto da Internet das Coisas, a qual permite que diversos dispositivos distintos possam se conectar à Internet utilizando diferentes meios de comunicação e protocolos diversos, como MQTT (STANDARD, 2019; OASIS Standard Incorporating Approved, 2015), CoAP (SHELBY *et al.*, 2014), ZigBee (ERGEN, 2004). Esses protocolos são utilizados em aplicações com objetivos variados, além de utilizarem sensores que coletam dados em formatos diferentes, como apresentado na Seção 2.4. Para minimizar o impacto dessa complexidade no desenvolvimento de novas aplicações e serviços, é possível reutilizar artefatos de software (ARAÚJO, 2018).

Nesse contexto de reutilização de artefatos de software, quatro deles são os mais comuns (ARAÚJO, 2018) e que permitem planejar mais adequadamente uma aplicação IoT desde o seu início. Esses quatro artefatos são arquitetura, *middleware*, *frameworks* e padrões de software, os quais já possuem exemplos para a Internet das Coisas, como é descrito em Araújo (2018). Esses artefatos ajudam a reduzir os custos e o tempo (TAHIR *et al.*, 2016) de desenvolvimento, facilitando, assim, a criação de uma nova aplicação. A importância para IoT é ainda maior, uma vez que ela engloba diversas áreas da computação, como descrito no Capítulo 1 e permite que o desenvolvedor consiga focar nos aspectos relativos ao domínio em que está implementado a aplicação, por exemplo, no domínio da saúde.

Embora existam outros artefatos de reuso, esta tese descreve apenas os *frameworks*, foco deste trabalho. Um *framework* corresponde a trechos de código semi-prontos e aptos para serem usados (PREE, 1995), que implementam funções comuns para facilitar o desenvolvimento de determinadas estruturas de código, abstraindo do programador detalhes de implementação, necessitando que esse apenas faça ajustes para a sua aplicação. Por exemplo, uma aplicação pode usar um *framework* para facilitar a conexão com um servidor ou para executar processamento dos dados.

Também é importante conhecer os processos de desenvolvimento de um *framework* encontrados na literatura. Em Koskimies e Mössenböck (1995), os autores propõem um processo de duas fases para projetar uma versão inicial de um *framework*. A primeira fase é a generalização do problema e inicia com a especificação de uma aplicação representativa do *framework* desejado e é feita uma generalização em uma sequência de passos até a forma mais geral, sendo esse processo feito em níveis.

A segunda fase presente em Koskimies e Mössenböck (1995) é o projeto do *fra-*

mework e nela ocorre a implementação da generalização feita na fase anterior em ordem reversa dos níveis especificados. Assim, a implementação de um nível mais abaixo pode ser utilizada em um nível mais acima. Os autores ainda citam que geram uma hierarquia de *frameworks* que ficam mais refinados a cada nível.

Em Yang *et al.* (1998), os autores propõem um processo de desenvolvimento de framework e citam que existem outros processos de desenvolvimento, sendo um baseado na experiência com aplicações e o outro na análise do domínio, propostos por Wilson e Wilson (1993).

O processo proposto em Yang *et al.* (1998) consiste de quatro fases, similares ao processo de desenvolvimento de software tradicional, i) análise, ii) projeto, iii) implementação e iv) teste. Na fase de análise é feita a especificação de requisitos através da extração de funcionalidades comuns a partir de um conjunto de especificações de requisitos de aplicações similares. A fase de projeto identifica as classes e os *hot spots*, elementos que variam em cada aplicação, determinando se vai ser utilizada uma abordagem caixa preta, caixa branca ou híbrida para projetar o *framework*. A fase de implementação é aplicada repetida e incrementalmente, assim, em cada ciclo de desenvolvimento novas funcionalidades são implementadas no *framework*. Cada iteração foca em um pequeno conjunto de requisitos, passando por todas as etapas de desenvolvimento. Por fim, a última etapa testa o *framework* para saber se ele atende aos requisitos especificados.

Já o primeiro processo proposto em Wilson e Wilson (1993) consiste primeiro no desenvolvimento de aplicações de um domínio, que podem ser duas (2), e, a partir de então, passar a identificar as características comuns em ambas aplicações e extraí-las em um *framework*. Em seguida, as aplicações devem ser reconstruídas utilizando o *framework*.

O outro processo de Wilson e Wilson (1993), análise de domínio, consiste em analisar o domínio do problema para identificar e entender abstrações bem conhecidas. Essa análise extrai abstrações de aplicações existentes que são elementos comuns a todas e a partir delas, é possível desenvolver o *framework* junto com uma aplicação de teste. Alterações no *framework* podem ocorrer se for identificado nessa última etapa. Em seguida, é construída uma segunda aplicação com o *framework*.

O trabalho de Stanojević *et al.* (2011) define que o processo de desenvolvimento seguido por ele é composto por três fases, i) desenvolvimento do *framework*, ii) uso do *framework* e iii) evolução e manutenção do *framework*. A primeira fase é composta por cinco sub-fases -

análise de domínio, projeto do *framework*, implementação, teste do *framework* e documentação do *framework*. A análise do domínio consiste de análise de três aplicações do domínio para definir a arquitetura geral das aplicações. Ao final da sub-fase devem ser definidas as classes comuns e os *hot spots*. O projeto do *framework* define a arquitetura, um dos mais importantes pontos, pois determina as transformações necessárias para os *hot spots*. Em seguida, deve ser feita a implementação do *framework*, seguido pelo teste dele. Por fim, é feita a documentação do *framework*. A segunda fase, o uso, executa um estudo de caso utilizando o *framework* e uma visão geral é mostrada. Por fim, deve ser realizada a evolução e manutenção do *framework*.

Para a IoT, já existem frameworks para atender a diversas características comuns da área. Por exemplo, o *framework* IoTivity (IOTIVITY, 2021; IOTIVITY, 2017), que é *open source* e tem como objetivo estabelecer a conexão dispositivo-dispositivo. O IoTivity é composto por três camadas: a camada de serviço, a camada base e a interface de Cloud. A camada base e a interface de Cloud tem os mesmos componentes, porém com diferenças entre eles, uma vez que executam em dispositivos distintos. A camada de serviço trabalha com o encapsulamento de recursos, permitindo a descoberta dinâmica deles.

O Gobot (GROUP, 2022; GROUP, 2017) é um *framework* desenvolvido para a manipulação de robôs e para a IoT escrito na linguagem Go. Dentre as principais características, cita-se a presença de *drivers* e adaptadores para controlar hardwares como Arduino (MCROBERTS, 2018; EVANS *et al.*, 2013), Raspberry Pi (OLIVEIRA, 2017; RICHARDSON; WALLACE, 2013), drones e outros dispositivos que podem ser utilizados para compor uma solução IoT.

Em Corallo *et al.* (2022), é apresentado um *framework* de *Big Data Analytics* como serviço baseado em modelos em um ambiente de manufatura inteligente. O trabalho de Yang *et al.* (2022) propôs um *framework* de compartilhamento de dados de borda distribuída, combinando a computação de borda e *blockchain* para resolver problemas de compartilhamento de segurança entre aplicativos de borda na *Industrial Internet of Things* (IIoT).

2.6 Conclusão

Este capítulo apresentou a fundamentação teórica necessária para a compreensão do trabalho aqui proposto. Primeiro, dentre os temas abordados, discute-se a Internet das Coisas, como primeiro item, sendo explanado na seção 2.1. Nela são detalhados os conceitos e como a IoT pode auxiliar na área da saúde com monitoramentos dos pacientes, especialmente na detecção

de quedas e como informações fisiológicas podem ser associadas buscando um atendimento mais adequado.

Também foram abordados os modelos de dados que podem existir na literatura, como modelos chave-valor, descrições lógicas, linguagens de marcação, orientados a objeto e ontologias. Essa última forma de representar o conhecimento permite uma ampliação dos conceitos de modo mais fácil do que os demais e permite a criação de regras de inferência, aumentando a possibilidade de extração de conhecimento.

Por fim, foram apresentados os *frameworks*, além de exemplos deles. Os *frameworks* são o foco deste trabalho e permitem a construção de códigos de tal forma que várias aplicações podem utilizar, diminuindo o esforço de construção de uma solução, permitindo que o desenvolvedor possa focar nas regras relativas ao domínio da aplicação desenvolvida.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados a esta tese, encontrados durante a revisão da literatura. Eles foram classificados em duas categorias relacionadas à IoHT: *frameworks* ou modelo de dados.

As seções seguintes deste capítulo são assim divididos: na Seção 3.1, são apresentados os *frameworks* encontrados; a Seção 3.2 contém os modelos de dados encontrados; uma discussão acerca dos resultados é apresentada na Seção 3.3; a Seção 3.4 conclui o capítulo.

3.1 *Frameworks* para Saúde

Como descrito na Seção 1.5, foi realizada uma revisão da literatura baseada em uma revisão sistemática, a qual buscou encontrar trabalhos na literatura cujo objetivo era apresentar um *framework* para o desenvolvimento de aplicações de Internet das Coisas voltado para a área de IoHT. Além disso, como *frameworks* possuem um caráter mais prático e relacionado ao desenvolvimento, também foram selecionados outros *frameworks* relevantes, mas que não possuem publicações relacionadas.

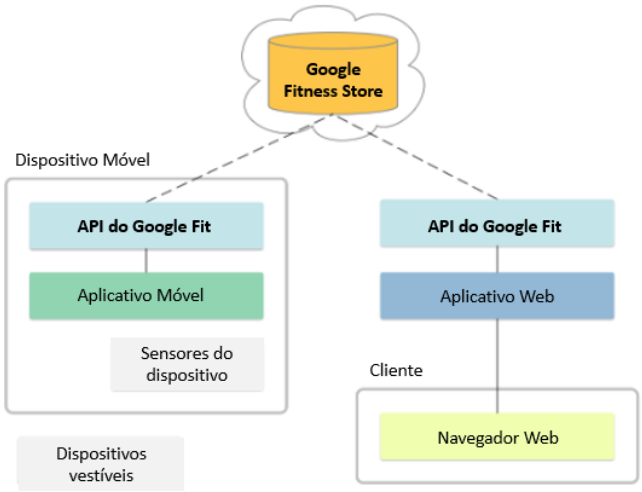
Na busca por *frameworks* para Internet of Health Things e *healthcare*, foram encontrados diversos deles, sendo três já em uso no mercado e foram desenvolvidos por grandes empresas que são o Google Fit (GOOGLE, 2022), Apple HealthKit (APPLE, 2022a) e Withings (NOKIA, 2022), sendo esse último o resultado dos esforços da Nokia. Analisando-os, verificou-se que eles são mais focados na coleta de dados e armazenamento de dados a partir de diferentes dispositivos. Entretanto, o Withings é dedicado aos produtos da empresa desenvolvedora, enquanto os demais são mais genéricos. Ainda quanto a eles, cada um possui suas particularidades, como as linguagens de programação utilizadas.

- Google Fit: Java e Kotlin
- Apple HealthKit: Swift e Objective-C
- Withings: Php

Além disso, o Google Fit possui um modelo de dados próprio com foco no armazenamento e disponibilização pública deles. Esses dados são armazenados na *fitness store* e podem ser acessados através do *framework* de sensor com representações de alto nível, como fonte de dados, tipos de dados, pontos de dados, *dataset* e sessões. A interação entre os componentes do Google Fit são apresentadas na Figura 10. As fontes de dados representam os sensores e

consistem em um nome, o tipo de dado coletado e outros detalhes. Os tipos de dados representam os diferentes tipos de dados de saúde e bem-estar como quantidade de passos e batimentos cardíacos. Esses dados consistem em um nome e a lista dos dados que podem ser variados, como um GPS, o qual possui três dados, e o peso com um.

Figura 10 – Arquitetura do Google Fit



Fonte: Adaptado de (GOOGLE, 2022)

Os pontos de dados complementam os dados coletados, pois representam os timestamps dos dados coletados. Além disso, esses pontos de dados podem ter um instante de coleta ou um *range*. Um *dataset* representa um conjunto de dados do mesmo tipo de uma mesma fonte durante um intervalo de tempo. Por fim, uma sessão representa um intervalo de tempo em que um usuário está realizando alguma atividade física.

O Apple HealthKit é um repositório central de dados para dispositivos da Apple, como iPhones e Apple Watches. Além de armazenar, ele permite visualizar e analisar os dados coletados, além de ser colaborativo para melhorar a experiência dos usuários. Como o Google Fit, ele possui o próprio modelo de dados e ele pode lidar com diferentes tipos de dados, como tipo sanguíneo, calorias consumidas, análise do sono. Para lidar com diferentes tipos de dados, o HealthKit organiza-os em classes diferentes, como dados que não mudam com o decorrer do tempo, numéricos ou categóricos, atividade executada, audiograma, eletrocardiograma, entre outros.

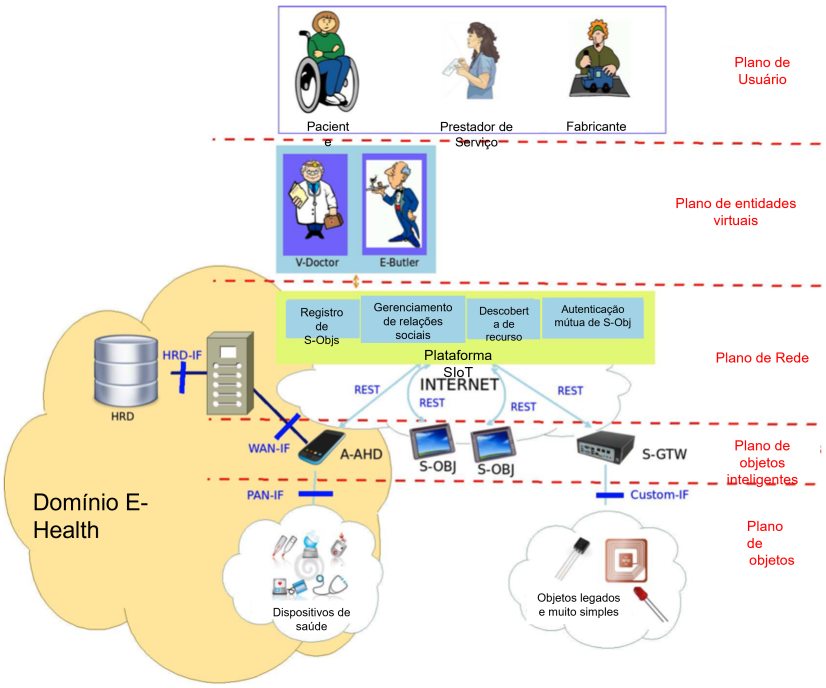
O Withings também possui seu modelo de dados para armazenamento, mas ele é limitado a alguns tipos de dados, como peso, altura, distância, temperatura, além de citar que está em evolução para aceitar o acelerômetro e um sensor para medir o volume de sangue nas

veias. Ademais, os dados são enviados via JSON, que é próximo ao modelo chave-valor.

Embora existam esses *frameworks* disponíveis no mercado, há outras iniciativas na literatura que desenvolvem soluções de reuso com diferentes objetivos. Por exemplo, o trabalho de RUGGERI e BRIANTE (2017) focou na descoberta dinâmica e automática de novos dispositivos que estão perto do paciente e não nos dados armazenados como os *frameworks* anteriores. Nesse cenário, os autores propõem um *framework* baseado no paradigma de IoT Social, o qual prescreve que os objetos podem estabelecer uma rede social entre si, facilitando a descoberta de novos recursos.

Para alcançar o objetivo, eles propõe uma arquitetura de cinco camadas baseada na definição da *Continua Health Alliance* (HIMSS, 2017) para o *framework* e é apresentada na Figura 11. A primeira das camadas contém objetos sem inteligência, de forma que não conseguem estabelecer relações sociais entre si, o que é feito na camada "plano de objetos sociais" a partir de objeto inteligentes. A camada superior contém um *middleware* que registra os dispositivos, gerencia as relações sociais, permite a descoberta de serviços e autenticação mútua entre os dispositivos. As duas últimas camadas focam no processamento dos dados e atuação no ambiente do usuário quando esse não puder atuar diretamente.

Figura 11 – *Framework* SIoT



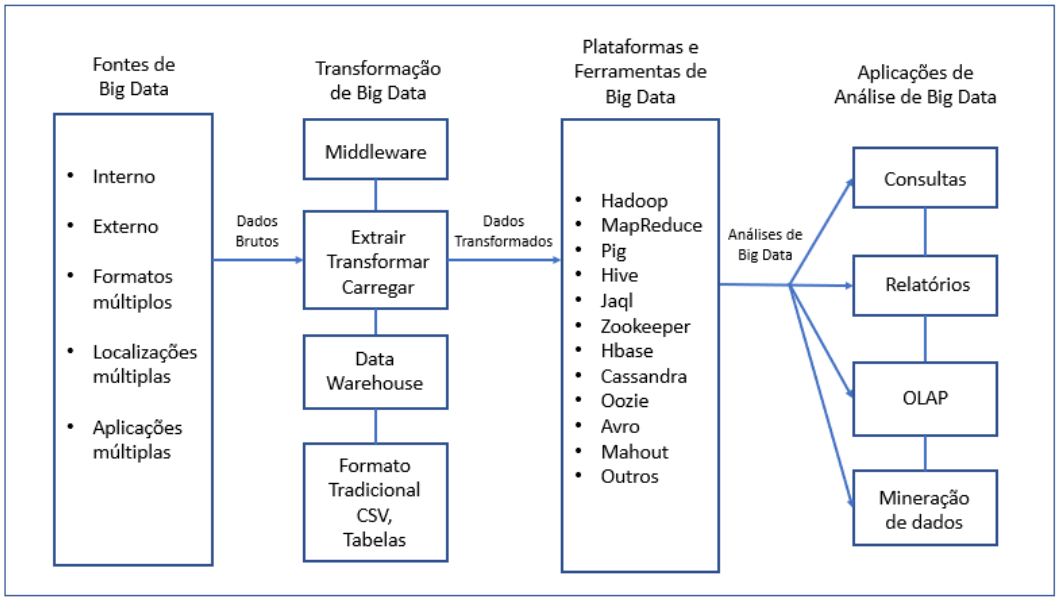
Fonte: Adaptado de (RUGGERI; BRIANTE, 2017)

Os trabalhos de Raghupathi e Raghupathi (2014) e Fang *et al.* (2016) são *frameworks*

conceituais focados no processamento de dados de saúde. O primeiro propõe um *framework* conceitual focado em análise de *big data*, a qual, segundo os autores, é similar ao processamento em outros sistemas de *healthcare*, porém com um volume maior. O *framework* é apresentado na Figura 12 e inicia com a identificação das fontes de dados utilizados, as quais podem ser registros eletrônicos de dados, dados de sistemas de apoio à decisão clínica, fontes governamentais, laboratórios, farmácias. Além disso, podem possuir diferentes formatos e estarem em diferentes locais.

Em seguida, os autores propõem que os dados brutos devem ser transformados ou agregados quando eles são de fontes diferentes. No próximo componente, os dados são submetidos a ferramentas e plataformas para design distribuído, armazenamento e de análises. Por fim, os dados são enviados para as aplicações para serem utilizados.

Figura 12 – *Framework* proposto por (RAGHUPATHI; RAGHUPATHI, 2014)



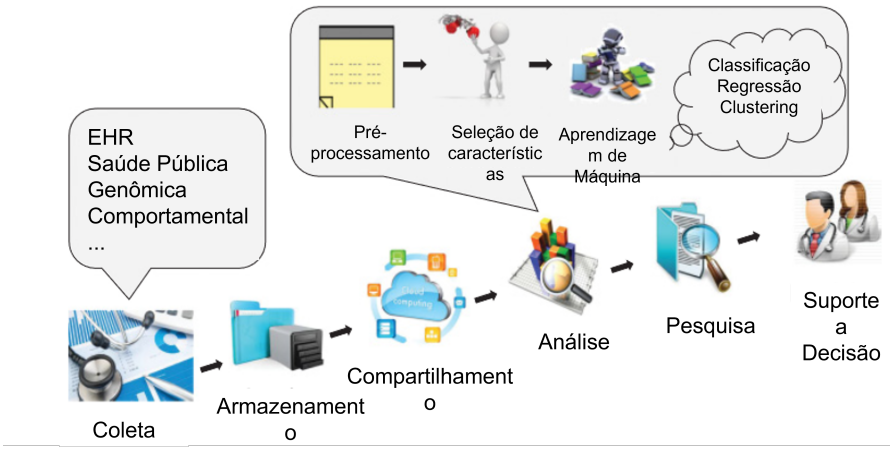
Fonte: Adaptado de (RAGHUPATHI; RAGHUPATHI, 2014)

No trabalho de Fang *et al.* (2016) é proposto um *framework* conceitual que é uma *pipeline* para o processamento de dados de saúde e consiste de seis fases: captura, armazenamento, compartilhamento, análise, busca e apoio à decisão. A captura consiste da coleta de dados, que podem ser dos mais variados tipos, não restringindo-se à sensores, enquanto o armazenamento é responsável pelo armazenamento efetivo dos dados e o compartilhamento é troca segura entre cientistas e médicos.

A análise consiste da execução de atividades de pré-processamento, seleção de características e aplicação de algoritmos de Aprendizagem de Máquina. Associada à essa etapa,

há a busca para extrair padrões significativos de interesse a partir dos resultados das análises. Por fim, há a tomada de decisão efetiva.

Figura 13 – *Framework* proposto por (FANG *et al.*, 2016)



Fonte: Adaptado de (FANG *et al.*, 2016)

O JEMF (MACHADO *et al.*, 2014) é outro *framework* encontrado na literatura, contudo o objetivo é o desenvolvimento de sistemas de emergências, os quais possuem diversos atores interagindo, como as vítimas de acidentes, testemunhas, equipes de atendimento, equipes de gerenciamento dos atendimentos, e busca facilitar a comunicação entre todos eles. Esse *framework* foca na construção de novas soluções móveis Android. A ideia é permitir que as emergências sejam registradas e possam acompanhar localização, atendimento, insumos médicos, visando prestar a melhor assistência.

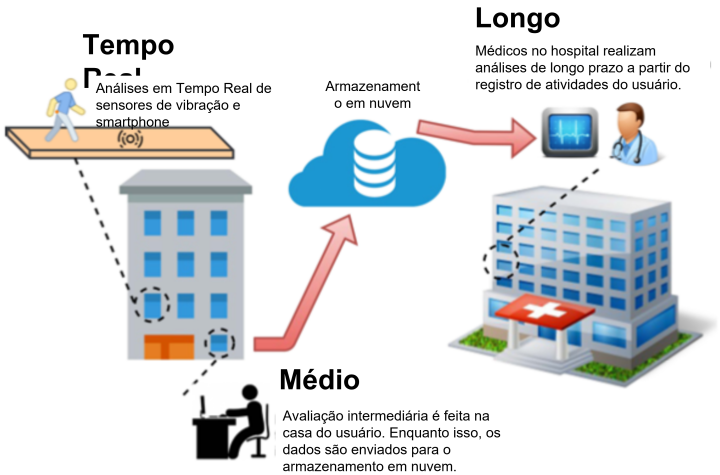
Esse *framework* foca em cinco funcionalidades comuns para esse tipo de sistemas, que são comunicação, persistência, integridade, interoperabilidade e disponibilidade. A primeira consiste da comunicação entre todos os atores, enquanto a segunda descreve que ele suporta o armazenamento dos dados, os quais devem ser íntegros, embora possa ter acessos múltiplos. Além disso, eles devem ser de um formato aberto, de forma a ser compartilhado por diversos dispositivos e devem estar disponíveis para todos que precisarem, em qualquer lugar e a qualquer tempo.

O *framework* proposto por Hemmatpour *et al.* (2018) possui um objetivo distinto dos demais já apresentados, que é detectar quedas e também preveni-las utilizando um composição de dispositivos, como *smartphones* e sensores do ambiente. Contudo, ele é um *framework* conceitual, portanto, não foi instanciado. Ele é composto por cinco componentes, os quais são apresentadas na Figura 14. O primeiro deles visa o monitoramento do usuário com os sensores

embarcados no ambiente a fim de acompanhar a movimentação do usuário. O segundo realiza uma análise em tempo real com algoritmos leves no *smartphone* a fim de detectar uma queda.

O terceiro componente é um *gateway*, cujo objetivo é realizar uma análise de prazo médio e mais robusta do que a anterior a fim de detectar uma queda, agregando os dados dos diferentes dispositivos em uma matriz. O quarto executa na nuvem a partir dos dados recebido do componente anterior enquanto o quinto consiste de uma análise por especialistas a fim de validar as ocorrências identificadas.

Figura 14 – *Framework* proposto por (HEMMATPOUR *et al.*, 2018)



Fonte: Adaptado de (HEMMATPOUR *et al.*, 2018)

3.2 Modelos de dados

Os modelos de dados podem ser chave-valor, linguagem de marcação, orientado a objeto, baseado em lógica e ontologias. Como visto na Seção 3.1, três dos frameworks apresentados possuíam um modelo de dados associado, sendo dois orientados a objeto, o Google Fit e o Apple HealthKit, e um modelo chave-valor, o do Withings.

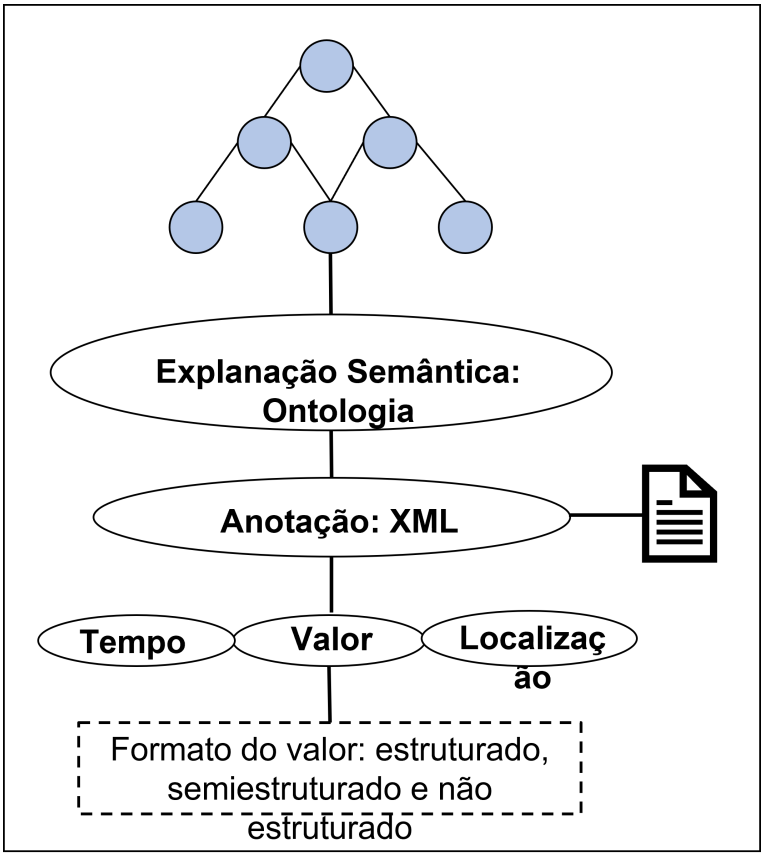
Além desses modelos, outros foram encontrados na literatura, os quais são, na sua maioria, ontologias para sistemas de saúde, como os trabalhos de Xu *et al.* (2014), Kumar (2015), Alamri (2018), Brouwer *et al.* (2018), Reda *et al.* (2018) e Carbonaro *et al.* (2018). Adicionalmente, é abordada nesta Seção, a ontologia encontrada em W3C (2018), cujo objetivo é descrever sensores, um dos focos deste trabalho.

O trabalho de Xu *et al.* (2014) propõe um modelo de dados para sistemas médicos baseados em serviços de sistemas de saúde de urgência, como ambulância ou com os médicos

em caso de acompanhamento durante um tratamento ou ainda em situações pós-tratamento (e.g., cobrança). A Figura 15 contém o modelo descrito e é dividido em três camadas, sendo uma de valor, outra de anotação e outra de explicação semântica. O valor refere-se ao intervalo dos dados que refletem o estado atual do paciente, pois indicam o valor, um tempo que representa o instante em que o dado foi coletado e a localização de onde o dado foi coletado. As anotações referem-se a uma descrição dos dados para recuperação em buscas, enquanto a explicação semântica refere-se às definições comuns dos dados para o propósito de compartilhamento.

Embora o foco seja em aplicações de saúde, a ontologia é direcionada para aplicações de gestão de saúde, envolvendo desde a coleta de dados referentes a saúde de um paciente em uma ambulância até a cobrança pelos custos do tratamento dele. Assim, outras situações de monitoramento do usuário não são contempladas por ela.

Figura 15 – Modelo de dados do trabalho (Xu *et al.*, 2014)

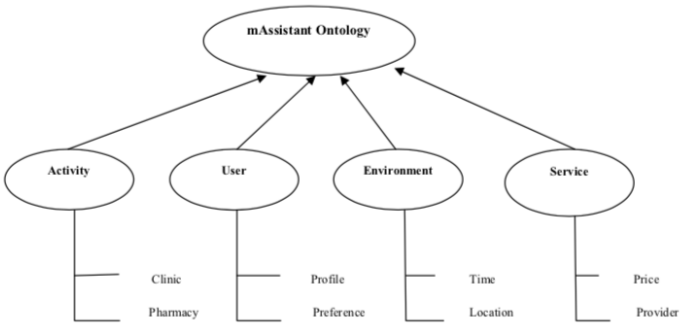


Fonte: Adaptado de (Xu *et al.*, 2014)

O trabalho de Kumar (2015), por sua vez, definiu uma ontologia para recomendações em sistemas de saúde. Essa ontologia é composta de dois níveis hierárquicos, sendo um superior e outro inferior, no qual o superior é uma ontologia de alto nível que captura características gerais do contexto, como atividade, usuário, ambiente e serviço. O nível inferior consiste da

especificação dos conceitos representados na camada superior, bem como suas características que podem ser o perfil ou preferências do usuário. Essa ontologia é apresentada na Figura 16.

Figura 16 – Ontologia definida no trabalho de (KUMAR, 2015)



Fonte: Adaptado de (KUMAR, 2015)

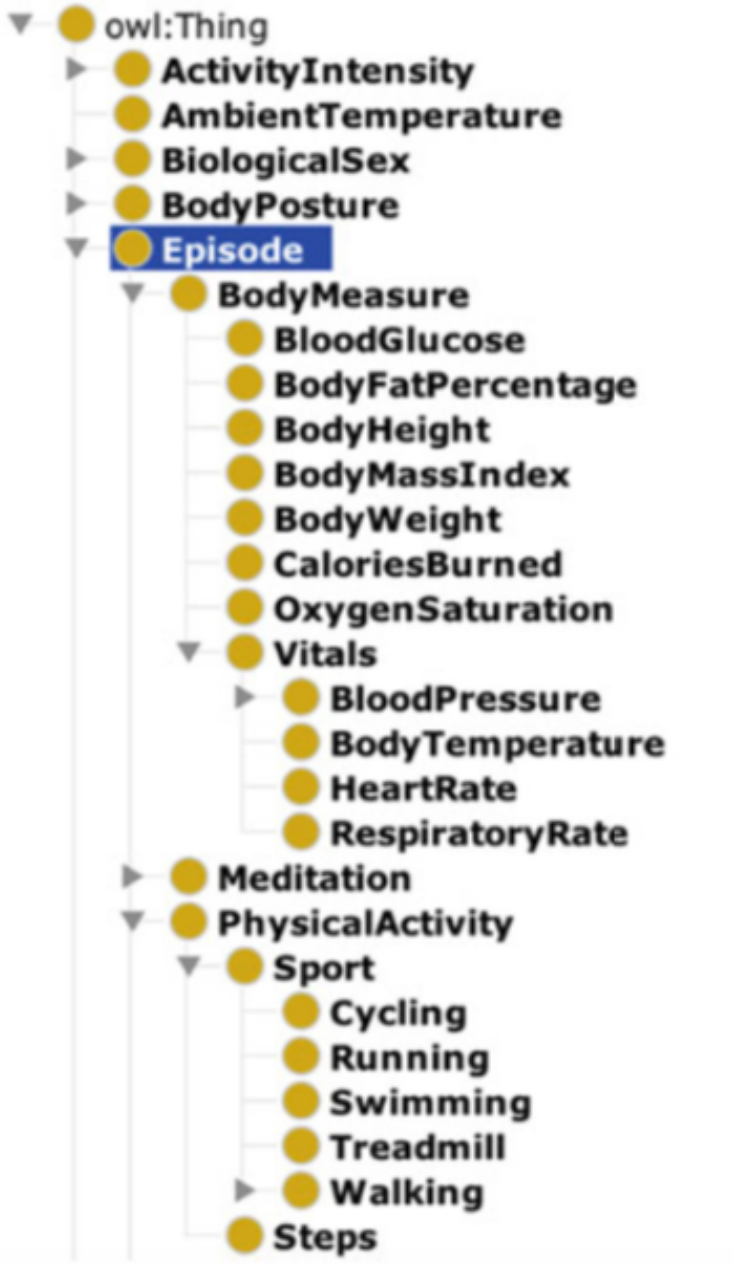
A ontologia *IoT Fitness Ontology* (IFO) é uma ontologia proposta nos trabalhos de Reda *et al.* (2018) e (CARBONARO *et al.*, 2018), cujo objetivo é registrar dados fisiológicos (e.g., batimentos cardíacos), antropométricos (e.g., altura e peso) e de atividade física dos usuários. A representação dela é demonstrada na Figura 17. É importante ressaltar que essa é uma das duas ontologia especificada focadas em IoHT, sendo as demais para outros contextos de saúde. Entretanto, não há muitos detalhes sobre ela, como mais informações sobre os dados, hierarquia, tipos deles.

As ontologias propostas por Alamri (2018) e Brouwer *et al.* (2018) são baseadas na Semantic, Sensor, Network (SSN) (W3C, 2018). Assim, primeiro é descrita a SSN, que é uma ontologia destinada a lidar com sensores, uma vez que eles são a principal fonte de dados na Web atualmente, e atuadores. A Figura 18 contém o modelo definido para a coleta de dados e inicia com a plataforma, cujo significado é o de um dispositivo. Essa plataforma contém sensores que realizam observações, que são compostas por outros sete (7) dados. Eles indicam a forma como foram coletados, a característica de interesse do sensor, a propriedade observada, os dados, o tempo (instante ou período) e o resultado.

Como descrito anteriormente, essa ontologia pode contribuir com a IoT, dado que o foco dela são sensores, contudo é genérica e deve ser aprimorada para cada contexto. Além disso, ela envolve apenas dados brutos coletados pelos sensores, o que requisita um esforço para tratá-los e torná-los mais significativos.

Após o entendimento da SSN, é possível explicar as ontologias propostas em Alamri (2018) e Brouwer *et al.* (2018), as quais são demonstradas nas Figuras 19 e 20. A de Alamri

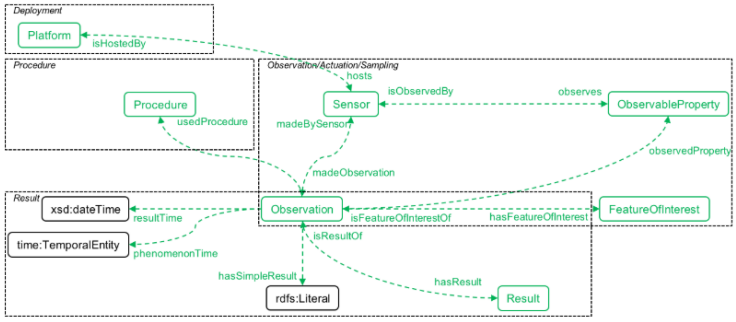
Figura 17 – Estrutura da ontologia definida no trabalho de (CARBONARO *et al.*, 2018)



Fonte: (CARBONARO *et al.*, 2018)

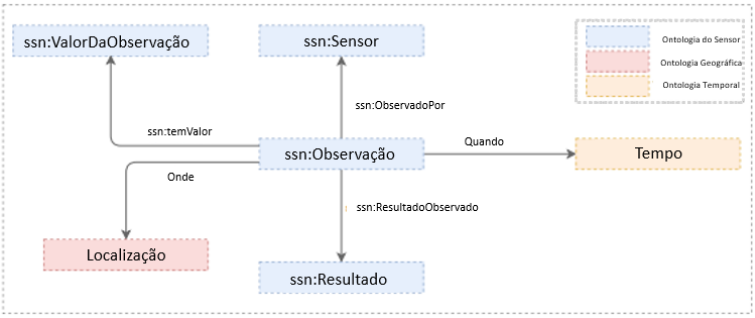
(2018) foi criada para dar suporte a um middleware que trata da interoperabilidade semântica de dados IoT e dados de saúde eletrônicos (EHR, do inglês *eletronic healthcare records*). A de (BROUWER *et al.*, 2018) foi desenvolvida visando o monitoramento de ciclistas em tempo real, buscando prover serviços personalizados para cada ciclista. Nela, é possível descrever zonas de treinamento, limites dos participantes, podendo receber comentários de especialistas, além do perfil fisiológico do ciclista, o qual pode indicar o ritmo cardíaco tanto em repouso quanto durante a atividade física.

Figura 18 – Ontologia SSN para a coleta de dados



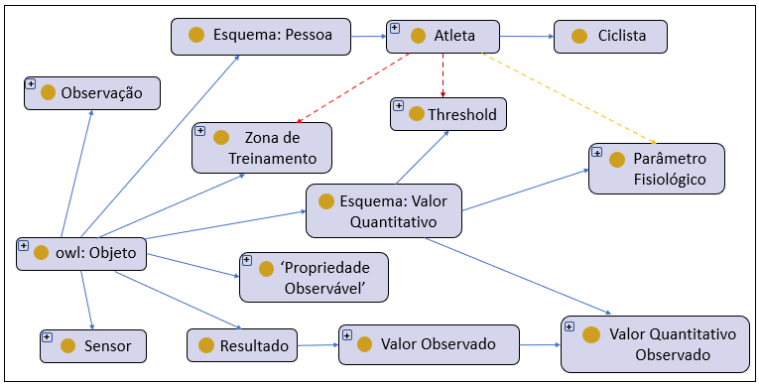
Fonte: (W3C, 2018)

Figura 19 – Ontologia definida no trabalho de (ALAMRI, 2018)



Fonte: Adaptado de (ALAMRI, 2018)

Figura 20 – Ontologia definida no trabalho de (BROUWER *et al.*, 2018)



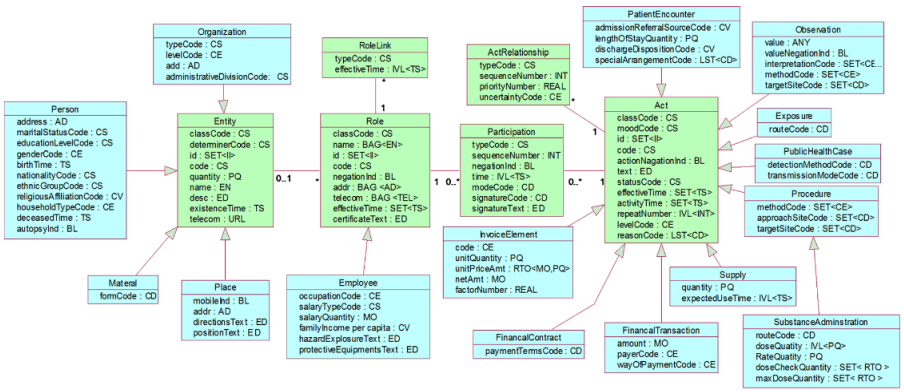
Fonte: Adaptado de (BROUWER *et al.*, 2018)

Além das ontologias citadas anteriormente, também foi encontrado o modelo orientado a objeto *Health Concept Data Model* (HCDM), proposto em Yang *et al.* (2022), e estende o modelo *Health Level* 7 (HL7) (BEELER, 1998; BLOBEL *et al.*, 2006), também orientado a objeto. O HL7 possui como *core* o Modelo de Informação de Referência (RIM, do inglês *Reference Information Model*) é focado na interoperabiliade sendo compatível com padrões de dados existentes e modelos de conhecimento de saúde e serve como base para integração de informações entre plataformas e sistemas.

O HCDM, por sua vez, evolui o HL7, que é considerado abstrato pelos autores,

tornando-o mais concreto, o que exigiu reorganização de classes, atributos e tipos dos dados, tendo as principais diferenças documentadas em Yang *et al.* (2022). O modelo HCDM é demonstrado na Figura 21 e possui seis (6) classes, quinze (15) subclasses. Essas classes e sub-classes podem estar relacionadas à pessoas, organizações, lugares que podem ter um papel e podem atuar, como encontrou um paciente, fez uma observação ou identificou um problema de saúde pública. Tanto o HL7 e o HCDM são modelos de dados genéricos para o contexto da saúde, o que requer uma adaptação para outros contextos, como o caso de detecção de quedas.

Figura 21 – Modelo de dados orientado a objeto proposto em (YANG *et al.*, 2022)



Fonte: (YANG *et al.*, 2022)

3.3 Discussão

Na literatura é possível encontrar diversos trabalhos que desenvolvam *frameworks* e modelos de dados diversos para a saúde, como os elencados na Seção 3.1 e 3.2. Entretanto, eles possuem diferenças entre si, sendo as principais os objetivos, o tipo de *framework* (e.g., de software, conceitual) ou de modelo de dados. Nesta Seção é feita uma discussão acerca deles apresentando as principais diferenças e limitações.

Quanto aos *frameworks*, foram elencados oito (8), dos quais quatro (4) são conceituais, que são um conjunto de metas gerais e fundamentos relacionados que definem objetivos gerais e conceitos para atingir os objetivos (NAZARI *et al.*, 2019). Os demais são de software, que permitem a construção de um projeto ou sistema (NAZARI *et al.*, 2019). Assim, apenas metade foi instanciado e pode ser utilizado na prática. Além disso, três (3) desses quatro (4) foram criados por multinacionais, focando na coleta de dados a partir de sensores integrados às plataformas de desenvolvimento delas, como Android, iOS e Withings. O último foi desenvolvido para facilitar a criação de sistemas de emergência móveis.

Em relação às tecnologias utilizadas para o desenvolvimento usando os *frameworks* de software, dois podem ser feitos em Java, podendo utilizar também Kotlin para desenvolver para um deles, o Google Fit. Para o da Apple, os desenvolvedores podem utilizar Swift ou Objective-C e para o Withings deve ser php. Quanto aos modelos de dados, dois deles usam modelos de dados orientado a objetos e um usa o modelo chave-valor. O último *framework* classificado como de software não descreve o modelo de dados utilizado, portanto, não pode ser categorizado. Os demais são *frameworks* conceituais e, portanto, não possuem um modelo próprio.

A funcionalidade comum capturada em cada *framework* tem relação com os objetivos para o qual cada um foi desenvolvido. Três deles focam na coleta de dados de diferentes dispositivos, sendo assim focados na interoperabilidade, funcionalidade também tratada pelo *framework* de (MACHADO *et al.*, 2014). Além disso, o JEMF trata da comunicação entre diferentes dispositivos, dado que deve haver a interação entre atores em diferentes lugares e acessando os sistemas desenvolvidos pelo *framework* a partir de dispositivos distintos. Adicionalmente, ele trata de armazenamento e integridade dos dados e disponibilidade do serviço.

Dois dos *frameworks* conceituais, (RAGHUPATHI; RAGHUPATHI, 2014; STANOJEVIć *et al.*, 2011) indicam os passos para alcançar o processamento de dados de ambientes de saúde, assim, o principal foco é o armazenamento e o processamento. Os outros dois tem objetivos mais distintos, sendo um focado em descoberta dinâmica e o outro é o único com foco em detecção e prevenção de quedas, porém também é um *framework* conceitual. Todos os critérios de comparação dos *frameworks* descritos anteriormente são sumarizados na Tabela 2.

Em relação aos modelos de dados, foram elencados onze (11), sendo três (3) deles relacionados aos *frameworks* descritos anteriormente, o Google Fit, o Apple HealthKit e o Withings. Os dois primeiros e os dois propostos por (YANG *et al.*, 2022; BEELER, 1998; BLOBEL *et al.*, 2006) são modelos de dados orientado a objetos, entretanto apenas o Google Fit e o Apple HealthKit foram pensados para o uso com dispositivos da IoT e são voltados à saúde. Os outros dois foram criados visando facilitar a interoperabilidade entre sistema de saúde genéricos.

Dentre os modelos de dados apresentados, apenas um (1) que é do tipo chave-valor, sendo ele o *Withings*, enquanto os seis (6) restantes são ontologias. Das ontologias, apenas duas foram criadas pensando em IoT e saúde, a IFO e a de (ALAMRI, 2018). A IFO é focada em atividades físicas, enquanto a segunda é focada em registro de dados eletrônicos, como dados

Tabela 2 – Tabela comparativa dos *frameworks*

framework	**Objetivo**	**Tipo**	**Funcionalidades Comuns**	**Tecnologia**	**Modelo de Dados**
Google Fit (GOOGLE, 2022)	Coletar dados de sensores	Software	Interoperabilidade	Java e Kotlin	Orientado a objeto
Apple HealtKit (APPLE, 2022a)	Coletar dados de sensores	Software	Interoperabilidade	Swift e Objective-C	Orientado a objeto
Withings (NOKIA, 2022)	Coletar dados de sensores	Software	Interoperabilidade	Php	Chave-valor
(RUGGERI; BRIANTE, 2017)	Descobrir dinâmica e automaticamente novos dispositivos usando redes sociais dos objetos	Conceitual	Descoberta dinâmica	-	-
(RAGHUPATHI; RAGHUPATHI, 2014)	Armazenar e processar grandes volumes de dados de saúde	Conceitual	Armazenamento e processamento de dados	-	-
(FANG *et al.*, 2016)	Armazenar e processar grandes volumes de dados de saúde	Conceitual	Armazenamento e processamento de dados	-	-
JEMF (MACHADO *et al.*, 2014)	Desenvolver sistemas de emergência móveis	Software	Comunicação, Armazenamento, Integridade, Interoperabilidade e Disponibilidade	Java	-
(HEMMATPOUR *et al.*, 2018)	Prevenção e Detecção de Quedas	Conceitual	Processamento de Dados	-	-

Fonte: O autor.

de pacientes registrados em um hospital. Quatro (4) são focadas em sistemas de saúde em geral, como sistemas de urgência e recomendação de sistemas médicos. A última é a SSN, uma ontologia focada em sensores, que pode ser utilizada para IoT e IoHT, embora não tenha sido o foco inicial delas. Todos os critérios de comparação dos modelos de dados descritos nesta Seção são sumarizados na Tabela 3 .

Diante desses dados, percebe-se a carência de *frameworks* de software e modelos de dados focados em detecção de quedas e suas causas. Assim, diante do quadro grave de quedas, como explanado no Capítulo 1, e diante das iniciativas que já existem para detectar quedas como Apple (2022b), Almeida *et al.* (2016) e (HSIEH *et al.*, 2014), é importante ter soluções de reuso visando facilitar o desenvolvimento de aplicações de detecção de quedas e associando com suas possíveis causas, dado que pode estar relacionado a um problema de saúde. Assim, é possível prover serviços mais rápidos, mais precisos e evita sequelas graves.

Tabela 3 – Tabela comparativa dos modelos de dados

Modelo	Objetivo	Tipo	IoHT
Google Fit (GOOGLE, 2022)	Armazenar dados de sensores.	Orientado a objeto	Sim
Apple HealtKit (APPLE, 2022a)	Armazenar dados de sensores.	Orientado a objeto	Sim
Withings (NOKIA, 2022)	Armazenar dados de sensores.	Chave-valor	Sim
(Xu *et al.*, 2014)	Ontologia para sistemas médicos baseados em serviços de sistemas de saúde de urgência.	Ontologia	Não
(KUMAR, 2015)	Recomendação de sistemas médico.	Ontologia	Não
(ALAMRI, 2018)	Apoiar a interoperabilidade semântica de dados IoT e dados de saúde eletrônico.	Ontologia	Sim
(BROUWER *et al.*, 2018)	Monitoramento de ciclistas em tempo real.	Ontologia	Não
IFO (REDA *et al.*, 2018) e (CARBONARO *et al.*, 2018)	Registrar dados fisiológicos, antropométricos e de atividade física.	Ontologia	Sim
SSN (W3C, 2018)	Lidar com dados de sensores e atuadores.	Ontologia	Não
(YANG *et al.*, 2022)	Interoperabilidade entre sistemas de saúde.	Orientado a objeto	Não
(BEELER, 1998) e (BLOBEL *et al.*, 2006)	Interoperabilidade entre sistemas de saúde.	Orientado a objeto	Não

Fonte: O autor.

3.4 Conclusão

Este capítulo apresentou os trabalhos relacionados ao que é proposto nesta tese. Eles foram classificados em duas categorias, sendo uma para *frameworks* e outra para os modelos de dados.

Foram encontrados oito (8) *frameworks*, sendo cinco (5) na literatura e três (3) desenvolvidos por grandes empresas. Desses oito *frameworks*, quatro são conceituais e quatro são de software. Apenas um dos oito visa a detecção e prevenção de quedas, sendo ele conceitual.

Dentre os modelos de dados, foram descritos onze (11) trabalhos, sendo seis (6) ontologias, quatro (4) orientados a objetos e um (1) chave-valor. Dos modelos orientado a objetos, dois focam no dados oriundos de sensores, assim como uma das ontologias e o modelo chave-valor, o que pode ser utilizado por aplicações IoHT. Além disso, duas das ontologias foram criadas com foco em IoHT e as demais são genéricas.

No próximo capítulo é apresentado a proposta desta tese, a qual consiste do Ágape, um *framework* para o desenvolvimento de aplicações IoHT com foco em detecção de quedas e suas possíveis causas a partir de dados oriundos de dispositivos vestíveis.

4 ÁGAPE

Neste capítulo é apresentado o *framework* Ágape[1], cujo objetivo é agregar dados de dispositivos distintos para identificar uma queda de idoso e suas possíveis causas. Para cumprir tal objetivo, o *framework* é dividido em quatro partes: i) *listener*; ii) *manager*; iii) agregador de dados; e iv) tomada de decisão.

O capítulo contém uma descrição do processo de desenvolvimento utilizado para o Ágape na Seção 4.1; na Seção 4.2 são detalhadas as experiências anteriores que contribuíram para o desenvolvimento do Ágape; visão geral do *framework* na Seção 4.3; na Seção 4.4 são descritos os módulos do Ágape e os algoritmos já embarcados nele; o modelo de dados especificado para o Ágape é descrito na Seção 4.5; enquanto que a Seção 4.6 a arquitetura é detalhada; a Seção 4.7 descreve elementos relacionados à implementação do *framework*; a Seção 4.8 contém as limitações do Ágape; e a Seção 4.9 conclui este capítulo.

4.1 Processo de Desenvolvimento do Ágape

O processo de desenvolvimento do Ágape foi uma adaptação dos processos descritos na Seção 2.5. O processo aqui descrito é composto de três fases com etapas e resultados bem definidos. A primeira fase é a definição do *framework*, seguida pelo desenvolvimento do *framework* e finaliza com a avaliação do *framework*, sendo essa avaliação descrita no Capítulo 5. A Figura 22 descreve o processo de desenvolvimento do Ágape.

Durante a definição do *framework* foram desenvolvidas duas versões com arquiteturas diferentes do WatchAlert, uma aplicação para detecção de quedas usando dados de *smartwatches*. Essa etapa é condizente com o processo baseado na experiência de aplicações proposto em Wilson e Wilson (1993). Essa experiência com o desenvolvimento foi documentada, gerando lições aprendidas que foram publicadas em Linhares *et al.* (2020). Tanto as aplicações como as lições são descritas na Seção 4.2 neste Capítulo.

Em seguida, foram elencados os principais elementos comuns a essas aplicações, além dos *hot spots*, que são os elementos que variam conforme cada aplicação. O processo de documentar a experiência e extrair os elementos comuns e diferentes foram executados em ciclos até obter-se um resultado que abrangesse as características mais relevantes a serem inseridas no Ágape. Após essa etapa, passou-se para a fase de desenvolvimento do *framework*, a qual inicia

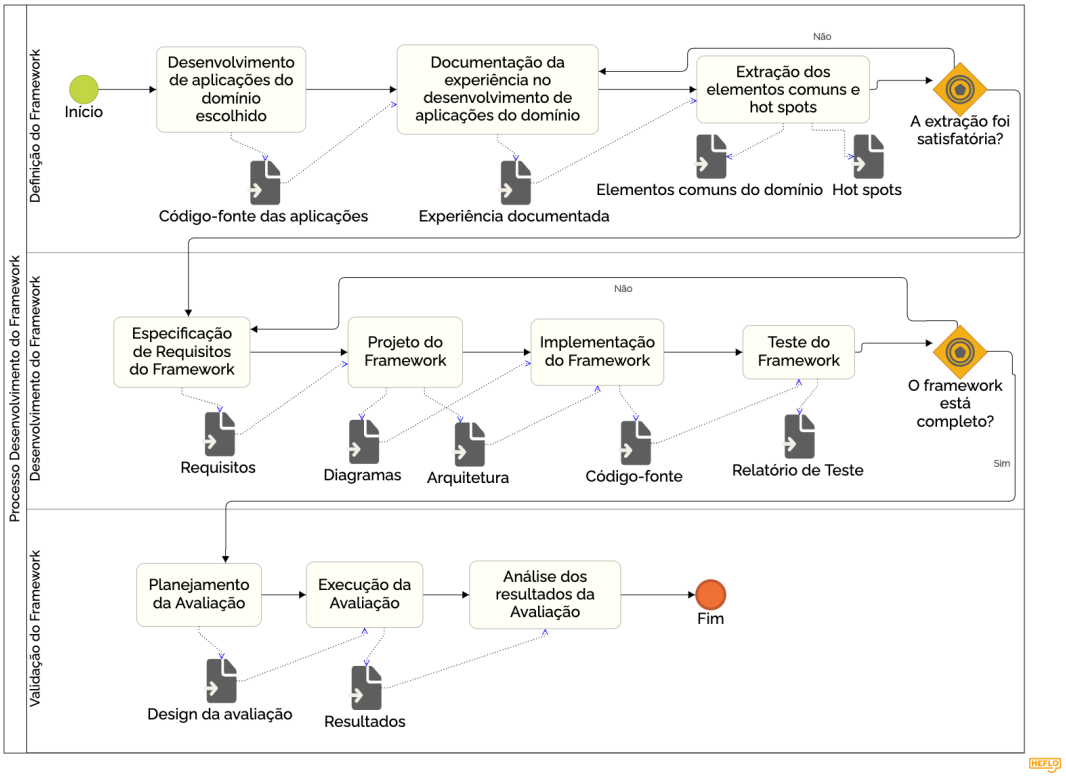
[1] O nome do *framework* é Ágape, pois é uma palavra grega que significa amor, o amor que se doa, que é incondicional. Esse amor leva a cuidar daqueles que se ama.

com a especificação de requisitos.

A etapa da especificação de requisitos consistiu da documentação dos elementos comuns e *hot spots* como requisitos, além de especificar outros requisitos que eram necessários para atender as necessidades, como a detecção das causas a partir das quedas. Em seguida, a partir dos requisitos, foi definida a arquitetura e gerados os diagramas de classe e sequência iniciais. Após essa etapa, iniciou-se a implementação do código-fonte do *framework*, que foi testado para assegurar que estava funcionando corretamente. Além disso, essa fase foi executada até que o *framework* atendesse a todos os requisitos, que eram revisitados a cada ciclo.

Ao final, foi feita a avaliação (última fase), a qual primeiro foi planejada, depois executada e por fim os resultados foram analisados, conforme detalhado no Capítulo 5.

Figura 22 – Processo utilizado para desenvolver o Ágape



Fonte: O autor.

4.2 Experiência em Aplicações IoHT

A experiência, essencial para o desenvolvimento do *framework* Ágape, foi adquirida com o desenvolvimento de uma solução inicial, a qual passou por uma evolução, incluindo melhorias no algoritmo e na arquitetura, o que contribuiu para o aumento da acurácia dos algoritmos. Esse processo de construção e evolução de uma solução, descrito nas Subseções

4.2.1 e 4.2.2, que teve início em 2016 e é um processo contínuo, também deu origem a um conjunto de lições aprendidas, as quais são descritas na Subseção 4.2.3, contendo o problema enfrentado, além da forma como foi obtida e a solução adotada em cada uma das situações.

4.2.1 WatchAlert: Primeira Versão

A primeira solução para a detecção de quedas desenvolvida e que contribuiu para a experiência adquirida foi o WatchAlert (ALMEIDA *et al.*, 2016), sendo ele uma aplicação para detecção de quedas a partir de dados de acelerômetro e giroscópio coletados por um *smartwatch* e com o apoio de um *smartphone*. Essa solução foi planejada ao observar a aplicação fAlert (PIVA *et al.*, 2014), que possuía o mesmo objetivo e não era IoT, contudo apresentava limitações para a sua utilização, pois contava com apenas um *smartphone* acoplado ao tórax do usuário, monitorando a aceleração do seu corpo, tornando seu o uso mais difícil, uma vez que não era natural.

Inicialmente, para a construção do WatchAlert, buscou-se resolver as limitações apresentadas no fAlert, e, para isso, a pesquisa teve como objetivo encontrar trabalhos que apresentassem algoritmos utilizando dispositivos acoplados no pulso dos usuários e que possuíssem uma alta acurácia. Um dos trabalhos encontrados e que norteou a criação do primeiro algoritmo do WatchAlert foi o Hsieh *et al.* (2014). Nele foi desenvolvido um dispositivo contendo dois sensores, acelerômetro e giroscópio, para ser acoplado nos dois pulsos dos usuários simultaneamente e um algoritmo de *thresholds* para a identificação das quedas. Embora atendesse aos critérios definidos, a solução ainda se apresentava como de difícil aplicação, uma vez que eram precisos dois dispositivos e não era acessível para aquisição.

A primeira versão do algoritmo do WatchAlert, como descrito anteriormente, foi uma adaptação do algoritmo descrito no trabalho de Hsieh *et al.* (2014), possuindo modificações na ordem das etapas e nos valores dos limites utilizados para se adequar a solução com apenas um dispositivo no pulso do usuário, que no caso do WatchAlert foi um *smartwatch*. O funcionamento do algoritmo não será detalhado nesta Seção, sendo a versão final descrita na seção 4.4.3.2. É relembrado aqui o que foi descrito no capítulo 2 que, para a detecção de uma queda, três etapas são necessárias. A primeira delas analisa se a pessoa passou por uma queda livre, sendo seguida pela etapa que verifica o impacto da pessoa com o solo, encerrando com um período de inatividade da pessoa, o que pode caracterizar que a pessoa requisita um atendimento de emergência. Caso haja uma situação de atividade intensa, isso pode definir uma atividade

cotidiana.

Nessa primeira versão, os dados do acelerômetro e giroscópio eram coletados em serviços executados no *smartwatch* e enviados para o *smartphone* realizar o processamento dos mesmos, buscando a identificação de uma queda. Ressalta-se que devido às limitações no hardware dos dispositivos, havia também a limitação dos algoritmos, sendo no *smartwatch* a restrição que impedia que o processamento ocorresse nele juntamente com a coleta de dados e no *smartphone* não havia a possibilidade de inserção de algoritmos de Aprendizagem de Máquina, possibilitando apenas os algoritmos de *thresholds*. Essas limitações foram essenciais para aprimorar os algoritmos nas versões posteriores do WatchAlert.

4.2.2 WatchAlert: Segunda Versão

Em análises mais detalhadas de cada etapa do algoritmo, verificou-se que os dados do giroscópio não estavam tendo grande influência para a detecção de quedas, além de verificarmos que uma mudança na última etapa da análise da inatividade do usuário poderia ser aprimorada. Anteriormente, para a análise da inatividade, era utilizado um somatório dos dados do acelerômetro após os instantes indicativos de queda livre e impacto com o solo e com a mudança passou-se a utilizar o desvio padrão desses dados, o que apresentou uma melhor acurácia, porém com mais acertos para situações cotidianas (especificidade) do que para as quedas (sensibilidade). As equações para os cálculos dessas métricas são apresentadas nas Equações 4.1, 4.2 e 4.3. Para entender as equações, é preciso entender as variáveis, que são demonstradas na Tabela 4 que contém uma matriz de confusão e na descrição a seguir:

- **Verdadeiro Positivo (VP):** Representa a quantidade de situações de queda identificadas corretamente;
- **Falso Negativo (FN):** Representa a quantidade de situações de queda identificadas como uma atividade diária;
- **Verdadeiro Negativo (VN):** Representa a quantidade de atividades diárias do usuário identificadas corretamente; e
- **Falso Positivo (FP):** Representa a quantidade de atividades diárias do usuário identificadas como queda.

$$Sensibilidade = \frac{VP}{VP+FN} \tag{4.1}$$

Tabela 4 – Matriz de confusão

		Identificados	
		Queda	Atividade Diária
Ações	Queda	VP	FN
	Atividade Diária	FP	VN

Fonte: O autor.

$$Especificidade = \frac{VN}{VN + FP} \tag{4.2}$$

$$Acurácia = \frac{VP + VN}{VP + FP + VN + FN} \tag{4.3}$$

As diferenças e os resultados entre os algoritmos são demonstradas na Tabela 5 e foram adaptadas do resultado encontrado e publicado em ARAÚJO *et al.* (2018).

Tabela 5 – Métricas dos algoritmos do WatchAlert

Algoritmo	Sensibilidade	Especificidade	Acurácia
WatchAlert com giroscópio (v1)	93.8%	95.5%	94.6%
WatchAlert sem giroscópio (v2)	92.9%	95.5%	94.2%

Fonte: Adaptado de (ARAÚJO *et al.*, 2018)

Embora houvesse o aprimoramento nos dados, buscou-se utilizar algoritmos de Aprendizagem de Máquina visando o aumento da acurácia, mas também com um aumento da sensibilidade, de forma que houvesse uma redução nas situações identificadas como atividades do dia a dia, mas que na verdade eram quedas, os falsos negativos. Na literatura indicava-se a detecção de quedas com os algoritmos *Support Vector Machine* (SVM), Árvore de Decisão e *Random Forest*, tendo o primeiro melhor resultado. Como esses algoritmos só podiam ser executados em uma Nuvem devido às restrições citadas anteriormente, modificou-se a arquitetura da aplicação para adicionar esse elemento de forma que não houvesse um aumento no tempo de detecção, uma vez que um atraso na resposta pode resultar em sequelas graves para um usuário que tenha caído.

Os resultados obtidos pelas análises também corroboravam os dados encontrados na literatura, possuindo o algoritmo SVM a maior acurácia, sensibilidade e especificidade. Contudo, foram feitas análises nas *features*, características extraídas a partir dos dados brutos, e os

resultados mostraram que haviam *features*, que, de modo análogo ao giroscópio, não agregavam ao resultado final e, portanto, foram substituídas. Esse processo resultou na modificação do algoritmo de Aprendizagem de Máquina com melhores resultados, modificando para o *Random Forest*. Até antes dessas análises os tempos eram baixos e similares, porém também foi verificado uma discrepância de 0.00102 segundos, contribuindo para a escolha do *Random Forest* como melhor algoritmo.

Ao final, o WatchAlert foi composto por uma arquitetura baseada na definida em Hiremath *et al.* (2014), a Wearable Internet of Things (WIoT), que é composta por três camadas, sendo a primeira a do dispositivos vestíveis, o que inclui *smartwatches*, e é a que os dados são coletados do usuário. A segunda desempenha um papel intermediário, estabelecendo a comunicação com a Internet, como um *gateway*, e a terceira é a Nuvem, em que podem executar os processamentos e o armazenamento de dados. Na camada dos dispositivos vestíveis, o *smartwatch* coleta os dados e os envia para o *smartphone* ao qual está conectado à Internet para que esse possa se comunicar com a Nuvem.

Na camada do *gateway*, o WatchAlert possui um algoritmo de *thresholds*, o qual é utilizado quando não há conectividade com a Nuvem, de forma que o usuário monitorado não fique sem o monitoramento adequado. Esse algoritmo é a última versão descrita nesta Seção, a qual utiliza apenas dados de acelerômetro e o desvio padrão na análise do comportamento da inatividade. Seja na interação com a Nuvem ou na detecção no próprio *smartphone*, ele envia uma mensagem para o serviço de comunicação do *smartwatch* visando a confirmação da queda com o usuário e a solicitação da ajuda de forma instantânea.

Na Nuvem, os dados recebidos pelo *gateway* são processados por modelos de AM que podem ser escolhido pelo desenvolvedor: i) *Random Forest*; e ii) *thresholds*. O *Random Forest* segue as definições encontradas na análise aqui descrita e o de *thresholds* é o mesmo inserido no *smartphone*. A diferença para esse último caso ocorre no local em que o algoritmo é executado e a capacidade de processamento de cada dispositivo, sendo na Nuvem mais robusta.

4.2.3 WatchAlert: Lições Aprendidas

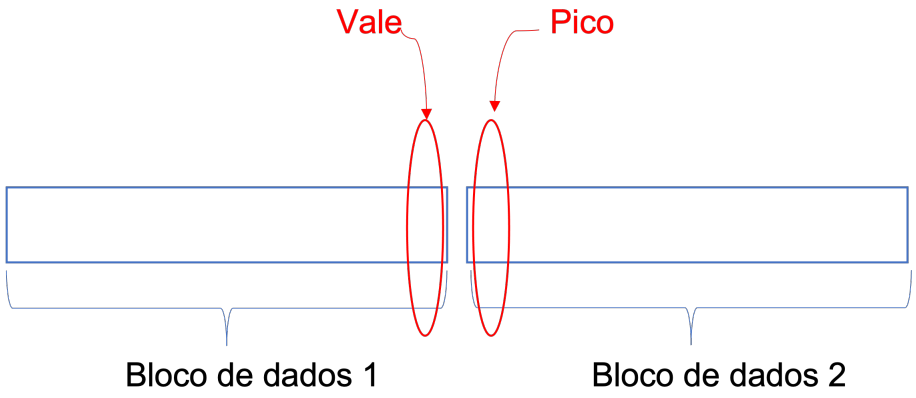
Todos esse processo de construção e evolução das aplicações foi documentado e publicado em (LINHARES *et al.*, 2020) juntamente com seis lições que foram aprendidas a partir do desenvolvimento da primeira versão do WatchAlert, iniciado em 2016. Essas lições estão relacionadas com aspectos arquiteturais, práticas de desenvolvimento de software, tolerância

à falhas, análises das *features* e seleção dos algoritmos de Aprendizagem de Máquina a serem utilizados em cada solução. Essas seis lições são descritas a seguir e cada descrição é composta de um problema, a situação que descreve a experiência vivida durante o desenvolvimento e a decisão tomada, representada pela lição.

LA01: Usar Programação baseada em Fluxo

- **Problema:** Inicialmente, considerou-se o envio de um bloco de dados para serem processados pelos algoritmos, contudo surgia problema quando a queda livre estava em um bloco diferente da identificação do impacto com o solo, como demonstrando na Figura 23, ou o estado do usuário em outro. Assim, como lidar com essa situação?
- **Experiência:** A situação foi evidenciada durante o o desenvolvimento do WatchAlert de forma que os dados eram coletados no *smartwatch* e o algoritmo executado estava no *smartphone*. Isso acabou gerando um grande número de situações consideradas como falsos negativos, pois os eventos característicos de uma queda estavam em blocos distintos. A solução encontrada foi a utilização de programação baseada em fluxo, que enviava constantemente os dados para o *smartphone*, de forma que pudesse unificar o final de um conjunto de dados com o novo conjunto de dados a fim de identificar uma queda.
- **Lição:** A solução encontrada foi a utilização de programação baseada em fluxo para reduzir esse problema de falsos negativos. Essa solução foi o melhor meio para lidar com a grande quantidade de dados coletados em tempo real e não bloquear a execução dos algoritmos em qualquer dispositivo, permitindo, assim, o monitoramento contínuo do usuário e o processamento dos dados.

Figura 23 – Problema da lição aprendida 1



Fonte: O autor.

LA02: Usar Arquitetura baseada em Serviços (SOA)

- **Problema:** Cada dispositivo executa pelo menos duas ações. No caso do *smartwatch*, o dispositivo coleta os dados, envia para para o *smartphone* e checa se o usuário está bem após a identificação de uma possível queda. No cenário do *smartphone*, ele recebe os dados do *smartwatch*, processa-os ou envia para a Nuvem e retorna o resultado de volta ao *smartwatch*, além de enviar uma mensagem para a família ou cuidadores. Como manter todas essas ações sem interromper qualquer uma das outras ações?
- **Experiência:** Nas primeiras versões do WatchAlert, tentou-se criar uma aplicação para enviar todos os dados recebidos, receber a resposta e checar o status da saúde do usuário, o que foi feito de uma forma inviável, atrapalhando o correto funcionamento dessas funcionalidades. Isso impactava em outros processos, sendo necessário um esforço para a aplicação do *smartwatch* funcionar com a utilização de serviços. Dessa forma, cada funcionalidade executada no *smartwatch* foi desenvolvida como um serviço executando em paralelo.
- **Lição:** A lição aprendida é que para desenvolver aplicações de IoHT é importante a utilização de Arquitetura Baseada em Serviço (SOA, do inglês *Service-Oriented Architecture*). Para o melhor uso dessa arquitetura, é importante que cada funcionalidade seja inserida em um serviço, que pode, por exemplo, executar a coleta de dados e a checagem do status da saúde do usuário sem que as funcionalidades interfiram ou impactem uma na outra.

LA03: Dividir o código em diferentes dispositivos

- **Problema:** Muitos dispositivos, especialmente os dispositivos vestíveis, possuem recursos limitados e muitas vezes não conseguem executar funcionalidades de forma paralela, como a coleta de dados e o processamento deles. Então, surge a dúvida: como resolver esse problema sem impactar no funcionamento da aplicação?
- **Experiência:** Baseada na experiência descrita nesta Seção, tentou-se, inicialmente, utilizar o *smartwatch* para executar dupla função de coletar e processar os dados. Contudo, a aplicação sempre deixava de funcionar e ocasionava o reinício do *smartwatch* devido às limitações do hardware. Assim, era inviável utilizar o *smartwatch* dessa maneira, o que requisitou que as funcionalidades fossem distribuídas para outros dispositivos, como o *smartphone* e a Nuvem.
- **Lição:** A sugestão para os desenvolvedores é analisar os recursos de cada dispositivo e dividir os códigos entre cada um deles baseado nessas informações. Isso permite melhorar a detecção de quedas e continuar monitorando o usuário, por exemplo. Além disso,

essa divisão pode incluir a inserção do processamento em uma Nuvem, de forma que o desempenho seja rápido e também não impacte na performence da aplicação.

LA04: Usar métodos de detecção offline integrados com algoritmos de Aprendizagem de Máquina

- **Problema:** Aplicações de saúde são críticas e o serviço fornecido deve estar sempre disponível. Ao utilizar algoritmos de Aprendizagem de Máquina (ML, do inglês *Machine Learning*), que tem uma acurácia maior, pode haver um comprometimento no serviço quando não houver conectividade com a Internet, uma vez que eles costumam executar na Nuvem, pois são muito robustos para executarem em dispositivos vestíveis. Então, como assegurar alta disponibilidade do serviço e modelos de detecção auto-adaptativos?
- **Experiência:** Foi utilizada uma solução de ML baseada em Nuvem, mas essa estratégia foi afetada por problemas de rede. Ao considerar o contexto crítico desse tipo de aplicação, foi essencial também utilizar métodos *offlines*, como uso de algoritmo de *thresholds* no *smartphone*, para assegurar a disponibilidade do serviço mesmo com problemas de rede. Dessa forma, foi possível garantir a disponibilidade do serviço e que também pode evoluir a partir do uso de novos dados.
- **Lição:** Se um desenvolvedor desejar utilizar algoritmos de ML para detecção de quedas, ele deve estar em uma Nuvem, mas é importante ter um método alternativo que não dependa de acesso à rede. Assim, sugere-se inserir pelo menos um algoritmo de *threshold* no *smartphone* para essas situações. Portanto, é possível assegurar a alta disponibilidade do serviço e um modelo de detecção auto-adaptável.

LA05: Análise e Seleção das melhores features para algoritmos de Aprendizagem de Máquina

- **Problema:** Quando utilizar ML para a detecção de quedas, é possível utilizar diversas *features* (características) para contribuir para a melhoria dos resultados. Além disso, um grande número de *features* pode, consequentemente, fazer o processamento do algoritmo ser mais lento. Assim, como selecionar as melhores *features* para algoritmos de ML?
- **Experiência:** Durante as análises das métricas sensibilidade, especificidade, acurácia, entre outras, percebeu-se que haviam *features* que contribuíam mais do que outras para os resultados do algoritmos de ML. Assim, essas *features* foram avaliadas com uma algoritmos de análise de correlação e de seleção de atributos. Consequentemente, ao utilizar as *features* selecionadas, os resultados obtidos foram melhores e o tempo de

processamento foi menor.

– **Lição:** Para os melhores resultados com algoritmos de ML, as *features* usadas podem ser selecionadas utilizando técnicas específicas para análises considerando os cenários de cada aplicação. Recomenda-se utilizar os algoritmos de análise de correlação e análise de atributos, como o *Information Gain* e o *SVM Attribute Evaluation*. No cenário de detecção de quedas, sugere-se o uso dos valores máximo, mínimo, raiz quadrada média e curtose quando utilizar apenas dados de acelerômetro de um *smartwatch*.

LA06: Escolher o algoritmo de ML mais adequado

– **Problema:** Em muitos trabalhos encontrados na literatura foram encontrados o uso de algoritmos de ML para detecção de quedas, mas sem a escolha deles ser justificada. Porém, alguns fatores impactam os resultados dos algoritmos, como o tipo de *feature*, a quantidade de dados e como os dados são organizados. Assim, qual algoritmo escolher?

– **Experiência:** A escolha dos algoritmos de ML baseada apenas na literatura se demonstrou ineficiente, pois foram avaliados diferentes algoritmos com diferentes *features*. Além disso, os resultados mostraram que com os dados coletados, o melhor algoritmo era o *Random Forest*, o que foi diferente do encontrado na literatura, que sugeria o uso do *Support Vector Machine*.

– **Lição:** A escolha do algoritmo de ML para detecção de quedas deve ser baseada nos dados disponíveis. Para os cenários de detecção de quedas, sugere-se avaliar, pelo menos, a variação dos algoritmos de SVM e dos baseados em árvores quando utilizarem apenas dados de acelerômetro.

4.3 Visão Geral do Ágape

O Ágape é um *framework* que foi planejado para auxiliar no desenvolvimento de aplicações IoHT com foco em detecção de quedas e suas possíveis causas. Para alcançar esse objetivo, duas aplicações desse domínio foram analisadas, o WatchAlert nas suas duas versões, além do fAlert. Essa análise juntamente com as lições aprendidas resultou nos requisitos descritos a seguir. Também foi feito um mapeamento dos requisitos com as lições aprendidas com a experiência adquirida no domínio e foi sumarizado na Tabela 6:

RF 1. O Ágape deve ser capaz de receber os dados dos dispositivos em forma de fluxo;

RF 2. O Ágape deve funcionar como um serviço, em que os aplicativos enviam os dados e recebem o serviço;

RF 3. O Ágape deve tratar dados de diferentes dispositivos e sensores;

RF 4. O Ágape deve permitir a inserção de novos dispositivos e sensores para serem reconhecidos por ele para monitorar o usuário para detectar quedas ou das possíveis causas;

RF 5. O Ágape deve permitir alterar ou remover os dispositivos e sensores reconhecidos por ele;

RF 6. O Ágape deve entender que cada par <dispositivo, sensor> observa uma propriedade;

RF 7. O Ágape deve possuir algoritmos gerados durante a experiência adquirida no desenvolvimento de outras aplicações e também da literatura para detecção de quedas ou das possíveis causas;

RF 8. O Ágape deve permitir a adição de novos algoritmos para detecção de quedas ou das possíveis causas;

RF 9. O Ágape deve permitir a seleção dos algoritmos a serem executados para detecção de quedas ou das possíveis causas;

RF 10. O Ágape deve tratar os dados usando análises temporais;

RF 11. O Ágape deve agregar os dados dos dispositivos que analisam propriedades iguais;

RF 12. O Ágape deve identificar uma queda e suas possíveis causas;

RF 13. O Ágape deve armazenar os dados em um banco de dados temporal.

Tabela 6 – Mapeamento dos Requisitos e Lições Aprendidas

Requisitos	**Lições Aprendidas**	**Justificativa**
RF1	LA01	Considerando que os dados devem ser processados de forma contínua, o Ágape deve ser capaz de receber os dados de tal forma evitando problemas na identificação de uma queda.
RF2	LA02	O Ágape deve ser instanciado de forma a atuar como um serviço, que recebe os dados dos dispositivos e retorna a detecção de quedas e suas causas, sendo o funcionamento transparente para os aplicativos.
RF7	LA05	O desenvolvedor pode gerar novos algoritmos, com sensibilidade e/ou especificidade maior para dispositivos e sensores, uma vez que o contexto de cada aplicação pode variar, como os dispositivos de monitoramento. Assim, o desenvolvedor deve ser capaz de adicionar esses novos algoritmos.
RF8 e RF9	LA06	O desenvolvedor deve ser capaz de escolher qual algoritmo ele quer executar, uma vez que ele sabe os melhores algoritmos para cada dispositivo e sensor.

Fonte: O autor.

Quanto à lição LA03, o Ágape já é fruto dela, uma vez que ele foi projetado para executar de forma independente do dispositivo de monitoramento do usuário. Em relação à lição LA04, é preciso que os desenvolvedores implementem alternativas a serem utilizadas no dispositivo para situações em que o Ágape não esteja disponível. Além disso, são inseridos

no Ágape dois algoritmos de *thresholds*, os quais requerem baixo poder de processamento, compatível com o de *smartphones*, que podem ser replicados para os dispositivos.

Além disso, destaca-se que o Ágape possui as características de comunicação, processamento de dados, armazenamento, distribuídas em cinco (5) módulos: i) *listener*; ii) *manager*; iii) processamento de dados; iv) agregação de dados; e v) tomada de decisão. O *listener* tem o papel de estabelecer a comunicação com os dispositivos, enquanto nos demais há o armazenamento dos dados a partir dos dados utilizados ou gerado em cada um, além de possuírem os processamentos desses mesmos dados.

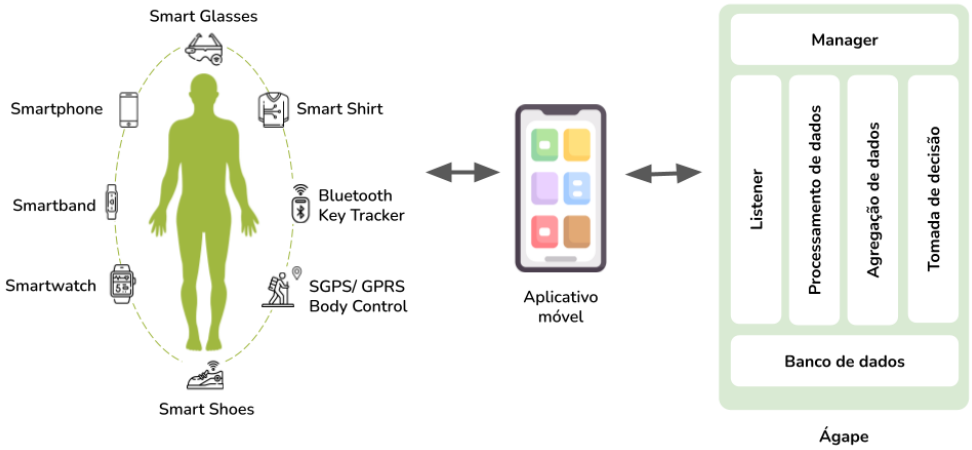
A visão geral do Ágape é demonstrada na Figura 24. Nela, é possível ver mais à esquerda a indicação do usuário sendo monitorado por dispositivos vestíveis, que possuem a comunicação com um *gateway*, o qual pode ser um *smartphone*. Mais a direita está o Ágape, com os cinco módulos que o compõe. O primeiro módulo é o *listener* e ele atua como interface de comunicação com os outros dispositivos, como o *smartphone*. Exceto pelo *manager*, que é um módulo transversal, todos os demais módulos estão alinhados em camadas abaixo do *listener*. O *manager* tem o conhecimento de todos os módulos para gerenciar a execução das atividades, bem como os seus status. Todos os módulos com exceção do *listener* tem acesso ao banco de dados temporal utilizado para armazenar os dados utilizado no Ágape.

Quando um dado é recebido pelo *listener*, ele repassa ao *manager* a fim de verificar se os dados tão consistentes e iniciar a execução dos outros módulos. Ele faz uma análise inicial dos dados e executa os algoritmos definidos para executar para aquele dispositivo e sensor. Em seguida, os episódios iniciais são gerados para serem analisados pelo módulo de agregação de dados. Após ser executado corretamente, o *manager* executa o módulo de tomada de decisão a fim de detectar se houve uma queda e as possíveis causas. Mais detalhes sobre os módulos são descritos na Seção 4.4.

Além disso, o Ágape pode ser reutilizado mais facilmente, pois atende a características dos *frameworks*, como possuir os *hot spots*, além de utilizar a propriedade herança da orientação a objeto para instanciar os elementos a serem modificados ou criados. Os principais *hot spots* do Ágape estão relacionados aos algoritmos utilizados e a forma como lidar com a agregação dos dados. Mais detalhes são descritos nas Subseções 4.4.3 e 4.4.4.

Ademais, destaca-se que o principal foco do Ágape é no processamento e agregação dos dados coletados pelos dispositivos vestíveis o que permite identificar uma queda e as possíveis causas. Ele foi planejado para ser independente de tecnologia a fim de evitar restrições

Figura 24 – Visão Geral do Ágape



Fonte: O autor.

de plataformas que possam utilizar o Ágape. Assim, ele é compatível com plataformas diferentes de aplicações, como Android e iOS. Portanto, os desenvolvedores conseguem focar no envio dos dados, seguindo os formatos dos dados compreendidos pelo Ágape, sem preocupar-se com a tecnologia em que a aplicação irá ser desenvolvida.

4.4 Módulos do Ágape

Nesta seção são descritos os módulos do Ágape, iniciando com a descrição do módulo que serve de interface com os dispositivos de monitoramento do usuário, o *Listener*, na Subseção 4.4.1; seguido pela descrição do módulo *Manager* na Subseção 4.4.2; na sequência, é detalhado o módulo *Processamento de Dados* na Subseção 4.4.4; enquanto na Subseção 4.4.5 é explicitado como o Ágape toma uma decisão para encontrar a possível causa de uma queda.

4.4.1 Listener

Considerando que o Angel executa em um servidor, que pode ser na Cloud, e deve processar os dados coletados por dispositivos vestíveis, é importante ter uma porta de acesso para a comunicação com os dispositivos vestíveis. Diante disso, foi definido este módulo, o *Listener*, cujo objetivo é justamente estabelecer a comunicação com os dispositivos, facilitando o recebimento dos dados coletados por dispositivos que monitoram o usuário a fim de permitir que tais dados possam ser analisados a fim de detectar uma queda e as possíveis causas da queda.

A Figura 24 contém a representação do módulo *Listener*, bem como as interações

dele com os dispositivos e os outros módulos. Cabe destacar que toda a interação inicia com o envio de dados pelos dispositivos, não cabendo a este módulo a solicitação de envio de nenhum dado para que seja processado. Após o recebimento, este módulo repassa os dados para o *Manager*, o qual é descrito na Seção 4.4.2.

Destaca-se que para lidar com o recebimento dos dados de variados e diferentes dispositivos dos usuários, o *Listener* foi planejado para trabalhar com filas de maneira que um conjunto de dados possa tentar ser processado diversas vezes, caso haja algum problema durante a execução. Além disso, o *Listener* permite a programação em fluxos, cabendo a aplicação enviar os dados em fluxos, como especificado na lição aprendida 1.

Ademais, há um *hot spot* nesse módulo relacionado ao tratamento dos dados recebidos. Cada aplicação deve enviar os dados em um formato de tupla, que é explicada na Seção 4.4.2. Contudo, a forma de enviar a tupla ou uma lista de tuplas quando forem vários dados pode variar de acordo com o padrão utilizado pelo desenvolvedor para enviá-los, o que pode ser ajustado para cada aplicação. Isso requer que os dados possam ser reorganizados para que as tuplas sejam identificadas pelo *Manager*.

4.4.2 Manager

Após o recebimento dos dados pelo *Listener*, eles são enviados para o módulo *Manager*. Neste módulo, os dados passam por uma etapa de pré-processamento e organização para serem armazenados em um banco de dados de séries temporais com o intuito de facilitar a análise deles nos outros módulos, além de manter o histórico de cada informação identificada.

Antes de realizar qualquer processamento, o *Manager* gerencia os dispositivos e sensores que são utilizados pela aplicação e associa os algoritmos para cada uma das combinações (dispositivo, sensor). Isso ocorre porque cada sensor monitora uma característica do usuário, que podem ser a aceleração do corpo visando a identificação de uma queda até um dado relativo à saúde do usuário, como os batimentos cardíacos ou a glicemia e possuem formas diferentes de serem tratados e interpretados.

O último ponto a ser registrado é a prioridade de cada dispositivo e sensor, a ser utilizado posteriormente no módulo *Agregação de Dados*. Essa prioridade é definida pelos desenvolvedores considerando a precisão dos algoritmos utilizados e o quanto eles sabem sobre a relevância de cada algoritmo para a solução. Os valores da prioridades começam em 1 e não possuem limite. Eles podem ser definidos como: dispositivo 1 tem a prioridade 3 e o dispositivo

2 tem a prioridade 1. Assim, o dispositivo 1 é 3 vezes mais preciso do que o dispositivo 2.

Dito isto, inicia-se o pré-processamento dos dados, o qual consiste na análise deles, de modo a checar se os dados possuem as informações necessárias e se obedecem ao padrão definido, o qual garante que os dados possam ser processados. Além disso, a padronização dos dados contribui também para a comunicação dos dispositivos de plataformas diferentes com o Ágape, uma vez que tais dispositivos precisam apenas seguir o padrão definido. Esse padrão segue a definição em forma de tupla demonstrada a seguir:

$$< timestamp, dados, sensor, dispositivo, usuário >$$

O primeiro elemento do padrão dos dados representa o instante em que os dados foram coletados, o que permite uma construção temporal, permitindo a identificação do momento inicial daqueles dados coletados até o instante final. Além disso, permite a comparação entre os dados de sensores com frequências distintas. No caso, o *timestamp* indica o instante ou o intervalo em que uma queda ou uma situação anômala ocorreu. Quanto à representação, os valores de tempos são representados por números Reais, os quais podem ser convertidos para dia, mês, ano, hora, minuto, segundo e milisegundo.

Na sequência, são inseridos os dados coletados pelos sensores, cuja quantidade pode variar devido às diferenças existentes em cada tipo de sensor. Isso se torna mais claro ao observar sensores que utilizam quantidade diferentes de valores para representar os comportamentos do usuário, como um sensor que envia apenas um único dado e outro que envia três valores, por exemplo. Esses dados costumam pertencer aos números Reais, devido à precisão desse tipo de dado, e, portanto, atribuiu-se o uso de números Reais ao Ágape.

Seguindo a análise dos elementos presentes nos dados enviados, o próximo elemento corresponde ao sensor que realizou a coleta dos dados, de forma que o Ágape inicia a identificação de possíveis algoritmos a serem utilizados para processar os dados recebidos. Esse campo deve ser preenchido com um valor pertencente aos números Naturais e que seja único para cada dispositivo. A escolha por essa representação ocorreu devido à prévia utilização por alguns dispositivos, o que eliminou a necessidade de um esforço a mais para realizar uma conversão desse valor para outra forma de identificar os dados.

A escolha dos algoritmos para processar os dados utiliza como primeiro elemento de definição o sensor, como visto anteriormente, mas não se limita a esse dado. Como cada sensor pode estar relacionado a um dispositivo diferente, o qual é utilizado em locais diferentes pelo

usuário, como um relógio ou um celular no bolso. Isso impacta nas análises dos dados, e, portanto, o dispositivo também é considerado e é identificado na tupla de dados que o Ágape consegue interpretar. Essa informação é descrita como um texto (*string*), a qual o desenvolvedor deve atentar-se ao desenvolver a aplicação para alinhar com o Ágape. Além disso, os desenvolvedores podem adicionar novos dispositivos ou alterar os existentes.

Por fim, o último elemento a ser inserido nos dados enviados pelos dispositivos corresponde ao usuário que aqueles dados pertencem. Essa informação possui dupla vantagem, sendo que a primeira é a privacidade do usuário, de modo que os dados gerados pelo monitoramento de um usuário não seja acessível por outro. A segunda vantagem consiste na identificação a quem pertence para que os dados de outro usuário não impactem nas análises o que poderia gerar resultados incorretos tanto para falsos positivos quanto para falsos negativos.

Após realizada a análise para cada dado enviado ao Ágape, se os dados obedecerem ao padrão descrito, eles são adicionados a um *buffer* único para cada par de dispositivo e sensor (dispositivo, sensor). Feito isso, eles são enviados para os algoritmos, que pertencem ao módulo de *Processamento de Dados*, que é detalhado na Seção 4.4.3. Os resultados dos processamentos são retornados e armazenados em um banco de dados temporal, associando a cada resultado, o tempo inicial e final, caso haja, do conjunto de dados que foi processado, o usuário, o dispositivo, o sensor e a propriedade observada pelo sensor.

Em seguida, este módulo executa o módulo de *Agregação de Dados* seguido pela execução do de *Tomada de Decisão*. Ao final da tomada de decisão, os dados retornados pelo módulo são encaminhados para o *Listener*, de modo que ele possa retornar ao dispositivo que enviou os dados, o resultado para que o mesmo seja analisado e em caso de uma situação de alerta, tomar as medidas necessárias.

4.4.3 Processamento de Dados

4.4.3.1 Descrição do Módulo

Este módulo é executado quando os dados recebidos e analisados pelo *Manager* estão de acordo com o padrão definido, garantindo que os dados poderão ser processados pelos algoritmos, pois eles contêm as informações necessárias. Conforme visto na Seção 4.4.2, é criado um *buffer* com todos os dados enviados e separados por dispositivos e são repassados para este módulo.

Após o recebimento dos dados, eles são armazenados em um *buffer* local do módulo, para que aquele conjunto de dados recebido seja tratado e processado pelos algoritmos. Esse tratamento nos dados consiste na geração de novas *features*, as quais são relevantes para os algoritmos de AM utilizados, pois pode variar para cada algoritmo e conjunto de dados e devem ser definidas pelos desenvolvedores, como consta nas lições aprendidas. Essas *features* representam novos valores que podem ser extraídos a partir dos dados brutos. Exemplos de *features* podem ser o desvio padrão, variância, entre outros.

Esse ponto de geração de novas *features* é o primeiro *hotspot* do Ágape, pois os desenvolvedores podem manter o que já foi inserido, alterar conforme as suas necessidades ou adicionar novas *features*, quando for o caso. A partir de então, há o segundo *hotspot*, que corresponde aos algoritmos, os quais também podem ser mantidos, atualizados ou novos podem ser adicionados.

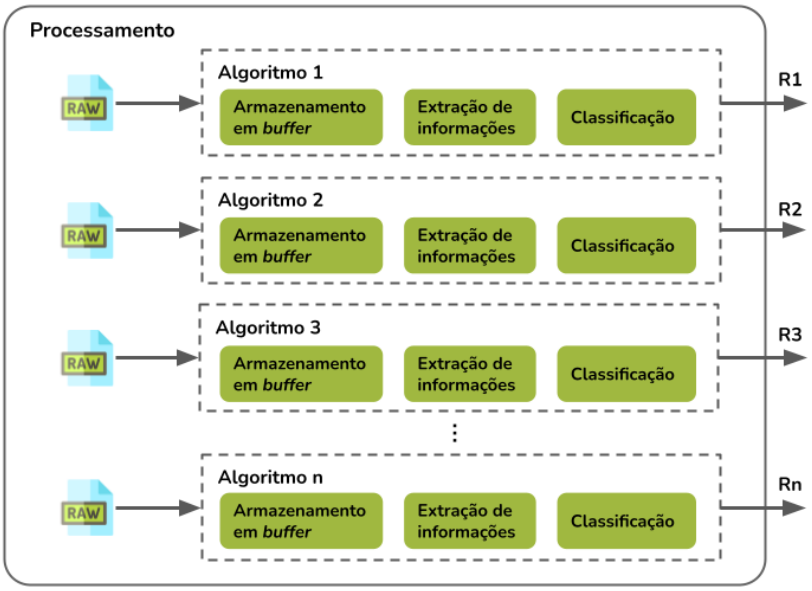
Nesse momento, os dados, que já foram analisados e que tiveram *features* geradas, são submetidos aos algoritmos. Os algoritmos, por sua vez, são executados buscando a identificação de uma queda ou de comportamentos que não correspondam aos padrões de normalidade dos dados. Ao final, cada resultado é enviado enviado para o *Manager*, o qual, como descrito na Seção 4.4.2, armazena os resultados e executa os demais módulos para que uma decisão possa ser tomada.

Quanto aos algoritmos, eles foram selecionados e definidos tendo como base a experiência do autor, correspondendo ao desenvolvimento de uma solução para detecção de quedas utilizando um relógio inteligente, o WatchAlert (ALMEIDA *et al.*, 2016) e também o conhecimento adquirido da literatura.

4.4.3.2 Algoritmos presentes no Ágape

Nesta Subseção são descritos os algoritmos inseridos no framework Ágape. Foram adicionados algoritmos de *thresholds*, como a última versão do WatchAlert e o de (HSIEH *et al.*, 2014). Além disso, foram adicionados algoritmos de Aprendizagem de Máquina, que podem ser evoluídos mais facilmente, dado que pode ser gerados novos modelos a partir de novos dados. Os algoritmos utilizados também foram utilizados no WatchAlert (e.g., SVM e *Random Forest*), bem como encontrados na literatura como o K-nearest neighbors (KNN) proposto por (HUSSAIN *et al.*, 2019).

Figura 25 – Funcionamento do Módulo de Processamento



Fonte: O autor.

4.4.3.2.1 Algoritmo do WatchAlert

Aqui é descrita a última versão do algoritmo de *threshold* do WatchAlert conforme a evolução descrita na seção 4.2 e fluxo visualizado em Figura 26. O algoritmo utiliza apenas dados de acelerômetro presente em um *smartwatch* e possui quatro etapas para identificar o momento de queda livre, impacto com o solo e inatividade. Para que as etapas possam ocorrer normalmente, é preciso compreender como são os dados gerados por um acelerômetro. Na versão do algoritmo, foram utilizados os dados de um acelerômetro de três eixos, correspondendo aos eixos X, Y e Z. Contudo, os dados isolados não possuem um significado completo, uma vez que eles podem ter sido coletados em qualquer instante. Assim, associado a esses valores, tem-se o *timestamp*, que representa o momento em que os dados foram coletados, além do valor que caracteriza o sensor que coletou a informação, no caso o acelerômetro. Ao final, a tupla enviada corresponde ao padrão a seguir:

$$< timestamp, eixoX, eixoY, eixoZ, sensor >$$

Embora os dados sejam distintos para os três eixos, é muito complexo analisá-los de maneira separada, sendo necessário realizar um cálculo para convertê-los em uma única variável. Tal cálculo segue a fórmula 4.4, semelhante a apresentada no Capítulo 2, a qual calcula a raiz quadrada da soma dos quadrados dos valores para cada eixo. Essa equação é utilizada em dois momentos: i) identificação do vale; e ii) identificação do pico.

$$VA(t_i) = \sqrt{{A_x}^2(t_i) + {A_y}^2(t_i) + {A_z}^2(t_i)} \tag{4.4}$$

O instante de identificação do vale é o instante que guia os outros cálculos. Assim, para a identificação do pico, ele tem que ser a partir do instante da queda até no máximo quinhentos (500) milissegundos depois. Fora desse comportamento não há mais uma indícios de queda. Os dois passos seguintes calculam o desvio padrão em instantes diferentes e os comparam com *thresholds* e o cálculo é apresentado na Equação 4.5.

Para entender a equação e as duas últimas etapas, é preciso explica-las. A variável A_i corresponde ao valor da aceleração obtido com a Equação 4.4 para cada instante i. No caso do passo 3 do algoritmo, ele corresponde a análise dos quinhentos (500) milissegundos após a identificação do vale, enquanto no passo 4 verificar os 1500 milissegundos após o instante i. Esse valor A_i é subtraído da média aritmética dos valores da aceleração e essa diferença é elevada ao quadrado. É feito o somatório de todos os valores resultantes dessa diferença e o final é dividio pelo número de amostras utilizadas para o cálculo.

$$DP = \sqrt{\frac{\sum_{i=1}^{n}(A_i - M_a)^2}{n}} \tag{4.5}$$

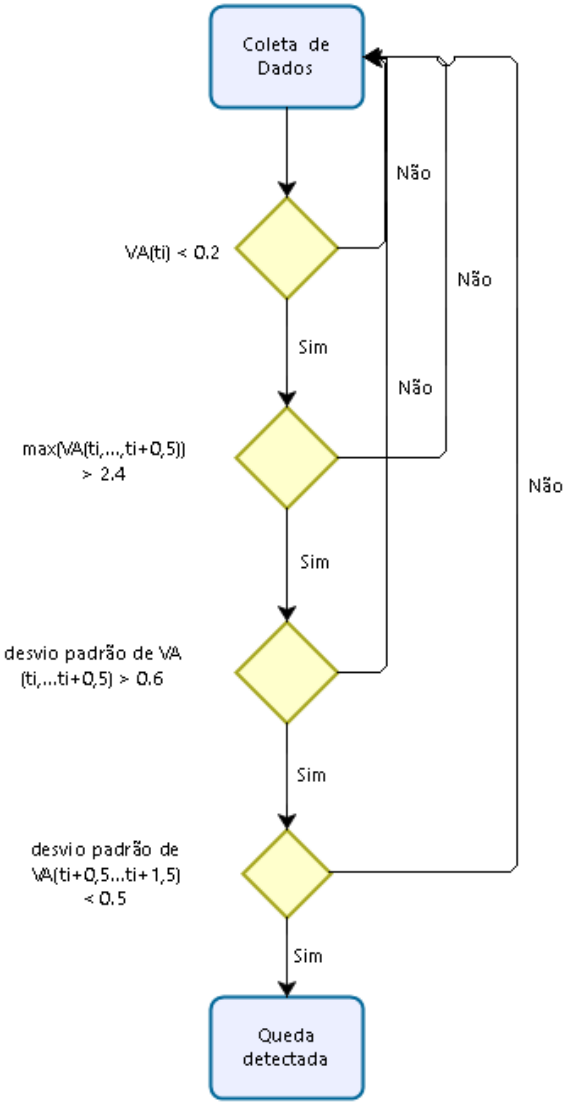
4.4.3.2.2 Algoritmo de (HSIEH *et al.*, 2014)

A Figura 27 contém o fluxo executado pelo algoritmo proposto por Hsieh *et al.* (2014) a fim de detectar uma queda. Antes de iniciar o fluxo, o usuário deve estar sendo monitorado por dois dispositivos embarcados com os sensores acelerômetro e giroscópio e devem alocados nos pulsos.

Em seguida, quando o dispositivo inicia o processo de coleta de dados, os dados de giroscópio são os primeiros a serem analisados. Ele é um sensor de três eixos como o acelerômetro, porém apenas dois, o Y e o Z, são utilizados para a conversão em um valor apenas a fim de fazer as análises para identificação das quedas. O cálculo é demonstrado na Equação 4.6 e é similar ao cálculo da aceleração.

Considerando o valor do giroscópio, ele é medido em graus por segundos ($^\circ/s$) e caso algum valor seja superior a $350^\circ/s$, o algoritmo interpreta como tendo o primeiro critério aceito para a identificação de uma queda. Na sequência, é feita uma análise do vale a partir dos

Figura 26 – Fluxo de execução do algoritmo do WatchAlert



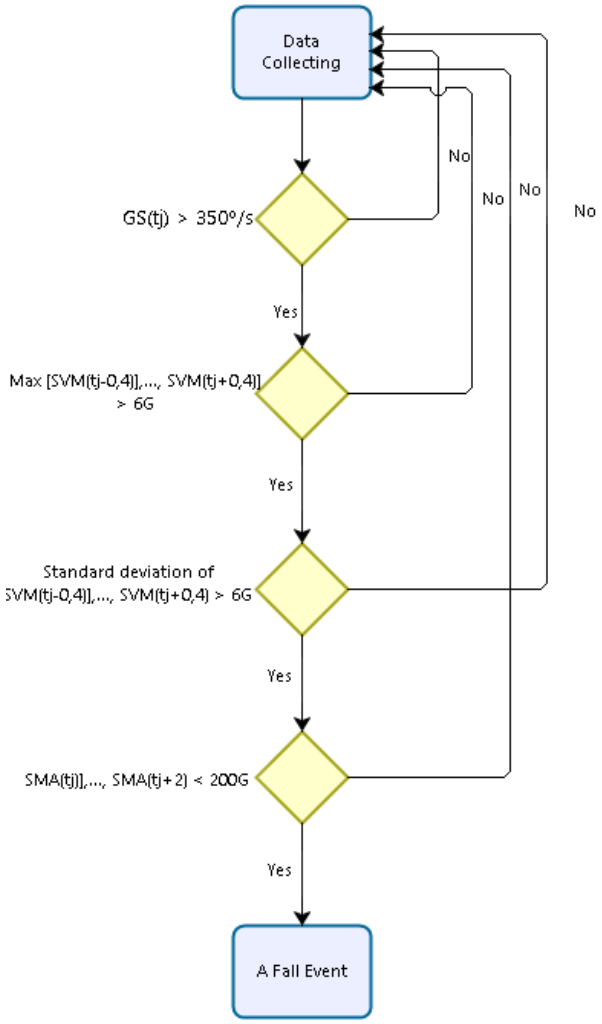
Fonte: O autor.

dados de acelerômetro, que utiliza a Equação 4.4 como base para também converter os valores dos três eixos em um só valor. O valor resultante para um dos pontos deve ser menor do que 1G, em que o G significa o valor da aceleração da gravidade.

$$GS(t_i) = \sqrt{{G_y}^2(t_i) + {G_z}^2(t_i)} \tag{4.6}$$

Em seguida, a partir do instante do vale, quando for identificado, o algoritmo busca um pico nos próximos quarenta (40) milissegundos. Esse pico é identificado com um valor superior a 6G, correspondendo a 6 vezes o valor da aceleração da gravidade. O próximo passo consiste de calcular o desvio padrão dos dados em um intervalo de tempo compreendido entre 40 milissegundos antes e depois do instante do vale. Para que uma queda seja caracterizada, o

Figura 27 – Fluxo de execução do algoritmo de (HSIEH *et al.*, 2014)



Fonte: Adaptado de (HSIEH *et al.*, 2014)

desvio padrão tem que ser maior do que 1.5G.

Por fim, o último passo faz um novo cálculo com base nos eixos do acelerômetro e é demonstrado na Equação 4.7. Ela faz o somatório da soma dos módulos dos três eixos do acelerômetro pelos dois (2) segundos seguintes ao instante do vale. O valor desse somatório deve ser inferior à 200G.

$$VA(t_i) = \sum_{i=1}^{n} [|A_x| + |A_y| + |A_z|] \tag{4.7}$$

4.4.3.2.3 Support Vector Machine

O algoritmo Support Vector Machine (SVM) é um algoritmo de Aprendizagem de Máquina do tipo supervisionado que pode ser utilizado para classificação ou regressão (ALZUBI *et al.*, 2018) e que é indicado quando não há um conhecimento sobre o domínio (RUSSELL *et*

al., 2009). Ele é dito supervisionado, pois é preciso ensiná-lo quais são as classes (categorias) que os dados podem ser divididos. Quanto à classificação, ele separa os dados nas categorias aprendidas e a regressão trata da predição de resultados. Para este trabalho, foi utilizada a forma de classificação.

É importante destacar que o SVM cria um espaço n-dimensional, sendo o *n* o número de *features* (características) presentes nos dados (ALZUBI *et al.*, 2018). A partir de então, o SVM busca criar planos de forma a separar os dados conforme as *features* utilizadas. Assim, o algoritmo aprende e replica o mesmo princípio para os novos dados.

No caso do Ágape, foi inserido o SVM desenvolvido e implantado para o WatchAlert após a criação do servidor na Nuvem. Foram realizados testes, que resultaram também nas lições aprendidas 5 e 6, e na definição das *features* a serem adicionadas aos dados brutos e o *timestamp*, que foram:

1. valor máximo;
2. valor mínimo;
3. raiz quadrada média (*root mean square*); e
4. curtose.

Para utilizar o SVM, também é preciso definir os hiperparâmetros, configurações utilizadas para determinar como o algoritmo deve atuar. Assim, a configuração ficou como a seguir:

- ***Kernel***: linear
  - Indica a operação de operação matemática a ser aplicada para a criação dos modelos
- **Custo**: 10
  - Parâmetro utilizado para compensar erros cometidos durante a fase de treinamento
- **Gama**: 0.001
  - Parâmetro de ajuste do hiperplano visando a generalização dos resultados

4.4.3.2.4 Random Forest

O *Random Forest* (RF) é um técnica de Aprendizagem de Máquina de classificação em conjunto (SARKER, 2021) e é baseada na Árvore de Decisão (*Decision Tree*), que classifica os dados em forma de árvore a partir da raiz até as folhas, podendo ter um nó de decisão em cada nó filho até chegar a folha, a qual contém a decisão (ALZUBI *et al.*, 2018).

No caso do *Random Forest*, ele é um conjunto de árvores de decisão que executam

em paralelo analisando diferentes amostras do conjunto de dados, que chegam ao resultado final a partir de uma decisão por maioria ou média dos resultados (SARKER, 2021). Para usar o algoritmo, foram definidos os hiperparâmetros e o melhor resultado foi obtido com:

- **Número de árvores**: 100
  - Indica quantas árvores serão criadas em paralelo
- **Profundidade da árvore**: 2
  - Indica quantos nós devem ser criados para obter o resultado

Além disso, as *features* utilizadas para obter o melhor são as mesmas utilizadas para o SVM: i) valor máximo; ii) valor mínimo; iii) raiz quadrada média; e iv) curtose.

4.4.3.2.5 K-nearest neighbors

O último algoritmo adicionado ao Ágape foi o *k-nearest neighbors* (KNN) baseado no modelo gerado em Hussain *et al.* (2019). Esse algoritmo é considerado de aprendizado lento, pois ele não gera um modelo geral, uma vez que armazena todas as instâncias de treinamento e realizando cálculos de similaridades entre o novo valor e os armazenados (SARKER, 2021).

Assim, é preciso indicar qual a quantidade de nós vizinhos que é necessário analisar para encontrar o resultado. Segundo Sarker (2021), o resultado vem a partir de uma votação entre o K vizinhos mais próximos de cada ponto analisado, sendo esse número de vizinhos o maior problema do algoritmo. No caso do modelo utilizado, foi utilizado o valor de K = 1 e com o cálculo é realizado com a distância Euclidiana.

Quanto às *features*, foram extraídas quatro (4), sendo duas delas iguais às utilizadas nos modelos anteriores: valor máximo e valor mínimo. As outras duas são média e variância.

4.4.4 Agregação de Dados

Este módulo é executado a partir do *Manager* para que ele possa unir os dados de dispositivos que analisam propriedades iguais depois dos dados processados no módulo de Processamento de Dados serem armazenados no banco de dados temporal. Dessa forma, o objetivo deste módulo é unir os dados e gerar apenas uma resposta para dados de dispositivos similares, buscando a maior precisão na identificação de quedas e outras situações que possam indicar a causa da queda.

Para atingir esse objetivo, o módulo utiliza o conceito de Episódios, os quais são qualquer evento identificado por um dispositivo de saúde segundo (REDA *et al.*, 2018) e no caso

deste trabalho é qualquer dispositivo vestível. Dessa forma, um episódio contém um intervalo de tempo que caracteriza o evento, o qual é indicado pelo tempo de coleta do primeiro e do último dado enviado, a informação indicativa do dispositivo e do sensor utilizados, bem como a propriedade que eles monitoram, o usuário e o resultado do dado processado.

Quanto ao primeiro e ao último instantes dos dados, é importante destacar o comportamento que pode ser distinto para cada dispositivos. Considerando como exemplo o glicosímetro, ele pode enviar uma tupla com o dado que é válido por horas, sendo o último instante para aquele dado o instante inicial da próxima coleta. Enquanto que para um acelerômetro, que durante uma coleta pode enviar 80 tuplas com os dados a serem analisados, o instante inicial é o da tupla 1 e o instante final corresponde ao da tupla 80.

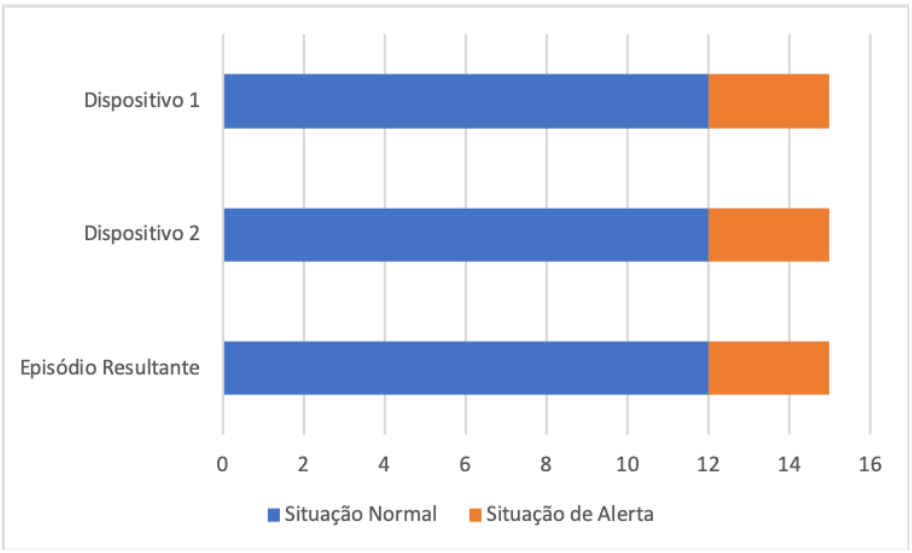
Isto posto, o módulo consulta o banco de dados temporal na busca pelos dados mais recentes referentes às quedas e as possíveis causas, os quais foram processados pelo módulo de Processamento de Dados e inseridos pelo módulo *Manager*. Essa busca permite identificar todas as situações que foram analisadas, sejam elas indicativas de um comportamento normal ou não. Feito isso, os dados gerados por dispositivos que observam as mesmas propriedades do usuário começam a ser agregados.

A Figura 28 contém representações de episódios detectados por diferentes dispositivos, mas que observam a mesma propriedade, como “aceleração do corpo”. Durante o intervalo de tempo de 0 à 12, dois dispositivos identificaram que a situação encontrava-se normal, porém de 12 a 15, eles identificaram que houve uma queda. Assim, o episódio resultante será semelhante ao que foi detectado pelos dispositivos e apresentado no terceiro eixo intitulado “Episódio Resultante” na Figura 28.

Outro cenário que pode ocorrer é o apresentado na Figura 29. Nele, três dispositivos detectam eventos de queda, porém em instantes diferentes, o que pode resultar em dúvidas e é preciso que o módulo gerencie os resultados.

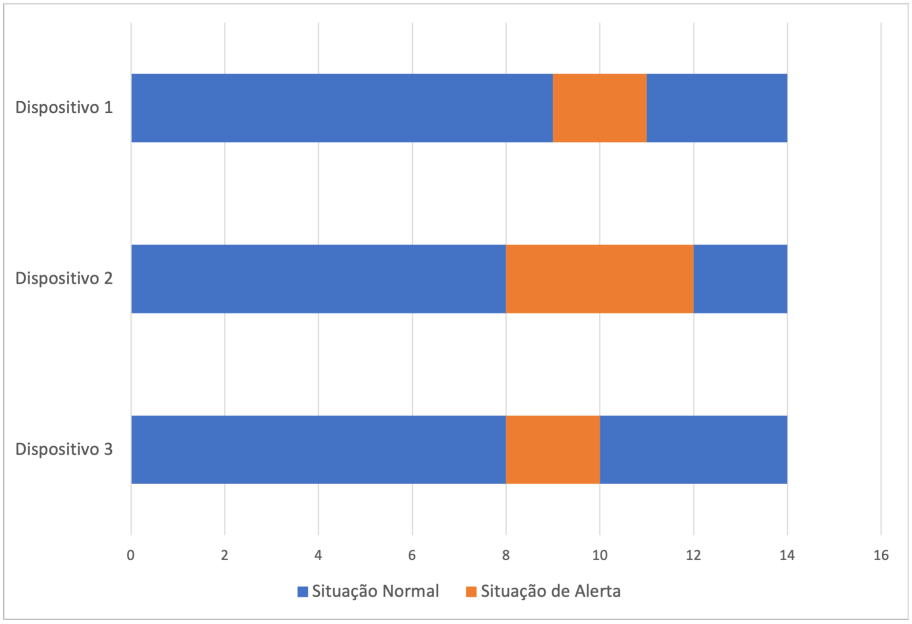
Nesse momento, as prioridades de cada dispositivo, adicionadas no *Manager*, são recuperadas para que os dados sejam unificados. Em seguida, são analisados quais dispositivos enviaram os dados recuperados no banco de dados com o intuito de filtrar quais as prioridades que devem ser utilizadas durante esta etapa. Ressalta-se que se fossem usadas as prioridades dos dispositivos que não enviaram dados no momento, mas que estão registradas no *Manager*, problemas nos cálculos para a identificação das quedas e suas possíveis causas poderiam ocorrer. Esse fato decorre da não correspondência entre os dados encontrados e os dispositivos entendidos

Figura 28 – Episódios gerados por dois dispositivos com dados correspondentes



Fonte: O autor.

Figura 29 – Episódios gerados por dois dispositivos com dados não correspondentes



Fonte: O autor.

pelo Ágape, o que poderia diminuir a importância de um dispositivo naquele instante.

A soma de todas as prioridades identificadas deve ser igual 1 e, para isso, é definido um valor base para as prioridades que deve ser proporcional às prioridades dos dispositivos ativos e que estão coletando e enviando dados para o Ágape. Esse valor base é identificado na equação 4.8 pela variável "v_b". Considere três dispositivos que observam a mesma propriedade (e.g., aceleração do corpo) com prioridades 2, 3 e 1, respectivamente. Com os valores preenchidos, a equação preenchida ficaria conforme a Equação 4.9. O que resultaria em um valor de $v_b = \frac{1}{6}$. Para

os dispositivos, o valor utilizado para o cálculo das prioridades ficaria $\frac{1}{3}$, $\frac{1}{2}$ e $\frac{1}{6}$, respectivamente.

$$p_{d1} \cdot v_b + p_{d2} \cdot v_b + ... + p_{dn} \cdot v_b = 1 \tag{4.8}$$

$$2 \cdot v_b + 3 \cdot v_b + 1 \cdot v_b = 1 \tag{4.9}$$

Assim, para o cálculo da prioridade utilizada em cada dispositivo, pode ser sumarizado pela Equação 4.10. Em seguida, o valor da prioridade de cada dispositivo é utilizado para preencher a matriz linha de tamanho n, onde n é a quantidade de dispositivos. Essa matriz é multiplicada por uma matriz vertical com os valores iguais ao das probabilidades de queda indicados pelos algoritmos, sejam de *thresholds* ou de ML.

$$p = 1 / \sum_{i=1}^{n} p_{di} \tag{4.10}$$

Ao final do cálculo da equação 4.11, há o resultado final (R_f) da agregação dos dados. Isso consiste em receber um percentual de chance da ocorrência de uma queda, considerando aqueles comportamentos registrados pelos dispositivos, por mais que os mesmos apresentem resultados divergentes. Dessa forma, nenhum valor recuperado pelos dispositivos é desconsiderado e isso contribui para uma identificação mais precisa das quedas e as possíveis causas, uma vez que o valor final é um valor percentual da chance de ter uma situação de alerta.

Por fim, os valores resultantes (R_f) da operação realizada pela equação 4.11 são armazenados no banco de dados temporal para que sejam recuperados no módulo de Tomada de Decisão, descrito na Seção 4.4.5. Esse armazenamento contém o instante da decisão, a propriedade observada, a qual deve ser comum para os dispositivos que geraram um resultado, o usuário e o próprio resultado.

$$\begin{bmatrix} pe_{d1} & pe_{d2} & \cdots & pe_{dn} \end{bmatrix} * \begin{bmatrix} r_{d1} \\ r_{d2} \\ \vdots \\ r_{dn} \end{bmatrix} = R_f \tag{4.11}$$

4.4.5 Tomada de Decisão

O último módulo é o de Tomada de Decisão, cujo objetivo é recuperar os dados unificados pelo módulo de Agregação de Dados tendo como base as propriedades observadas por cada dispositivo. A partir de então, o módulo busca no banco de dados os dados mais recentes sobre as propriedades observadas pelos dispositivos integrados à solução para iniciar o processo de tomada de decisão.

Essa busca inicia-se com a verificação da ocorrência ou não de uma queda, que consiste, primeiramente, de uma análise dos episódios de queda registrados pelo módulo de Agregação de Dados nos últimos trinta (30) segundos. Esse valor ajuda a assegurar que dados enviados não sejam ignorados ao mesmo tempo que não são antigos o suficiente para gerar alertas repetidos. Além disso, os dados de acelerômetro são gerados em milissegundos, o que gera muita informação em trinta segundos. Além disso, é feita uma busca por aqueles dados que possuem o R_f, que é o resultado de uma ponderação sobre as probabilidades dos algoritmos, maior do que um valor mínimo.

Foi definido o valor de 50% para o R_f mínimo para ser compreendido como queda. Esse valor foi especificado tendo como base um valor que indique uma probabilidade maior ou igual de queda do que atividade diária. Isso também ocorreu, pois todo evento de queda deve ser alertado, optando por ter mais situações de alerta de quedas do que ignorar quedas que realmente ocorreram. Destaca-se que esse limite também pode ser alterado conforme as necessidades da aplicação a ser desenvolvida.

Havendo dados com percentual de probabilidade de ocorrer maior ou igual a 50% nos últimos trinta segundos, o módulo passa a analisar os dados das outras propriedades observadas. Tais dados permitem indicar as possíveis causas da queda identificada. Para realizar tal identificação, os dados coletados são confrontados com padrões de normalidade para essas propriedades, os quais são de fácil alteração ou inserção, uma vez que podem ser adicionados mais dispositivos que monitoram novas propriedades o que exige que novos padrões precisam ser especificados.

O módulo considera o padrão de comportamento normal para duas propriedades: a glicemia e os batimentos cardíacos. Para a glicemia e para os batimentos cardíacos, os valores padrões foram recuperados em diretrizes determinadas pelas Sociedades Brasileiras de Endocrinologia e Cardiologia. Considerando os valores para a glicemia, ela podem ser considerada baixa se for inferior a 70 mg/dl e alto se for superior a 110 mg/dl. No caso dos

batimentos cardíacos, os valores são considerados baixos quando são inferiores a 60 bpm e altos acima de 120 bpm.

Na sequência, o módulo realiza outra consulta ao banco de dados temporal, buscando por valores em um intervalo de tempo específico para cada tipo de propriedade observada. Por exemplo, para o batimento cardíaco a análise considera os últimos 10 minutos e para a glicemia as últimas 4h, uma vez que a frequência de coleta é diferente para cada caso. Em seguida, os dados retornados são analisados para verificar se alguns deles correspondem à valores fora dos limites especificados, o que pode sugerir a condição que causou a queda do usuário.

Após isso, os dados de queda e das possíveis causas são analisados para identificar os pontos de alerta para as propriedades como batimentos cardíacos e glicemia. Caso haja uma queda e uma situação de alerta, elas são relacionadas e indicadas para o módulo *Manager*, cujo papel nesse momento é reportar para o dispositivo que enviou os dados para o Ágape. Destaca-se que cada desenvolvedor pode adicionar mais sensores e possíveis causas monitoradas, precisando mapear as condições consideradas normais e os limites que caracterizam como valores altos ou baixos.

4.5 Modelo de Dados

Para que o Ágape possa lidar com dados distintos é importante destacar que foi especificado um modelo de dados orientado a objetos devido às suas características, as quais são as mesmas da orientação a objeto, como herança, polimorfismo, interface, entre outros, além de ser de fácil extensão. Assim, nesta seção é apresentado o modelo de dados orientado à objetos do Ágape.

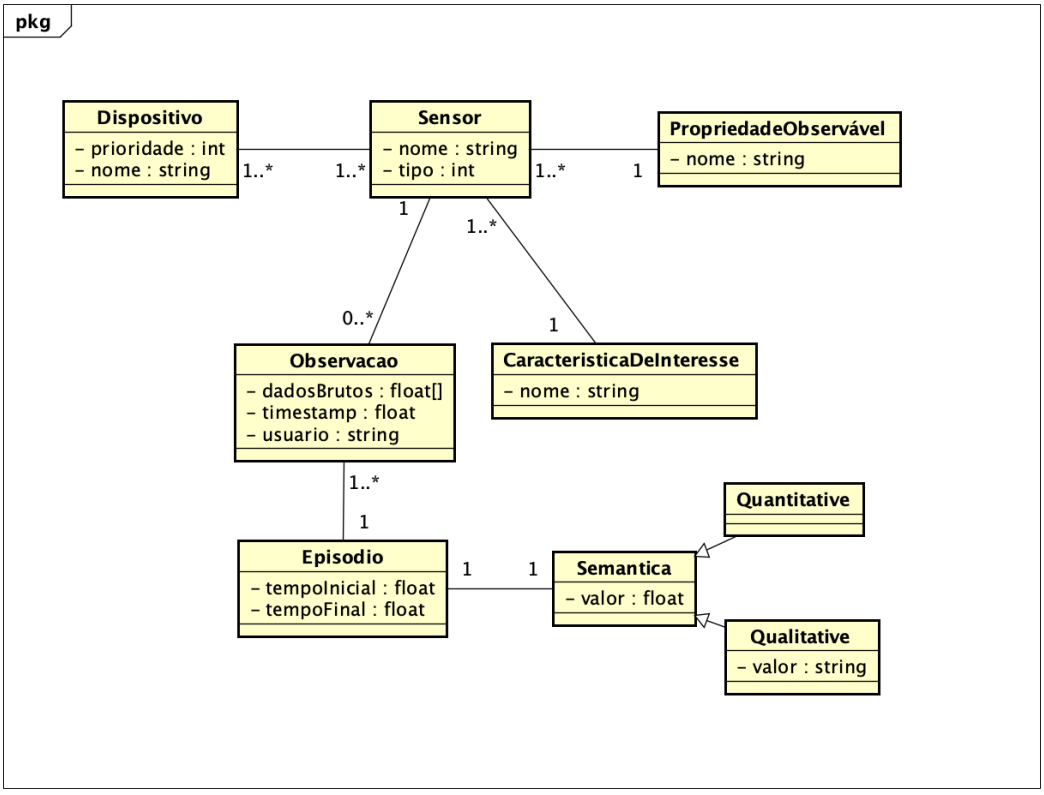
Primeiramente, é importante destacar que o *framework* deve lidar com diferentes dados, os quais podem variar em relação à tipo (e.g., inteiro, float), formato, pois cada dispositivo pode coletar diferentes quantidades de dados (e.g., 1 medição com 1 dado, dados de 3 eixos). Além disso, é essencial que o modelo possa ser estendido para suportar dados oriundos de novos dispositivos.

O modelo de dados do Ágape foi inicialmente baseado no nos conceitos de duas ontologias, uma vez que elas contém a representação de conceitos utilizados com sensores, como a SOSA (*Sensor, Observation, Sample e Actuator*) (W3C, 2016; JANOWICZ *et al.*, 2018), que é a base para a *Semantic Sensor Network* (SSN) (W3C, 2018) e episódios (NOGUEIRA *et al.*, 2018), abordando dados relacionados com o espaço-tempo.

Embora tivesse a base dos conceitos em ontologias e pudesse ter sido criado uma para o Ágape, uma ontologia não seria tão simples para estender e evoluir para lidar com os requisitos específicos de cada aplicação que utiliza o Ágape. Isto posto, o modelo de dados do Ágape foi criado como orientado à objetos, mas mantendo os conceito e as relações entre eles e pode ser visto na Figura 30.

Quanto aos objetos, o primeiro deles é o dispositivo (classe: *Dispositivo*), que representa os dispositivos de monitoramento, como *smartwatch*, *smartphone*, etc. Esse objeto deve possuir um *nome* parar especificá-lo e diferenciá-lo dos demais objetos, além de ter um atributo correspondente à *prioridade* de cada dispositivo, valor que é utilizado durante a agregação dos dados.

Figura 30 – Modelo de dados orientado a objetos do Ágape



Fonte: O autor.

Um dispositivo deve ter pelo menos um (1) sensor, o qual é responsável por realizar as coletas e para diferenciar cada sensor presente no dispositivo, eles devem possuir um *nome*, como *acelerômetro*, *giroscópio*, *barômetro*. Há ainda outro atributo relevante associado à sensor que permite dissociá-lo dos demais, que é o *tipo*, a ser identificado na tupla de dados enviada ao Ágape.

Existem outros conceitos relacionados mais diretamente ao sensor, devido à ligação

com as coletas que são realizadas por ele. Assim, é importante registrar que cada sensor possui uma *característica de interesse* (*CaracterisitcaDeInteresse*), a qual representa uma qualidade observável a ser medida pelo sensor (JANOWICZ *et al.*, 2018). Associado à *característica de interesse*, há a *propriedade observável* (*Observable Property*), que corresponde ao que deve ser medido pelo sensor. Um exemplo de característica de interesse é a *glicemia* e a *movimentação do corpo*, enquanto as propriedades podem ser o nível de açúcar no sangue e a aceleração do corpo.

Além disso, um sensor realiza *Observações* (classe: *Observação*), as quais são representações das medições realizadas pelos sensores, e, portanto, deve possui o *timestamp* de cada coleta, além dos dados brutos de cada sensor. Por exemplo, uma observação corresponde a uma tupla contendo o *timestamp* e os três eixos (X, Y e Z) de um acelerômetro ou o valor coletado pelo glicosímetro.

A partir das *observações*, é possível criar os *episódios* (classe: *Episódio*) como descrito na Seção 4.4. Cada episódio contém o tempo inicial de cada coleta, o instante final e o usuário associado. Além disso, ele possui uma *semântica* (classe: *Semântica*), que inicialmente pode ser um número real, mas que pode ser alterado por cada classe filha. No caso da classe que representa valores qualitativos (classe: *Qualitativa*), o valor é um texto. Já a *quantitativa* (classe: *Quantitativo*), mantém o número real.

4.6 Arquitetura do Ágape

Conforme descrito anteriormente, o Ágape é um *framework* para desenvolvimento de soluções IoHT voltadas para a detecção de quedas e agrega essa informação com dados sobre as possíveis causas da queda. Ele foi projetado executar em um servidor, que pode ser na nuvem, e possui uma arquitetura em camadas. Essa arquitetura é composta por cinco módulos e ela segue o que é demonstrado na Figura 24. Os módulos são detalhados na Seção 4.4.

4.6.1 Diagrama de Classe

O diagrama de classes apresentado na Figura 31 contém a estrutura das classes e os pacotes que correspondem ao *framework* Ágape. Visando facilitar o entendimento e o desenvolvimento, cada módulo do Ágape foi especificado como um pacote. Dessa forma, o diagrama é composto de cinco pacotes: i) *Listener*; ii) *Manager*; iii) *Processamento de Dados*; *Agregação de Dados*; e v) *Tomada de Decisão*.

A considerar o módulo *Listener*, ele contém apenas uma classe, a *Servidor*, cuja função é estabelecer a comunicação com os dispositivos que enviam dados para serem processados. Nessa comunicação, o servidor envia o resultado do processamento, o qual informa se houve ou não queda, bem como a causa, sendo essas associações realizadas pelo módulo *Tomada de Decisão*. Além disso, este módulo possui a capacidade de trabalhar com fluxos de dados, como consta na lição aprendida 01.

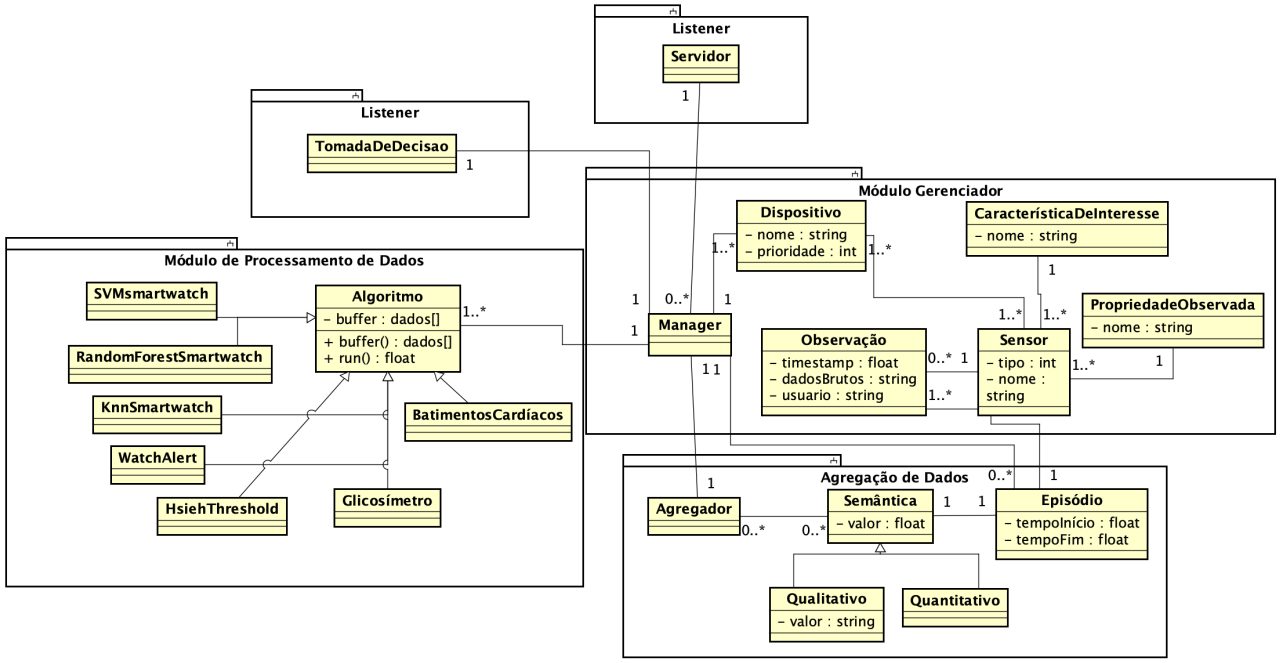
Na sequência dos módulos, há o *Manager*, o qual é composto de cinco (5) classes, sendo a primeira, também chamada *Manager* e que há uma relação com a classe *Servidor*. Isso ocorre devido ao papel desempenhado pela classe *Manager*, que é de receber os dados e gerenciá-los, além de executar os demais módulos. Para auxiliar o gerenciamento dos dados, inicia-se a integração com o modelo de dados descrito na Seção 4.5. Assim, foram criadas quatro (4) classes, sendo uma a de dispositivos (classe: *Dispositivo*) a segunda é a de sensores (classe: *Sensor*), a terceira é a propriedade observável (classe: *PropriedadeObservavel*).

A quarta classe, por sua vez, foi criada para registrar os dados a serem processados pelos algoritmos, sendo chamada de *Buffer*. Ela está relacionada com a classe *Manager*, mas também a *Sensor* e a *Dispositivo*, pois um *buffer* é gerado para uma chave representada pelo par <*dispositivo, sensor*>. Essa chave é criada para evitar o embaralhamento dos dados com o de outros dispositivos e sensores, o que geraria lixo e faria os algoritmos interpretarem os resultados de forma equivocada. Ademais, essa classe ajuda a organizar a quantidade de dados recebidos de maneira que um algoritmo que requisite, por exemplo, oitenta (80) tuplas, ele consegue armazenar antes de submetê-los ao algoritmo, evitando, assim, que o resultado seja equivocado.

Analisando o módulo *Processamento de Dados*, há a definição de uma classe, identificada como *Algoritmo*, cuja função é fornecer uma estrutura para a definição de outros algoritmos de forma que possam ser entendidos pelo Ágape e permita a extensão dele para aceitar novos algoritmos. Dessa forma, para que um desenvolvedor consiga adicionar um novo algoritmo, ele deve herdar da classe *Algoritmo*. Essa classe ainda contém um atributo *data* que permite criar um *buffer* local para a classe, além de manipular os dados sem impactar nos dados do *buffer* da classe *Manager*.

Além disso, a classe *Algoritmo* ainda possui dois métodos, *buffer* e *run*. O primeiro é usado para gerar o *buffer* local a partir dos dados encaminhados pelo módulo *Manager*, enquanto o segundo é um método abstrato que deve ser implementado para cada algoritmo adicionado.

Figura 31 – Diagrama de Classe do Ágape



Fonte: O Autor.

Como padrão, são providos pelo Ágape cinco algoritmos, os quais são para dados de *smartwatch*, sendo dois deles de *thresholds* e três de Aprendizagem de Máquina, o *Support Vector Machine*, o *Random Forest* e o *KNN*, conforme descrito na Seção 4.4.3.2. Também foram adicionados outros dois algoritmos para extrair os dados do glicosímetro e de batimentos cardíacos.

Na sequência, há o módulo *Agregação de Dados*, que contém a geração dos *episódios* (classe: *Episódio*) em que cada um possui um significado semântico (classe: *Semântica*), que pode ser quantitativo (classe: *Quantitativo*) ou qualitativo (classe: *Qualitativo*). Além disso, nesse módulo há uma classe responsável pela agregação dos dados, a *Agregador*. É nele em que ocorre todo processo de unificação de episódios, gerando episódios únicos para propriedades similares observadas.

O último módulo é o *Tomada de Decisão*, cujo objetivo é tomar a decisão a partir dos episódios gerados anteriormente e possui apenas uma classe, a *TomadaDeDecisao*. Nessa classe é onde ocorre a tomada de decisão explicada na Seção 4.4.

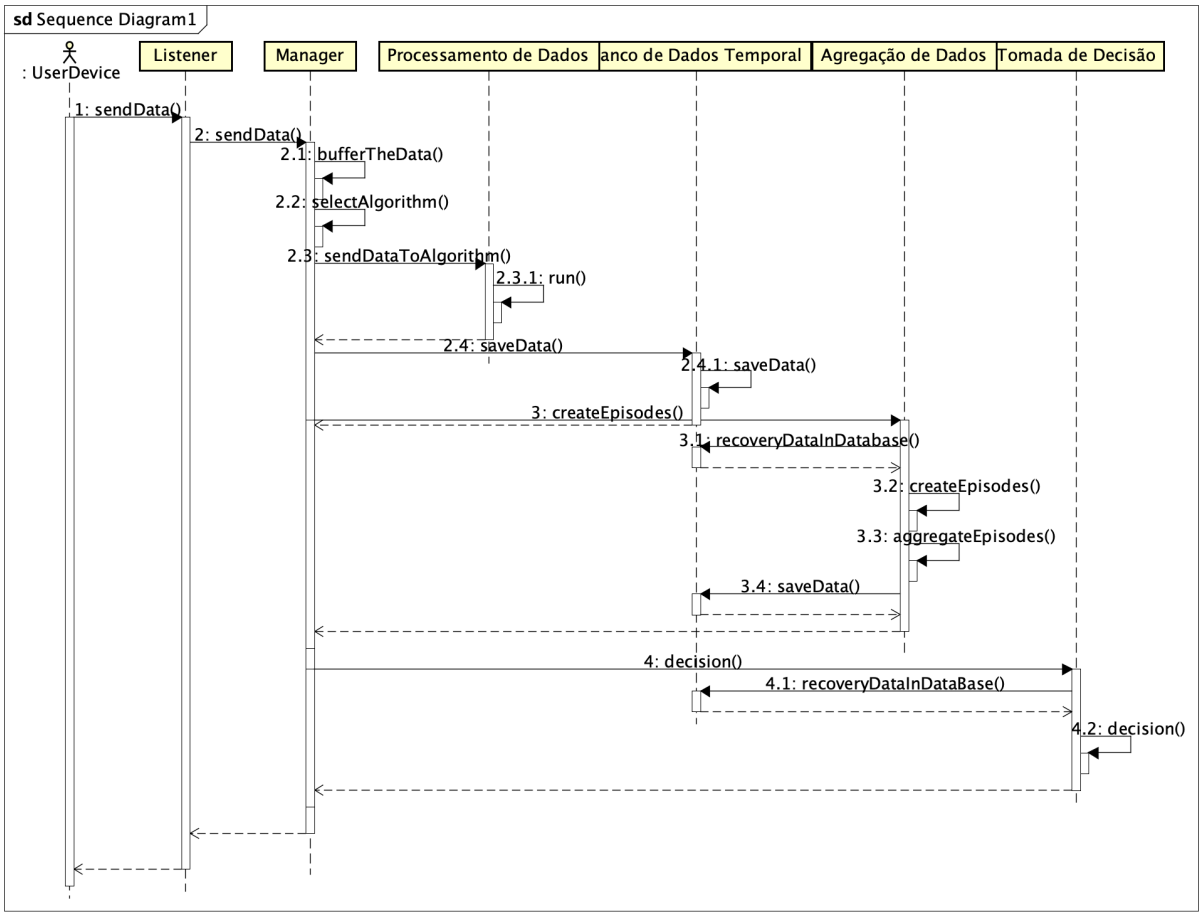
4.6.2 Diagrama de Sequência

Para entender o funcionamento do Ágape é importante compreender as interações entre os módulos, bem como o fluxo e as mensagens trocadas eles entres. Para isso, foi criado o diagrama de sequência apresentado na Figura 32. Nele, estão indicados os módulos, o dispositivo

do usuário e o banco de dados temporal utilizado para armazenar as informações.

O fluxo inicia com o dispositivo de monitoramento do usuário enviando os dados coletados para o *Listener* que, por sua vez, repassa para o *Manager*. Em seguida, esse módulo armazena os dados em *buffer* e seleciona o algoritmo a ser utilizado para processar aqueles dados tendo como base o dispositivo, o sensor que coletou e o algoritmo definido pelos desenvolvedores, uma vez que vários algoritmos podem ser adicionados para o mesmo par <*dispositivo, sensor*>.

Figura 32 – Diagrama de Sequência



Fonte: O Autor.

Em seguida e ainda no módulo *Manager*, ele envia a mensagem para o módulo *Processamento de Dados* com a finalidade dele executar o algoritmo identificado anteriormente. Ao final da execução, esse módulo retorna o valor encontrado para o módulo *Manager* que deve acessar o banco de dados temporal a fim de realizar o armazenamento daqueles dados, que deve conter o tempo inicial, tempo final, resultado encontrado, usuário, dispositivo e sensor, o que caracteriza o início da geração dos episódios.

Após salvar os dados, o *Manager* dispara a execução do módulo *Agregação de Dados* para que o mesmo possa organizar os episódios a partir dos dados armazenados anteriormente,

gerando os episódios a serem unificados a seguir e conforme descrito na Subseção 4.4.4. Após realizar a agregação, ele salva os dados no banco de dados e sinaliza para o *Manager* continuar com a execução, quando as operações forem executadas com sucesso.

Por fim, o *Manager* envia a mensagem para o módulo *Tomada de Decisão*, o qual recupera no banco de dados temporal os dados dos episódios mais recentes gerados e passa a analisar os resultados e a buscar a possível causa da queda, caso uma tenha sido identificada. Esse módulo, por sua vez, retorna o valor para o *Manager*, que encaminha para o *Listener*, que retorna para o dispositivo do usuário. Nesse momento, cabe a cada aplicação tratar a mensagem retornada.

Ressalta-se que esse é o fluxo principal de execução do Ágape. Contudo, podem existir erros durante a execução e, quando isso ocorrer, é lançada uma exceção a ser tratada pela aplicação do usuário. Também é importante destacar que os dados utilizam uma fila e aguardam um tempo até serem processados ou não conseguir executar corretamente, o que permite que a fila possa passar para os próximos dados.

4.7 Instanciação do Ágape

Visando instanciar os conceitos explicitados anteriormente neste Capítulo, como a arquitetura e o funcionamento dos módulos, foi implementado o Ágape utilizando a linguagem de programação *Python* (PSF, 2022) na versão *3.7*. Para o banco de dados temporal, foi utilizado o InfluxDB (INFLUXDATA, 2022) na versão *1.8*.

Isto posto, inicia-se o detalhamento dos módulos. O *Listener* tem o papel de estabelecer a comunicação entre os dispositivos e o *Manager*, e, para dar suporte a essa relação foi o utilizado o *RabbitMQ* (PATHAK; KALAIARASAN, 2021). Ele é um *framework open source* comum que implementa diferentes meio de comunicação, como *publish-subscribe*, filas, roteamento e chamada de procedimento remoto. Para o Ágape, foi utilizada a forma de filas, que recebe os dados, processa e retorna o valor para o dispositivo do usuário para que a aplicação possa tratar a informação recebida.

Como descrito na Seção 4.4.1, o *Listener* recebe uma mensagem enviada pela aplicação que está executando no dispositivo do usuário. Contudo, surge um problema quando cada aplicação define o seu padrão de envio das tuplas, por exemplo, entre colchetes, parênteses, ponto e vírgula. Assim, ao receber cada conjunto de dados, é preciso extraí-los para submetê-los em forma de lista para que o *Manager* possa tratá-los corretamente. Essa extração é feita no

método *extract_body* contido na linha 2 do Algoritmo 1 e pode ser ajustado para tratar os dados de acordo com cada aplicação. Ressalta-se que o *body* corresponde ao corpo da mensagem enviada pelo dispositivo do usuário. A terceira linha corresponde a especificação da fila e que estará "ouvindo" todas as requisições feitas nela.

Algoritmo 1: Trecho do código que recebe a requisição do dispositivos

```
def on_request(ch, method, props, body):

    response = manager(extract_body(body))
    channel.basic_consume(queue='rpc_queue', on_message_callback=on_request)
```

Ainda na linha 2 do Algoritmo 1, ele chama o método *Manager*, que uma instância do módulo *Manager* como demonstrado no Algoritmo 2. Em seguida, há a inicialização dos dispositivos configurados para serem entendidos pelo Ágape, que é apresentado no Algoritmo 3. Em seguida, a instância do *Manager* recebe os dados para serem processados e retorna o valor para o *Listener*.

Algoritmo 2: Trecho do código que recebe a requisição do dispositivos

```
def manager(data):
    print("Starting the manager...")
    instance_manager = Manager()
    instance_manager.initializeDevices()
    return instance_manager.receives(data)
```

Quanto ao Algoritmo 3, é instanciado um *smartwatch* e um sensor, que é o acelerômetro. Em seguida, foram adicionados três algoritmos relacionados ao sensor e depois foi definido qual deles deveria executar ao receber os dados. Além disso, o dispositivo é associado ao sensor, além de ter sua prioridade definida. Ao final, duas ações ocorrem: i) a adição do dispositivo a uma estrutura de dados de mapeamento <*Chave,Valor*>, chamada de dicionários em Python, para indicar ao Ágape que aquele dispositivo existe e deve ser compreendido pelo *framework*; e ii) criar o *Buffer* para a chave que é o par <*dispositivo, sensor*>.

Para executar os algoritmos, é preciso que os algoritmos herdem da classe *Algoritmh* como comentado nas seções anteriores. Essa classe é apresentada no Algoritmo 4 e possui os dois métodos citados na Subseção 4.4.3: i) *buffer* que armazena os dados recebidos do *Manager*; e ii) *run*, o qual deve ser implementado por todos os algoritmos, uma vez que cada um possui

Algoritmo 3: Trecho do código que inicializa os dispositivos

```
def initializeDevices(self):
    # create a new device
    smartwatch = Device("smartwatch")

    # create a new sensor
    accelerometer = Sensor()
    accelerometer.defineName("accelerometer")
    accelerometer.defineSensorType(1)
    accelerometer.defineObservableProperty("Gravity")

    # add algorithm to the sensor
    accelerometer.addAlgorithm(pm.SmartWatchThresholdAlgorithm())
    accelerometer.addAlgorithm(pm.SVMSWAlgorithm())
    accelerometer.addAlgorithm(SVM)

    # choose the algorithm
    accelerometer.chosenAlgorithm_ = SVM()
    # add new sensor to the device
    smartwatch.addSensor(accelerometer)
    smartwatch.definePriority(3)
    dictDevices[smartwatch.deviceName_] = smartwatch

    ####### add buffer to smartwatch
    bufferDevices["0"] = Buffer(smartwatch, accelerometer)
```

um comportamento diferente.

Algoritmo 4: Classe Algoritmo a ser herdada por outras classes

```
class Algorithm():
    def __init__(self):
        self.dataBuffered = []
        pass

    @abc.abstractmethod
    def run(self):
        print("Abstract Method")
        pass

    def buffer(self, data):
        self.dataBuffered = data
        result = self.run()
        self.dataBuffered.clear()
        return result

    tBuffer = threading.Thread(target = buffer)
```

Após o processamento e a geração dos dados, eles são estruturados para serem armazenados como episódios no banco de dados temporal. No Algoritmo 5, a variável *generated_episode* contém o resultado dos processamentos realizados pelos algoritmos, enquanto são armazenadas as informações sobre o dispositivo, sensor, usuário, tempos inicial e final, sendo o tempo inicial atribuído na variável *generated_episode*.

Algoritmo 5: Trecho que salva os episódios no banco de dados

```
self.client.write_points(dataframe=generated_episode, measurement="episodes",
                         protocol="line",
                         tags={'device': bufferAux.device_.deviceName_,
                               "sensor": bufferAux.sensor_.sensorName_,
                               "property": bufferAux.sensor_.observableProperty_.name_,
                               "timestampFinal": str(finalTimeStamp),
                               "user": str(user)})
```

Em seguida, no módulo *Agregador* ocorre a unificação dos episódios de uma mesma propriedade observada por diferentes sensores e dispositivos em apenas um, o episódio resultante. Para isso, foram criados quatro (4) métodos principais que, em conjunto, recuperam as informações, verificam os dados, executam as operações de agregação e retornam o valor para o *Manager*, além de salvar no banco de dados. Ressalta-se que existem outros métodos auxiliares para gerar os episódios resultantes conforme descrito na Subseção 4.4.4.

No Algoritmo 6, são apresentados os quatro métodos citados anteriormente com os comentários que os descrevem. O método *aggregation* é o método principal e que organiza a execução dos métodos para alcançar o resultado esperado. Para isso, ele utiliza outros três métodos principais para recuperar os episódios que possuem as mesmas propriedades, além de compor episódios resultantes com os dados iguais (*composeEpisodesSameDeviceAndSensor*) ou com diferentes (*resultantEpisodesComplete*).

Destaca-se que ainda nesse módulo, é possível que os desenvolvedores façam ajustes, uma vez que podem alterar a forma de fazer a agregação dos episódios. Atualmente, é feito de forma a considerar a prioridade apenas, mas podem ser implementadas novas formas de fazer, como a utilização do método Kappa (WARRENS, 2012).

Algoritmo 6: Trecho do módulo Agregador

```
    """
    In this method occurs the aggregation of the data from devices with the same observable property.
    """
    def aggregation(self, client, user, times, dictDevices):...
    """
    Considering the devices and sensors that monitoring the same property and same value, this method aggregate these sensors.
    """
    def composeEpisodesSameDeviceAndSensor(self, episodes):...
    """
    This method generate the Resultant Episodes with different values during the aggregation
    """
    def resultantEpisodesComplete(self):...
    """
    This method recovery the episodes with same observable property from database
    """
    def recoveryOnlyEpisodesWithSameProperty(self, property):...
```

Por fim, o módulo de Tomada de Decisão, com trecho apresentado no Algoritmo 7, verifica se uma queda foi identificada e busca encontrar as causas. Assim, a verificação por uma queda é realizada a partir da consulta inicial com o objetivo de checar se há dados associados

Algoritmo 7: Trecho do módulo Tomada de Decisão

```
def decision(self, client, user):
    try:
        self.client = client
        global lastTime
        fall = 0
        result = self.client.query("select * from epResults where \"observableProperty\"= \'gravity\'")
        results = result["epResults"]
        time = results.last_valid_index()
        time = time.tz_localize(tz=None,ambiguous=False, nonexistent='shift_forward')
        time = time.to_pydatetime()
        query = "select * from epResults where \"userAux\"= \'" + user + "' and time > '" + str(time) + "' - 30s and \"observableProperty\"= \
        result = self.client.query(query)
        if result != None and len(result) > 0:
            items = result["epResults"]
            time_for_fall = items["0"].idxmax()
            line_of_table = items.loc[[time_for_fall],["0"]]
            line_in_dict = line_of_table["0"].to_dict()
            fall = line_in_dict[time_for_fall]
            time_for_fall = time_for_fall.tz_localize(tz=None,ambiguous=False, nonexistent='shift_forward')
            time_for_fall = time_for_fall.to_pydatetime()
            for key, value in dictTime.items():...
            return fall, dictProblemHigh, dictProblemLow
    except Exception as e:
        print("Deu erro")
        print(e)
    return fall, dictProblemHigh, dictProblemLow
```

Algoritmo 8: Dicionários em Python utilizados para identificar a causa

```
dictTime = {"glucose": "4h", "bpm": "10m"}
dictHigh = {"glucose": "110", "bpm": "120"}
dictLow= {"glucose": "70", "bpm": "60"}
```

à aceleração do corpo através da propriedade "gravidade "– *"select * from epResults where "observableProperty" = 'gravity' "*. A partir de então, é feita a identificação do último índice que serve de base para identificar um dado de queda com a probabilidade maior ou igual a 50%.

Na estrutura de repetição contida em *"for key, value in dictTime.items()"*, é feita uma checagem no banco de dados em busca de outras propriedades que possam indicar a possível causa. No Algoritmo 8, há a indicação das propriedades observadas, como glicose (*glucose*) e batimentos cardíacos (*bpm*), bem como dos limites que caracterizam o que é um valor baixo, alto e até o tempo em que um desses dados ainda continua válido (e.g.,a validade da glicose é de 4 horas e os batimentos são 10 minutos).

4.8 Limitações

O Ágape ajuda a processar os dados para identificação de quedas e causas, contudo, o *framework* possui limitações quanto à sincronização dos dados para fazer a agregação e aspectos de segurança. Quanto à sincronização, os dados são organizados de acordo com o instante em que são encontrados e os últimos registros do banco de dados são recuperados com os dispositivos e sensores. Assim, não são utilizadas técnicas de sincronização entre os dados.

Quanto à segurança, os dados são acessados apenas com o *login* de acesso ao banco de dados, mas não são armazenados com criptografia, assim como na transmissão dos dados.

Esse aspecto deve ser aprimorado, uma vez que dados de usuários armazenados são sensíveis e já existem leis[2] para manter a privacidade dos mesmos.

4.9 Conclusão

Este capítulo apresentou o Ágape, um *framework* para o desenvolvimento de aplicações para detecção de quedas e suas possíveis causas. Para alcançar esse objetivo, primeiro explicou-se o processo de desenvolvimento utilizado, seguido pela experiência adquirida na área que culminou com o conhecimento das funcionalidades comuns necessárias para a construção do *framework*. Além disso, este capítulo contemplou lições aprendidas publicadas no artigo (LINHARES *et al.*, 2020). Na sequência, uma visão do *framework* foi explicada, seguida pelos módulos que o compõem.

Além disso, como *framework*, ele possui *hot spots*, que são elementos no código que podem ser ajustados conforme a necessidade dos desenvolvedores. Um desses *hot spots* está relacionado aos algoritmos utilizados, que podem variar de acordo com os dispositivos e sensores utilizados para a detecção de quedas e suas causas. Ademais, foram descritos cinco algoritmos utilizados para detecção de quedas, sendo dois de *thresholds* e três de Aprendizagem de Máquina.

Também foi explanado sobre a arquitetura do *framework*, com a apresentação de diagramas de classe e de sequência. Por fim, foram apresentados aspectos da implementação do Ágape contendo trechos de códigos que correspondem às funções principais de todos os módulos.

[2] Como a Lei Geral de Proteção de Dados (LGPD) no Brasil

5 AVALIAÇÃO DO ÁGAPE

A avaliação do Ágape, *framework* proposto nesta tese para facilitar o desenvolvimento de aplicações de detecção de quedas associando-as com possíveis causas intrínsecas ao usuário, é descrita neste capítulo.

Para isso, este capítulo está assim dividido: a Seção 5.1 contém o planejamento da avaliação, o que inclui as hipóteses; na Seção 5.2 são apresentados os resultados da avaliação; a discussão a partir dos resultados obtidos é feita na Seção 5.3; as ameaças à validade são descritas na Seção 5.4; e a Seção 5.5 conclui o capítulo.

5.1 Planejamento da Avaliação

O Ágape foi avaliado em um estudo empírico em um ambiente sintético, o qual é definido como uma representação do ambiente real com algumas restrições, pois a reprodução integral das condições reais de um projeto podem ser muito custosas e trabalhosas, além de possuir variáveis que podem impactar na avaliação do trabalho (ZELKOWITZ *et al.*, 2003). Dois critérios foram essenciais para a definição da escolha desse ambiente sintético:

- Não havia dispositivos vestíveis suficientes para serem utilizados com os participantes; e
- A pandemia do novo coronavírus, que exigiu o distanciamento social.

Durante o planejamento da avaliação, ficou definido que a avaliação tem como objetivo analisar o Ágape quanto à facilidade de uso e utilidade, segundo a visão de desenvolvedores de aplicativos da área da saúde. Além disso, foi observado o tempo de configuração do Ágape e o tempo de desenvolvimento de uma aplicação de detecção de quedas e suas causas com o suporte do *framework*. Em suma, a avaliação consistiu de:

Analisar o ***Ágape***
para o propósito de ***avaliá-lo***
com respeito à ***facilidade de uso, utilidade e tempos de desenvolvimento e configuração***
do ponto de vista de ***desenvolvedores***
no contexto do ***desenvolvimento de uma aplicação IoHT para detecção de queda e suas causas***.

Nesse contexto, foram definidas três (3) questões de pesquisa, sendo duas (2) quantitativas e uma qualitativa. As questões de pesquisa são elencadas a seguir, iniciando com a descrição das quantitativas:

QP_{qt} 1. *O tempo de configuração do ambiente do Ágape compromete a utilização dele?* Para usar um *framework*, é importante que a configuração do mesmo não desestimule ou comprometa o seu uso, pois se o esforço para configurar for maior do que os benefícios do uso, os desenvolvedores podem optar por outras alternativas. Nesse cenário, o Ágape é avaliado, a fim de identificar se os desenvolvedores gastam mais tempo para configurar (tc) ou utilizar (tu) o Ágape? Com base nisso, foram criadas duas hipóteses, a nula e a alternativa que são apresentadas nas Equações 5.1 e 5.2.

$$H_0 : tc_a = tu_a \tag{5.1}$$

$$H_1 : tc_a < tu_a \tag{5.2}$$

QP_{qt} 2. *O tempo de desenvolvimento é reduzido quando se utiliza o Ágape?* O tempo em que uma solução leva para ser desenvolvida e estar disponível no mercado é importante, pois pode impactar na aceitação dos usuários, por exemplo. Assim, uma das formas de reduzir o tempo é com o suporte de artefatos de reuso como *frameworks*, padrões, entre outros. Nesse cenário, essa questão busca investigar se o Ágape cumpre o papel de reduzir o tempo de desenvolvimento de uma solução para detecção de quedas e suas causas. Para analisar se houve uma redução no tempo, foi feita uma comparação com o tempo de desenvolvimento do WatchAlert.

Na sequência, é descrita a questão qualitativa:

QP_{ql} 1. *Existe correlação entre os critérios de facilidade de uso e utilidade?* Considerando o objetivo do Ágape, essa questão visa identificar se os desenvolvedores percebem a utilidade do *framework* e entendem que ele atende ao que se propõe e ele é útil. Para auxiliar na medição dessa questão, um formulário baseado no TAM com respostas na escala Likert (JOSHI *et al.*, 2015).

Para a análise das hipóteses quantitativas, que são relacionadas ao tempo, os participantes registraram o tempo necessário para cada etapa, configuração e desenvolvimento, reportando-os ao final. Para a questão qualitativa, o TAM foi utilizado como base, o qual permite avaliar a percepção da facilidade de uso e de utilidade de uma nova tecnologia. Para definir as assertivas utilizadas no TAM, seguiu-se o processo definido em (DAVIS, 1989), o qual indica que o primeiro passo é elaborar tópicos a serem analisados.

Para gerar os tópicos a serem abordados, levou-se em consideração os objetivos a serem avaliados, o que gerou 8 afirmações para a percepção de facilidade de uso e 10 para a percepção de utilidade. Todas as afirmativas foram enviadas a pesquisadores, sendo três (3) nessa avaliação, para que os mesmos pudessem analisá-las a fim de identificar afirmações repetidas ou mal elaboradas. Além disso, também foi feita a análise para sugerir, caso achassem válido, o agrupamento das assertivas a partir dos tópicos abordados em cada uma. Ao final, ficaram 5 afirmativas para a percepção de facilidade de uso e 9 percepção de utilidade e sem classificação delas em subgrupos. O resultado é mostrado nas Tabelas 7 e 8.

Tabela 7 – Afirmativas da percepção de facilidade de uso

	Percepção de facilidade de uso
1	Eu me senti confuso ao usar o *framework*.
2	Eu cometi erros quando usei o *framework*, pois não sabia utilizá-lo.
3	Eu me senti frustrado frequentemente.
4	Eu precisei consultar o tutorial para usar o *framework*.
5	Eu achei fácil recuperar de erros que aconteceram no *framework*.
6	Eu achei fácil fazer o *framework* executar o que eu queria.
7	O *framework* se comportou de modo inesperado frequentemente.
8	Eu achei desconfortável utilizar o *framework*.
9	Foi fácil lembrar como executar uma tarefa no *framework* (e.g., adicionar um algoritmo).

Fonte: O autor.

Tabela 8 – Afirmativas da percepção de utilidade

	Percepção de utilidade
1	O uso de um *framework* facilitou o desenvolvimento de um software IoHT.
2	O *framework* reduziu o tempo de desenvolvimento de um software IoHT.
3	O uso do *framework* contribui para melhorar qualidade do meu trabalho.
4	O *framework* ajuda a processar os dados e melhor a detecção de quedas.
5	O *framework* é útil no desenvolvimento de software.

Fonte: O autor.

Quanto ao perfil dos participantes da avaliação, eles deveriam ter conhecimento no desenvolvimento de aplicações móveis, mais especificamente na plataforma Android, pois a avaliação consistiu do desenvolvimento de um aplicativo e o Android pode ser desenvolvido em qualquer sistema operacional (e.g., MacOS, Windows e Linux) e é uma das tecnologias mais comuns para dispositivos móveis. Além disso, os participantes deveriam ter conhecimento na linguagem de programação Python, uma vez que o Ágape foi desenvolvido nessa linguagem.

Quanto aos conceitos necessários para o entendimento do Ágape, como IoHT e o próprio *framework*, deviam ser explicados a todos os participantes como uma forma de nivelar os conhecimentos e evitar viés durante a validação, dado que os participantes poderiam ter

conhecimentos distintos quanto ao desenvolvimento de aplicações de IoHT. O mesmo não foi feito com o desenvolvimento Android e Python, pois as linguagens são requisitos mínimos para os voluntários participarem do experimento, dado que a avaliação não era sobre as tecnologias em si.

Para selecionar os participantes, era necessário coletar dados sobre o conhecimento deles acerca das tecnologias a serem utilizados durante a avaliação. Foi criado um formulário para isso, o qual continha informações pessoais, como nome, e-mail e escolaridade e informações acerca da experiência deles com as tecnologias e conceitos, como Android, Python, IoT, *frameworks* e aplicações para saúde. O Apêndice A apresenta o formulário aplicado.

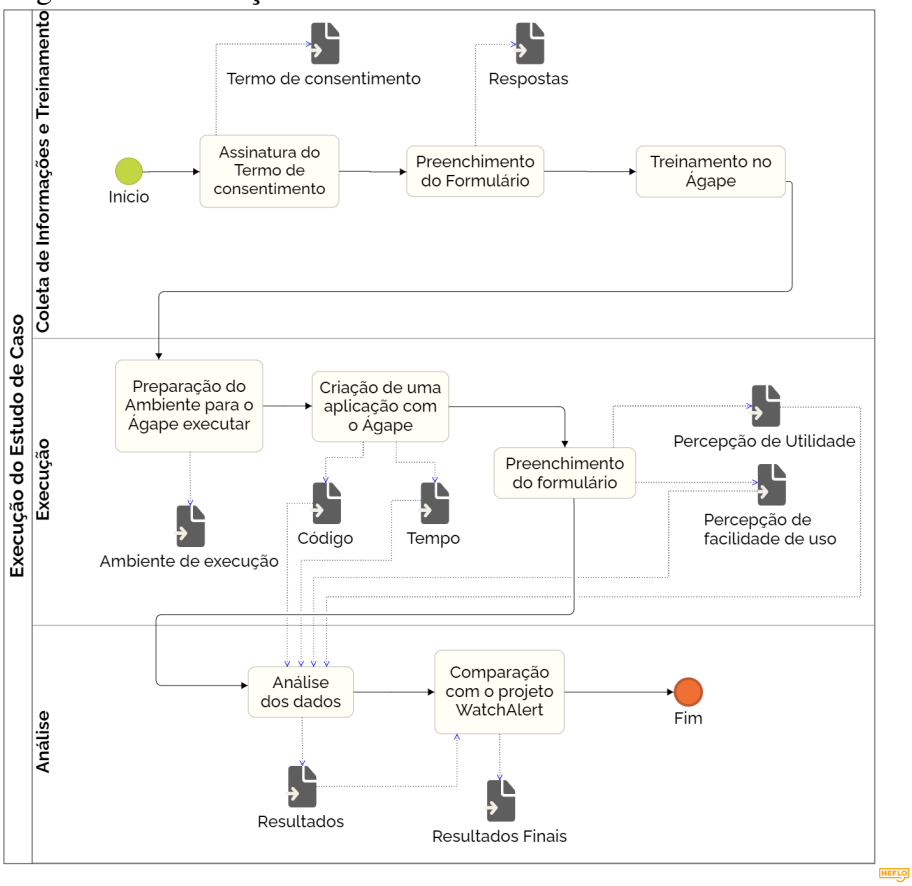
Além disso, no formulário havia o consentimento dos participantes quanto à participação na avaliação e a permissão da utilização dos dados obtidos de forma anônima para fins de pesquisa e publicação de artigos. Assim, todos que preenchiam o formulário davam o seu consentimento. Essas etapas e as próximas relacionadas à execução da avaliação são apresentadas na Figura 33.

Após o preenchimento do formulário, os participantes foram submetidos a um treinamento teórico-prático, o qual contou com a explicação do objetivo do Ágape, quais os módulos e como funcionam, a arquitetura dele, como é o padrão de dados compreendido pelo Ágape e a estrutura do código. Foi disponibilizada uma máquina virtual com o ambiente configurado para que eles pudessem participar do treinamento. Além disso, durante o treinamento, os participantes realizaram três atividades de forma a entender como funciona o Ágape, sendo as atividades: *i) criação de um novo dispositivo no Ágape; ii) definição de um novo algoritmo no Ágape; iii) execução local do Ágape.*

Em seguida, eles tiveram que configurar o ambiente de execução do Ágape, o que poderia ser feito manualmente ou utilizando um *script* para ambientes Linux. A forma manual consistia da instalação do banco de dados, RabbitMQ e as bibliotecas Python, que eram para acessar o banco de dados (*influxdb*), fazer cálculos matemáticos (e.g., *numpy, pandas*) e para Aprendizagem de Máquina (e.g.,*scikit-learn*). Na avaliação, três participantes usaram a forma manual e dois a versão com *script*.

Na sequência, após a realização do treinamento e configuração do ambiente, foram executadas as atividades referentes à avaliação propriamente dita. Assim como no treinamento, foram executadas três tarefas: *i) adicionar um novo dispositivo e um novo sensor; ii) adicionar e associar um novo algoritmo para o dispositivo e o sensor adicionados; iii) criar uma aplicação*

Figura 33 – Execução do estudo de caso



Fonte: O autor.

Android com coleta de dados de sensores do smartphone e smartwatch e caso o desenvolvedor possua os dispositivos, utilizá-los ou simular o envio de dados com dados previamente coletados. Esses últimos foram fornecidos pela equipe de avaliação para os participantes quando necessário. Diante da atividade de desenvolver uma aplicação, foi definido que os desenvolvedores poderiam realizar tal atividade em um prazo de até sete (7) dias. Em cada uma das atividades, o tempo de execução foi registrado e, ao final, os participantes preencheram um formulário de avaliação baseado no TAM e que é apresentado no Apêndice B.

Ao final, os dados inseridos no formulário foram analisados com cálculos estatísticos, os quais são apresentados na Seção 5.2. Além disso, os tempos foram comparados aos tempos obtidos durante o desenvolvimento do WatchAlert.

5.2 Resultados

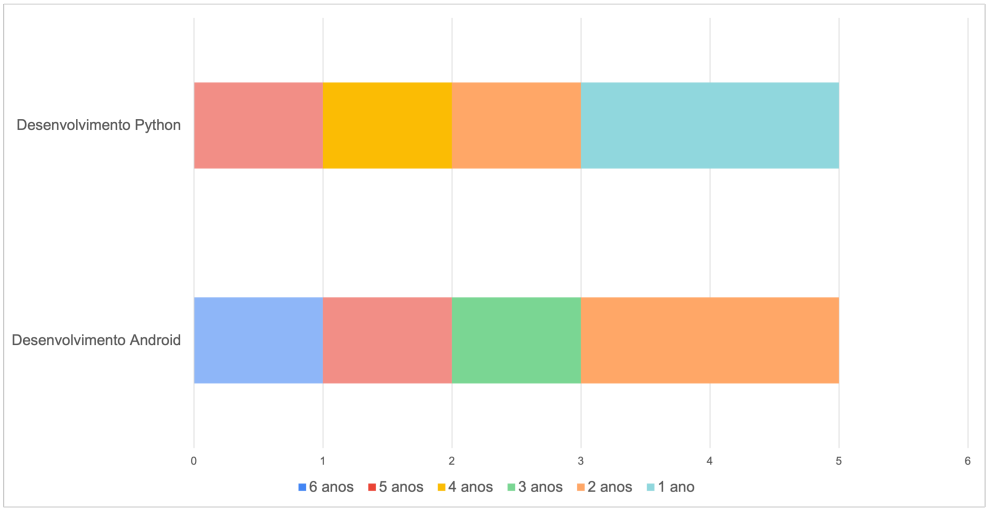
5.2.1 Perfil dos Participantes

Quanto aos participantes, sete (7) foram selecionados, mas dois (2) não finalizaram o experimento, sendo um por questões pessoais e o outro não registrou os tempos corretamente, portanto, os dados deles não são considerados para análise. Assim, ao final foram utilizadas as respostas de cinco (5) voluntários. Desses cinco voluntários, três (3) fazem doutorado, um (1) é mestre e o último era estudante de mestrado.

Sobre os conhecimentos acerca das tecnologias, os participantes responderam aos questionamentos sobre o tempo de desenvolvimento com Android e Python. Como é mostrado na Figura 34, um (1) dos participantes possuía seis (6) anos de experiência com o desenvolvimento Android, enquanto outro possuía cinco (5) anos. Dois (2) deles possuíam três (3) anos de experiência e os dois (2) últimos tinham dois (2) anos.

Quanto ao desenvolvimento com Python, um (1) participante era o que possuía a maior experiência, sendo essa de cinco (5) anos. Outro tinha um pouco menos, sendo quatro (4) anos e o terceiro, por sua vez, tinha dois (2) anos. Os dois últimos são os que tinham menos experiência, com um (1) ano.

Figura 34 – Conhecimento dos Participantes acerca das tecnologias Android e Python



Fonte: O autor.

Outra questão respondida pelos participantes diz respeito aos *framework* já utilizados por eles. As respostas foram elencadas na Tabela 9, a qual contém o nome do *framework*, bem como a quantidade de citações dadas pelos participantes. Ressalta-se que não havia um limite de

frameworks a serem informados e que os participantes responderam de forma espontânea. Ao todo, vinte (20) *frameworks* foram citados, sendo o que recebeu maior número de citações foi o Hibernate com três (3). Em seguida, apareceram três com o segundo maior número, que foi dois (2): Django, Spring MVC e Swing. Todos os demais foram citados apenas uma (1) vez.

Tabela 9 – *frameworks* citados pelos participantes

	Framework	**Citações**
1	Hibernate	3
2	Django	2
3	Spring MVC	2
4	Swing	2
5	Android Annotations	1
6	Demoiselle	1
7	EJB	1
8	Entity *Framework*	1
9	Jquery	1
10	Jsf	1
11	Maven	1
12	POI	1
13	PyQt	1
14	Ruby	1
15	SASeS	1
16	Spring	1
17	Spring Security	1
18	Succeed	1
19	SWT	1
20	VRaptor	1

Fonte: O autor.

Os participantes também responderam sobre o conhecimento que tinham sobre IoT, além de informar se já haviam desenvolvido algum aplicativo para IoT. Quanto ao conhecimento, todos afirmaram possuir, contudo quatro (4) já tinham tido a experiência de desenvolvimento. Além disso, todos asseguraram que já haviam desenvolvido algum aplicativo que utilizasse sensores, os quais foram elencados na Tabela 10.

No total, os voluntários já utilizaram quinze (15) sensores diferentes, dos quais o acelerômetro foi utilizado por quatro (4) deles, enquanto o giroscópio e o GPS foram utilizados por três (3). Outros dois (2) sensores foram elencados duas (2) vezes: luminosidade e proximidade. Os outros dez (10) sensores foram citados apenas uma vez pelos participantes da avaliação.

Quanto ao desenvolvimento de aplicações de saúde, dois deles já haviam feito aplicações, tendo um dos participantes desenvolvido duas aplicações, sendo uma de IoHT. O outro participante desenvolveu uma aplicação que também era de IoHT. Quanto aos domínios, o primeiro deles desenvolveu uma aplicação de monitoramento do usuário e de plano de saúde, enquanto o outro criou uma para detecção de quedas. Por fim, nenhum dos participantes havia

Tabela 10 – Sensores já utilizados pelos participantes

	Sensor	**Citações**
1	Acelerômetro	4
2	Giroscópio	3
3	GPS MVC	3
4	Luminosidade	2
5	Proximidade	2
6	Batimentos Cardíacos	1
7	Magnetômetro	1
8	Microfone	1
9	NFC	1
10	Peso	1
11	Presença	1
12	Pressão	1
13	Sensor de Corrente	1
14	Sensor hidráulico	1
15	Temperatura	1

Fonte: O autor.

utilizado um *framework* para auxiliar o desenvolvimento das aplicações de IoHT citadas.

5.2.2 Análise dos Tempos para Configuração e Desenvolvimento

Após o desenvolvimento dos aplicativos, os participantes enviaram os dados registrados e são apresentados na Tabela 11. Nela, estão contidas as referências aos participantes, bem como os tempos em horas, minutos e segundos. Para os tempos de configuração do Ágape, o menor valor foi treze minutos e 41 segundos, enquanto o maior tempo foi quase duas horas, sendo uma hora e cinquenta minutos. Para facilitar os cálculos, todos os tempos foram convertidos para minutos. Assim, obteve-se o tempo médio para a configuração de 61,46 minutos.

Por outro lado, os tempos para o desenvolvimento foram no mínimo iguais ou maiores. Um dos participantes fez em uma hora e dez minutos a configuração e o mesmo tempo no desenvolvimento. Exceto por um, todos os demais tiveram um tempo menor de desenvolvimento. Ao final, o menor tempo de desenvolvimento foi de quarenta minutos, que foi obtido por dois participantes e o maior foi de seis horas, doze minutos e quinze segundos. A média desses tempos foi de 114,72 minutos.

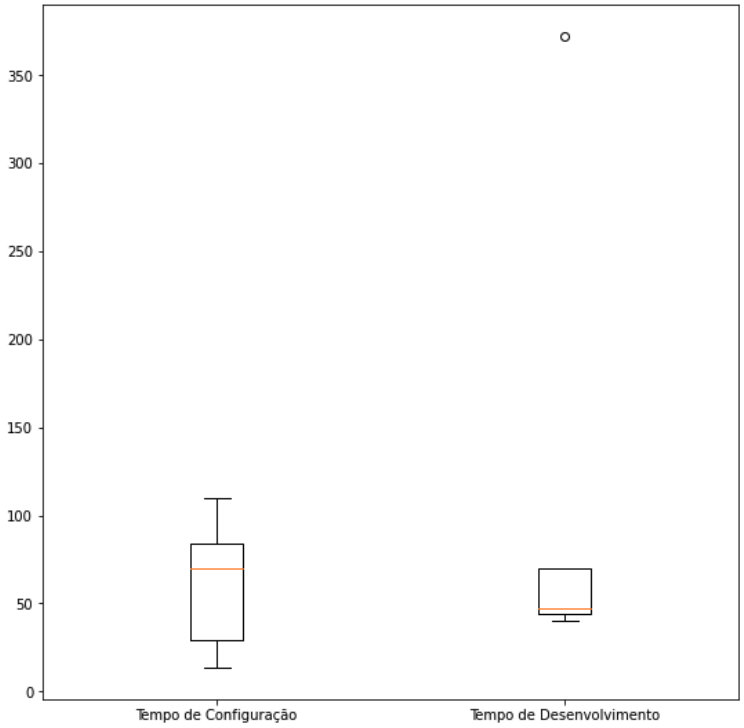
Tabela 11 – Tempos coletados durante a avaliação

Voluntário	**Tempo de configuração**	**Tempo para desenvolver usando o *framework***
V1	01:14:01	06:12:15
V2	00:29:36	00:44:00
V3	01:10:00	01:10:00
V4	01:50:00	00:40:00
V5	00:13:41	00:47:20

Fonte: O autor.

Isto posto, iniciou-se os testes estatísticos para checar se a hipótese nula foi aceita ou rejeitada. Primeiramente, buscou-se analisar os tempos em busca de *outliers*, o que foi feito ao analisar um gráfico do tipo *boxplot* (MALHOTRA, 2016). Assim, com a análise do gráfico apresentado na Figura 35, um dos participantes foi categorizado como *outlier* dado que ele se mostrou distoante dos demais no tempo de desenvolvimento. Devido à isso, os dados dele foram removidos para os dois tempos a fim de evitar problemas nas análises dos participantes.

Figura 35 – Boxplot dos tempos dos participantes



Fonte: O autor.

Além disso, foram calculados outras métricas para os dados sem *outliers*, como média, mediana, desvio padrão, quartil 25% e quartil 75%. Também foi calculada a moda, porém, como não houve um valor repetido para o tempo de configuração e para o tempo de desenvolvimento, essa métrica não foi apresentada. Todos essas métricas são apresentados na Tabela 12 e em minutos.

Todos as métricas estão com valores maior para o tempo de configuração, pois houve diferenças nas formas de configurar o ambiente, que como descrito anteriormente, pode ter sido manual ou usando um *script* para Linux. Essa pode ter sido a causa dos valores maiores e divergentes, o que pode ser demonstrado pelo desvio padrão, que mostra os dados mais dispersos do que o tempo de desenvolvimento. Ressalta-se que os dados são mais homogêneos quando estão mais próximos de 0.

Os quartis separam os dados em partes iguais, o que indica que o percentual do quartil (e.g., 25%, 75%) dos dados estão abaixo do valor encontrado. Assim, para o tempo de configuração o quartil 25% indica que 25% dos dados estão abaixo de 25,62 minutos, enquanto que para o tempo de desenvolvimento 25% dos dados estão abaixo de 43 minutos. O limite para o quartil de 75% é 80 e 53 minutos para os tempos de configuração e desenvolvimento respectivamente.

Tabela 12 – Tempos coletados durante a avaliação

Tempo (min)	Média	Mediana	Desvio Padrão	Quartil 25%	Quartil 75%
Tempo de Configuração	55,5	49,8	37,42	25,62	80
Tempo de Desenvolvimento	50,33	45,66	11,65	43	53

Fonte: O autor.

A partir de então, iniciou-se a análise dos dados visando identificar o teste estatístico adequado para o contexto analisado. Primeiramente, buscou-se identificar se os dados são normais o que foi feito com o uso do teste Shapiro-Wilk com o valor de $\alpha = 0.05$. Para que o resultado do teste indicasse normalidade o *p-value* deve ser maior do que o α. Assim, como apresentando na Tabela 13, o *p-value* para os dois tempos atende a esse critério, pois para o tempo de configuração foi 0,713 e para o outro foi 0,158.

Além disso, para identificar o teste a ser executado é necessário identificar se as variâncias das duas amostras são homogêneas, o que pode ser calculado com o teste de Levene, que foi escolhido por ser uma distribuição contínua e pelo tipo de dado, que é numérico. Assim como o de Shapiro-Wilk, assumiu-se o valor de $\alpha = 0,05$ e da mesma maneira, o valor obtido deve ser maior do que o α. Há uma diferença para o Shapiro-Wilk, pois esse resulta em um valor para cada tempo, enquanto o de Levene resulta em um único valor, pois analisa os dois dados ao mesmo tempo. O *p-value* obtido também foi apresentado na Tabela 13, sendo ele 0,111, o que é maior do que o α. Então, conclui-se que as variâncias são homogêneas.

Por fim, ao analisar a normalidade dos dados e a homogeneidade das variâncias, além do tipo de dado a ser analisado estatisticamente, chegou-se ao uso do Test-T (MALHOTRA, 2016; WOHLIN *et al.*, 2012), que é um teste paramétrico que analisa a média das amostras de dados independentes. Destaca-se que as variáveis de tempo são dependentes, porém as amostras são independentes, pois o tempo de configuração não possui associação direta com o tempo de desenvolvimento e vice-versa. Por exemplo, se o tempo de configuração for elevado, ele não implica em um tempo de desenvolvimento também elevado.

Executando o Test-T, obteve-se o *p-value* apresentado na Tabela 13, que foi de

0,225. O α para esse teste também foi $\alpha = 0,05$, de forma análoga aos demais testes, contudo, a diferença é que o o *p-value* deve ser inferior ao α para que a hipótese nula seja rejeitada. Portanto, como o valor é superior ao α, a hipótese nula não foi rejeitada, sendo aceita. Assim, os resultados indicam que os tempos necessários para configurar e para desenvolver usando o Ágape são similares.

Tabela 13 – Resultados dos testes estatísticos

	p-value		
Tempo	**Shapiro-Wilk**	**Levene**	**Test-T**
Tempo de Configuração	0,713	0,111	0,225
Tempo de Desenvolvimento	0,158		

Fonte: O autor.

5.2.3 Análise dos Tempos de Desenvolvimento

Para analisar os tempos de desenvolvimento usando o Ágape e sem ele em uma aplicação real, foi utilizado o WatchAlert como base observando os *commits* do desenvolvedor responsável a fim de identificar o tempo médio gasto para o desenvolvimento da aplicação considerando o ambiente de execução dos algoritmos de Aprendizagem de Máquina utilizados no WatchAlert.

Para chegar a esse tempo, foi utilizado o tempo médio de 4h/dia, que era a carga horária do desenvolvedor responsável pelo WatchAlert. Quanto aos *commits*, foram encontrados onze (11) relacionados à parte descrita anteriormente. Assim, obteve-se o esforço para a criação do WatchAlert com o servidor e o tempo total de 44 horas, o que convertido resulta em 2640 minutos.

O tempo médio de desenvolvimento usando o Ágape conforme identificado na Tabela 12 foi de 50,33 minutos, o que significa uma redução de 98,09% em comparação com o tempo de desenvolvimento do WatchAlert. Assim, os resultados indicam uma redução do tempo de desenvolvimento de soluções de detecção de quedas e suas causas.

5.2.4 Análise da Correlação das Variáveis

Como descrito na Seção 5.1, o Ágape foi avaliado seguindo o TAM e buscando identificar se o *framework* é fácil de usar e é útil. Além disso, após a avaliação, os participantes responderam ao questionário apresentado no Apêndice B. Quanto à facilidade de uso, as respostas das nove perguntas são apresentadas nas Figuras 36 e 37 e as de utilidade são visualizadas na

Figura 38.

Antes de tratar das assertivas, é importante destacar que a numeração delas corresponde à numeração apresentada na Seção 5.1. Quanto à facilidade de uso, a assertiva A1 trata analisa se houve confusão durante o uso do Ágape, o que foi confirmado por três participantes ao responderem "Concordo", enquanto dois (2) mantiveram-se neutro à esse item. A A2 afirma sobre os usuários terem cometido erros ao usar o Ágape, o que teve a concordância dos cinco participantes.

Em contrapartida, ao falarem sobre a frustração quanto à utilização do Ágape, quatro (4) dos voluntários asseguraram que não se sentiram assim, enquanto um (1) se mostrou neutro. O que sugere que, mesmo eles tendo cometido algum erro, esses erros não causaram frustração nos participantes. Por outro lado, a frustração pode ter sido diminuída pelo fato de terem consultado o tutorial, ação contida na A4 e que foi realizada por todos.

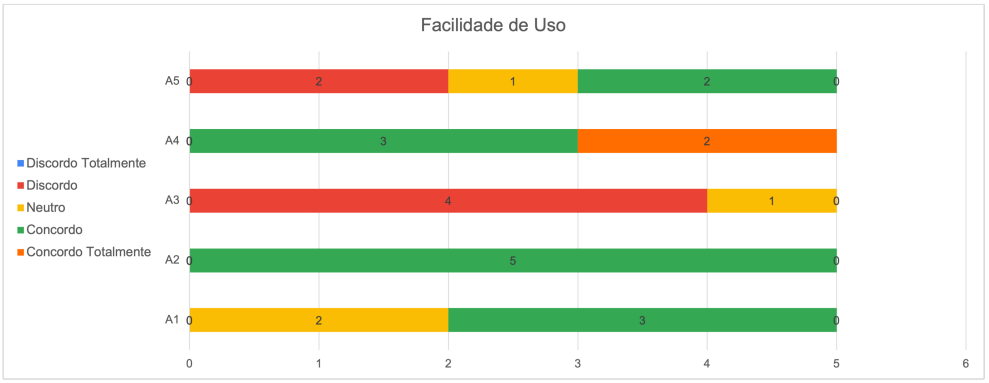
A A5, por sua vez, lida com a facilidade de se recuperar de um erro. Dois (2) participantes concordaram que foi fácil, dois (2) afirmaram que discordam dessa afirmativa, enquanto um (1) mostrou-se neutro. A próxima lida com a facilidade de dizer ao Ágape o que deveria ser feito, tendo dois (2) participantes concordado enquanto os outros três (3) mostraram neutralidade.

Quanto à A7, três (3) discordaram, um (1) discordou totalmente e outro mostrou-se neutro no que se refere à comportamentos inesperados do Ágape. A A8 tratava do fato dos participantes poderem sentir um incômodo ao usar o Ágape, o que foi refutado por quatro (4), sendo dois (2) com discordo e outros dois (2) com discordo totalmente. O último colocou neutro. A última assertiva tratava sobre a facilidade de relembrar como executar uma operação no Ágape, o que todos afirmaram concordar.

Em termos percentuais, a assertiva A1 obteve concordância de 60% dos participantes e 40% se mostrou neutro. Para a A2, houve concordância de 100%, enquanto que para a A3 houve discordância de 80% e 20% de neutralidade. Para a A4, houve concordância de 100% dos participantes e a A5 obteve 40% de concordância e discordância e 20% de neutralidade. Para a A6, foi 40% de concordância e 60% para neutro, a A7 obteve 80% de discordância e 20% de neutros. A A8 teve percentuais iguais ao da A7 e a A9 obteve 100% de concordância.

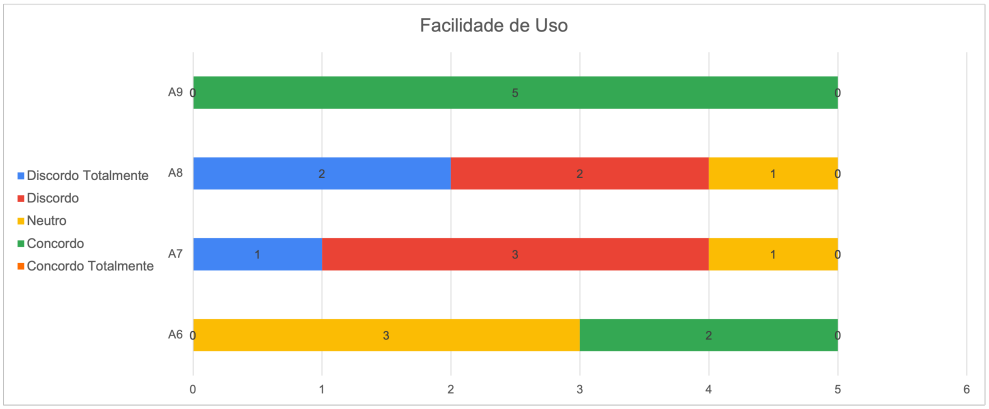
Quanto à utilidade, a primeira assertiva (A1) foi avaliada pelos participantes com um (1) "Concordo Totalmente" e quatro (4) "Concordo". Essa assertiva corresponde à "O uso do *framework* facilita o desenvolvimento de um software IoHT.". Da mesma maneira, os

Figura 36 – Respostas dos participantes para a facilidade de uso - questões 1 à 5



Fonte: O autor.

Figura 37 – Respostas dos participantes para a facilidade de uso - questões 6 à 9



Fonte: O autor.

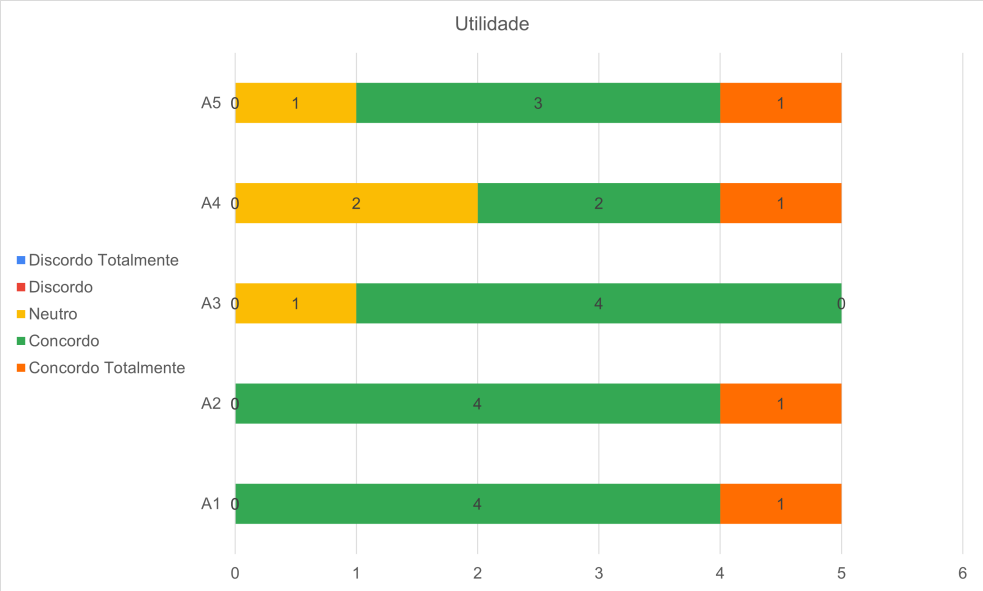
participantes responderam à A2, que trata da economia de tempo no desenvolvimento ao utilizar o Ágape.

A A3 obteve concordância similar às duas primeiras, mudando a quantidade de resposta de “Concordo Totalmente” para “Neutro” e mantendo quatro (4) em “Concordo”. Tal assertiva corresponde ao aumento da qualidade do trabalho realizado com o uso do Ágape. A A4, por sua vez, que trata da contribuição do Ágape para ajudar a processar os dados e a detectar uma queda, teve duas (2) respostas em “Neutro”, duas (2) em “Concordo” e uma (1) em “Concordo Totalmente”. Por fim, a A5, que avalia se o Ágape é útil no desenvolvimento de software, três (3) responderam que concordavam, um (1) que concordava totalmente e o último foi neutro.

Diante desses dados, percebe-se que a A1 e a A2 tiveram 100% de concordância dos usuários, enquanto a A3 e a A5 tiveram 80% de concordância e a A4 60%. Ainda quanto a A3 e a A5, houve 20% de neutralidade dos participantes, enquanto que para a A4 essa taxa chegou a 80%.

Para fazer uma análise de respostas ordinais, como é o caso das respostas desta

Figura 38 – Respostas dos participantes para a utilidade



Fonte: O autor.

avaliação que usa a escala Likert, é importante realizar um teste de confiabilidade para verificar a confiança das respostas, o que pode ser feito a partir do cálculo do α de Cronbach. Essa validação da confiabilidade também foi feita em outros trabalhos, como (DAVIS, 1989; ALANAZI; SOH, 2019). Para chegar ao resultado da confiabilidade, foram analisadas as assertivas considerando apenas a i) facilidade de uso, ii) apenas a utilidade e iii) as duas juntas.

Destaca-se que os cálculos foram feitos com o apoio da ferramenta SPSS Statistics. Nos primeiros testes, a variância para as assertivas A2 e A9 da facilidade de uso foram identificadas como zero (0), o que zeradas, foram removidas durante o cálculo do alpha de Cronbach. O valor obtido foi -0,972, o que é considerado "pobre", conforme a escala encontrada em (LANDIS; KOCH, 1977). Em (GLIEM; GLIEM, 2003), é descrito que é possível remover assertivas a fim de analisar a confiabilidade buscando aumentá-la.

Visando descobrir quais as assertivas que mais contribuem para a redução da confiabilidade, foi feito o cálculo com a exclusão de itens, com a A2 e a A9 já removidas, e o resultado sugeriu que as assertivas A1, A4 e A7 da facilidade de uso poderiam ser removidas. Feito isso, a confiabilidade da facilidade de uso subiu para 0,780, o que é considerado um valor "substancial", ainda conforme a escala definida em (LANDIS; KOCH, 1977). Isso também impactou no valor da confiabilidade geral, que anteriormente era 0,511, valor considerado "moderado", e subiu para 0,892, valor considerado "quase perfeito". Para a utilidade, o valor inicial com as cinco (5) assertivas foi 0,841 e não precisou de nenhum ajuste. Todos os dados são demonstrados na Tabela 14. Ao final, as assertivas restantes foram:

– (3) Eu me senti frustrado frequentemente.
– (5) Eu achei fácil recuperar de erros que aconteceram no *framework*.
– (6) Eu achei fácil fazer o *framework* executar o que eu queria.
– (8) Eu achei desconfortável utilizar o *framework*.

Tabela 14 – Confiabilidade medida pelo alpha de Cronbach

Variável	**Com todas as afirmativas**	**Com a remoção de afirmativas**
Facilidade de Uso	-0,972	0,780
Utilidade	0,841	-
Geral	0,511	0,892

Fonte: O autor.

Além disso, outros testes foram realizados visando verificar a validade da consistência dos dados, como a análise fatorial (FA, do inglês *Factor Analysis*) e a Análise Fatorial Confirmatória (CFA, do inglês *Confirmatory Factor Analysis*). O primeiro teste é utilizado para reduzir as dimensões dos dados facilitando a análise deles. Contudo, é preciso fazer o teste de esfericidade de Bartlet, o que indica se a análise fatorial é adequada para o problema, pois analisa os dados e verifica se eles correspondem a uma matriz identidade. Para isso, o *p-value* deve ser menor do que 0,05 e o valor obtido foi de 0, portanto os dados podem ser submetidos à analise fatorial.

Visto que os dados podem ser submetidos à análise fatorial, ela foi aplicada para as assertivas restantes, que foram A3, A5, A6 e A8 da facilidade de uso e todas de utilidade. Ao final, o resultado mostrou que os dados estavam relacionados a apenas um fator, que é uma variável que relaciona as variáveis observadas. Essa informação a ser utilizada na CFA. A CFA mede a confiança das afirmativas e depende da quantidade de fatores identificados anteriormente. Os dados são confiáveis se o resultado do CFA for maior do que 0,70, conforme encontrado em (BRAR *et al.*, 2022; ALANAZI; SOH, 2019), e foi obtido o valor 1 para todas as afirmativas. Assim, todos os dados são confiáveis.

Em seguida, foi feita a correlação de Pearson para analisar a correlação entre a percepção de facilidade de uso e a percepção de utilidade. Essa medição também foi feita com o suporte da ferramenta SPSS Statistics, a qual identificou que elas possuem correlação de 1. Após identificada a correlação, foi executada a regressão linear para identificar se a facilidade de uso tem um efeito direto na utilidade e vice-versa.

Para que a regressão linear identifique o efeito direto de uma variável em outra, o *p-value* deve ser menor do que o α, cujo valor é 0,05. Assim, o *p-value* que verificou o efeito

da facilidade de uso na utilidade foi 0,022, que é menor do que o α. Entretanto, ao analisar o *p-value* do efeito inverso, foi obtido 0,799. Portanto, há um efeito direto da facilidade de uso na utilidade, mas o inverso não é verdadeiro.

Tabela 15 – Correlação de Pearson para a facilidade de uso e utilidade

Variável	**Facilidade de Uso**	**Utilidade**
Facilidade de Uso	-	1
Utilidade	1	-

Fonte: O autor.

Tabela 16 – Regressão linear para a facilidade de uso e utilidade

Variável	**Facilidade de Uso**	**Utilidade**
Facilidade de Uso	-	0,022
Utilidade	0,799	-

Fonte: O autor.

5.3 Discussão

Analisando os tempos obtidos para configurar o Ágape e o tempo para desenvolver uma aplicação utilizando o mesmo e os testes estatísticos realizados, a hipótese nula não foi rejeitada como esperava-se, o que indica que os dois tempos são iguais estatisticamente. Esse ponto pode ser explicado por comentários deixados pelos participantes da avaliação na última questão do formulário pós-avaliação. Dentre os comentários, destacam-se problemas para instalar as dependências do Ágape, como o RabbitMQ e o InfluxDB, o que poderia ser facilitado, pois para a instalação do primeiro também foi preciso instalar outras dependências, requisitando mais esforço e tempo. Destaca-se que esses pontos foram elencados pelos participantes que não utilizaram o *script* criado para ambientes Linux.

Além disso, ainda sobre o tempo de desenvolvimento, os participantes tiveram a percepção de que o tempo para construir uma nova solução usando o Ágape foi reduzido, o que ficou demonstrado pela afirmativa A2 da variável utilidade e pela comparação de tempo com o desenvolvimento do WatchAlert. Ademais, também foi percebido que ele facilitou o desenvolvimento pelas assertivas A1 da utilidade e A6 da facilidade de uso e pelo comentário de um participante que citou a facilidade para inserir novos dispositivos, sensores e algoritmos. Entretanto, esse mesmo voluntário citou uma possível melhoria para o Ágape quanto ao mapeamento dos novos dispositivos e sensores para todos os módulos, otimizando tal atividade a fim de evitar o mapeamento manual em cada um dos módulos.

Observa-se também que os participantes não demonstraram frustração quanto ao uso do *framework*, o que corrobora a percepção deles de que o Ágape facilitou o desenvolvimento de uma nova solução. Muito embora não tenham se sentido frustrados, 60% dos participantes não concordaram com a facilidade para fazer com que o Ágape fizesse o que desejavam. Esse ponto pode ser justificado pelos comentários adicionais, os quais sugeriam melhorar o manual ou o *readme* do Ágape, indicando todos os passos para adicionar um novo dispositivo e sensor, por exemplo. Outro sugeriu melhorar os comentários no código de forma a facilitar o entendimento pelo próprio código, enquanto um sugeriu refatorar algumas partes a fim de tornar o código um pouco menos complexo.

Apesar dessas sugestões de melhoria quanto ao uso do Ágape, 80% dos participantes não acharam que o uso dele fosse incômodo, enquanto 20% se mostrou neutro. Esse ponto pode ser confirmado pela percepção da melhoria da qualidade do trabalho desempenhado pelos desenvolvedores, além deles terem percebido que o *framework* realmente ajuda no processamento dos dados e na detecção de quedas, além de se mostrar útil para o desenvolvimento de software. Ainda nesse contexto, um dos participantes comentou que o uso do *framework* não é tão complexo, tendo o participante gasto mais tempo no desenvolvimento da aplicação Android.

Diante dos resultados obtidos, percebe-se que o Ágape tem o potencial de ser aplicado para desenvolver outras soluções de detecção de quedas e suas causas. Além dos resultados, isso foi demonstrado pelo comentário de um dos participantes, o qual ressaltava a importância do Ágape, além de descrever que após algum tempo de uso é fácil utilizá-lo, além de possuir o potencial de causar um grande impacto na sociedade no futuro.

5.4 Ameaças à Validade

Nos estudos empíricos podem haver ameaças à validades, a quais podem ser classificadas em quatro categorias, segundo (WOHLIN *et al.*, 2012): i) Validade de conclusão; ii) Validade interna; Validade de constructo; e iv) Validade externa.

A validade de conclusão foca na relação estabelecida entre o tratamento e os resultados obtidos (WOHLIN *et al.*, 2012). No caso deste trabalho, o tratamento utilizado foi o Ágape e os resultados foram as medidas de tempo e dos formulários de avaliação. Fatores que podem afetar a conclusão, de acordo com (WOHLIN *et al.*, 2012), são ruídos externos durante a avaliação, heterogeneidade dos participantes e confiabilidade das medidas. Quanto aos ruídos externos, os participantes só realizaram a avaliação quando estivessem disponíveis e em

condições adequadas para eles a fim de evitar intervenções externas. Outro aspecto refere-se à quantidade de participantes, que embora tenham sido poucos, já sugerem que há indícios de que o Ágape cumpre o papel para o qual criado.

Em relação à heterogeneidade dos participantes, essa ameaça foi mitigada com a definição dos requisitos para seleção dos participantes quanto ao conhecimento que deveria estar presente. Assim, foi definido como sendo o conhecimento em Python e Android, com experiência no uso de sensores e todos os participantes atendiam a esse critério. Quanto à confiabilidade das medidas, o tempo é uma medida que não depende do julgamento dos participantes, enquanto as outras medidas foram baseadas no TAM, que é um modelo conhecido para aceitação de tecnologias, e foram avaliadas com a escala Likert. Além disso, as assertivas formuladas foram submetidas à especialistas a fim de validarem as mesmas e evitar que houvesse algum viés. Quanto ao tempo do WatchAlert, foi considerado o tempo mínimo utilizado para o desenvolvimento de uma nova solução.

A validade interna trata da identificação de causa e efeito na relação entre o tratamento e os resultados obtidos (WOHLIN *et al.*, 2012). Uma das ameaças é em relação ao uso de um único grupo para a realização da avaliação, que pode ter impacto dos participantes selecionados, mas como descrito anteriormente, foram selecionados aqueles que atendiam aos requisitos mínimos para participar. Além disso, os participantes não foram selecionados a partir de um filtro de um grupo maior, mas de voluntários que se colocaram à disposição para participar e nesse grupo de voluntários também estavam os dois que não finalizaram a avaliação. Quanto à execução, buscou-se deixar os participantes a vontade para que possam desenvolver a aplicação no tempo deles a fim de evitar um desgaste ou desinteresse.

A validade de constructo analisa a relação de causa e efeito entre o tratamento e os resultados, de forma que nessa relação o tratamento seja a causa e os resultados o efeito (WOHLIN *et al.*, 2012), não podendo ser invertido. Ela está relacionada a problemas no projeto da avaliação ou aspectos sociais. No tocante a problemas no projeto, tais problemas podem surgir, por exemplo, ao conter termos mal utilizados ou um único tratamento e uma variável. Quanto aos termos utilizados na avaliação do Ágape, ressalta-se que eles foram revisados por especialistas e quanto aos tratamentos e variáveis, foi um tratamento, mas foram medidas quatro variáveis.

Quanto aos aspectos sociais que ameaçam a validade de constructo, há a possibilidade dos participantes adivinharem as hipóteses, o que poderia mudar a atitude do participante

positivamente ou negativamente. Contudo, os participantes só precisaram coletar o tempo durante a execução das atividades, o que pode ser medido, sem imaginar qual a relação a ser feita entre eles. O formulário foi preenchido apenas após a avaliação, o que não permitiria gerar um impacto positivo ou negativo nos voluntários. Outro aspecto social que poderia impactar é o caso de algumas pessoas que sentem medo de serem avaliadas. Para mitigar essa ameaça, os participantes foram informados que o objeto de avaliação é o Ágape e não eles.

O foco da validade externa é a generalização dos resultados (WOHLIN *et al.*, 2012). Os autores citam que um dos problemas pode ser a seleção de participantes que não correspondam ao público-alvo da solução avaliada, porém, durante o processo de seleção dos participantes, já foram escolhidos conforme os requisitos mínimos, que corresponde ao público-alvo da solução. O outro ponto de ameaça à avaliação deste trabalho é a interação do histórico com o tratamento, sendo o histórico algum dia ou momento inoportuno para o participante, o que, como já descrito anteriormente, foi feito apenas quando eles se mostravam disponíveis.

5.5 Conclusão

Neste Capítulo foi apresentado o estudo empírico que foi realizado em um ambiente sintético, pois simulava o ambiente real devido à limitações de dispositivos reais a serem utilizados e a pandemia da novo coronavírus. A descrição do estudo contemplou desde o planejamento até os resultados. Quanto ao planejamento, a avaliação deste trabalho buscou medir o tempo necessário para configurar o Ágape e o tempo para desenvolver uma nova solução usando o mesmo. Esses critérios podem impactar na aceitação do mesmo pelos desenvolvedores.

Quanto à comparação entre os tempos, os testes mostraram que os tempos eram similares. Além disso, o tempo de desenvolvimento foi comparado com outra solução já desenvolvida a fim de identificar se o Ágape realmente já conseguia reduzir o tempo de desenvolvimento. Quanto a essa resposta, os resultados foram promissores, indicando que havia sim uma redução no tempo que foi de 98,09%.

Também foi feita uma análise baseada no *Technology Acceptance Model* para verificar se os desenvolvedores participantes da avaliação percebiam que o Ágape possuía uma facilidade de uso e se ele útil. Nessa avaliação é importante analisar a correlação entre essas variáveis para verificar como elas se comportam, mas para isso foi preciso primeiro entender a confiabilidade dos dados obtidos. Os testes indicaram que os dados eram confiáveis e que poderia ser feita a correlação entre as variáveis, além de interpretar os resultados obtidos. Por

fim, foram apresentadas as principais ameaças à validade do estudo empírico.

6 CONCLUSÃO

Este trabalho propôs um *framework*, o Ágape, com o objetivo de auxiliar os desenvolvedores na construção de novas soluções para detecção de quedas e suas causas.

Este capítulo conclui esta tese e é dividido como a seguir. Na Seção 6.1 é feita uma visão geral desta tese; a Seção 6.2 sumariza os principais resultados relacionados a esta tese; a Seção 6.3 revisita a hipótese e compara com os trabalhos relacionados a esta tese; na Seção 6.4 são descritas as limitações deste trabalho; e na Seção 6.5 são apresentados os trabalhos futuros.

6.1 Visão Geral

A Internet das Coisas permite a comunicação entre objetos do dia-a-dia (GUBBI *et al.*, 2013) e com isso serviços mais robustos são providos aos usuários. Essa robustez decorre do fato dos objetos coletarem dados dos usuários e do ambiente e compartilharem entre si ou enviarem para uma nuvem a fim de processar tais dados. Assim, a IoT pode aprimorar os serviços providos, como a detecção de quedas e suas possíveis causas. Isso ocorre, pois é possível monitorar o usuário e identificar as ações executadas por ele.

A queda é um evento, no qual uma pessoa vai de encontro ao solo ou a um local mais baixo do que se encontre, consciente ou inconscientemente, com lesão ou não (FREITAS *et al.*, 2016). Além disso, uma queda pode ter causas intrínsecas, como problemas de saúde, que também pode ser monitorados por diferentes dispositivos vestíveis que medem, por exemplo, os batimentos cardíacos ou a glicemia. O uso desses diferentes dispositivos traz complexidade para lidar com os dados oriundos deles, pois cada um pode apresentar um formato distinto. Além de ser necessário o uso de algoritmos específicos para a detecção de quedas, o que requer uma integração com a detecção das possíveis causas.

Embora existam soluções para detecção de quedas, como Almeida *et al.* (2016) e Hsieh *et al.* (2014), soluções prontas, que não podem ser reutilizadas, nem aprimoradas de forma fácil e limitando-se à detecção de quedas. Diante disso, como descrito no Capítulo 3, buscou-se na literatura e em soluções da indústria por iniciativas que visassem o reuso, mais especificamente *frameworks*, dado que podem agregar comportamentos comuns de aplicações já existentes de detecção de quedas, além de permitir a inserção da análise das possíveis causas. Os *frameworks* deveriam ter algoritmos para detectar quedas e integrá-los com os dados coletados por outros sensores a fim de detectar as possíveis causas. Considerando esse contexto, foi

encontrado apenas um (1) *framework* conceitual para detecção e prevenção de quedas. Ademais, não havia a associação com as causas.

Nesse cenário e parar suprir a carência identificada, foi proposto o Ágape, como explanado no Capítulo 4. Ele é um *framework* composto por cinco módulos, capazes de i) receber dados de diversos sensores, ii) processar os dados oriundos de sensores para identificar quedas, iii) converter os dados em episódios significativos, iv) agregar dados de sensores que são similares e v) analisar os dados de diferentes sensores a fim de identificar as possíveis causas.

Para lidar com dados de diferentes sensores, foi definido um modelo de dados orientado a objeto que fornece suporte a eles, bem como faz a agregação dos dados de sensores que analisam características similares, além de permitir a análise de dados de sensores distintos para identificar as causas.

Adicionalmente às propostas do *framework* e do modelo de dados, o Ágape foi instanciado na linguagem de programação Python, seguindo as práticas de orientação a objeto. Para lidar com os dados que obedecem sequências temporais, foi utilizado um banco de dados que atenda a esse critério, o InfluxDB.

Uma avaliação do Ágape foi realizada e reportada no Capítulo 5. Tal avaliação consistiu da análise de tempos necessários para configurar e desenvolver usando o Ágape, além de compará-lo com o tempo gasto para o desenvolvimento de uma aplicação real, sendo nesse caso o WatchAlert. Além disso, foi feita uma análise qualitativa baseada no TAM da facilidade de uso e utilidade percebidas pelos desenvolvedores, a qual verificou se as duas possuem uma correlação e indicam se o Ágape é fácil de usar e é útil.

Os resultados sugerem que o Ágape consegue reduzir em 98,09% o tempo de desenvolvimento de uma solução para detecção de quedas e suas causas. Além disso, esse tempo se mostrou igual ao tempo para configurar o Ágape. Além disso, a avaliação baseada no TAM indica que há correlação entre a utilidade e a facilidade de uso, além de indicar, a partir dessa relação, que ele é fácil de usar e útil.

6.2 Principais Resultados

Os principais resultados desta tese e que são detalhados no Capítulo 4 e avaliados no Capítulo 5 são descritos a seguir:

– ***Framework* Ágape:** é o *framework* proposto para detectar quedas e associar com as possíveis causas a partir de dados coletados por dispositivos vestíveis. Ele é composto de

cinco módulos, em que é feita a conversão dos dados brutos em episódios, que são eventos temporais com significados semânticos. Esses episódios são analisados a fim de detectar a queda e as possíveis causas. Essa associação das quedas e causas é feita com os dados de sensores fisiológicos sendo associados temporalmente com os dados de queda, a fim de identificar as possíveis causas, caso já houvesse uma situação que pudesse ser de alerta para o usuário.

- **Modelos de dados:** foi especificado um modelo de dados orientado a objeto para contribuir com o *framework* a fim de gerir os dados oriundos dos sensores e os episódios gerados, contribuindo para a tomada de decisão acerca das quedas e suas possíveis causas.

Além disso, durante o desenvolvimento da tese, foram obtidos resultados secundários, mas que contribuíram para a construção dessa tese. Eles são descritos a seguir:

- **WatchAlert:** desenvolvimento do aplicativo WatchAlert, que detecta quedas utilizando dados oriundos de um smartwatch.
- **Algoritmos:** foram desenvolvidos algoritmos de *threshold* e modelos de Aprendizagem de Máquina para os algoritmos *Random Forest* e SVM.
- ***Dataset* de quedas:** durante o desenvolvimento dos algoritmos, foi preciso coletar dados para validá-los. Assim, foi gerado um *dataset* com dados de acelerômetro e giroscópio coletados por um smartwatch. Dados disponíveis em https://bit.ly/31wNiiQ. Também é possível adicionar mais dados à base de dados.

Durante o doutorado, também foram publicados 17 artigos e capítulos de livro, sendo dez (10) relacionados à tese e a área de saúde e sete (7) de outros temas que contribuíram para o amadurecimento do aluno em pesquisa. Esses artigos estão elencados nas Tabelas 17 e 18, respectivamente.

Ademais, durante o desenvolvimento desta tese, o autor participou de projetos de pesquisa, desenvolvimento e inovação relacionados ao tema de saúde e quedas, os quais são descritos na Tabela 19. Durante os projetos *Cloud AI Dev CE* e *Health Risk Detection*, foram desenvolvidas duas aplicações para detecção de quedas usando dados de *smartwatches*, sendo uma das aplicações para o paciente monitorado e a outra para o cuidador do paciente. Ao final, elas foram avaliadas em um ambiente real de pacientes, o Centro de Hematologia e Hemoterapia do Ceara (HEMOCE).

Além disso, houve a coorientação do trabalho de conclusão de curso de um aluno de graduação da Engenharia da Computação, no qual foi desenvolvida uma aplicação para

Tabela 17 – Publicações durante o doutorado

Título	Autores	Conferência / Journal	h-index	Ano
Gerontecnologias e internet das coisas para prevenção de quedas em idosos: revisão integrativa (DINIZ *et al.*, 2022)	Jamylle Diniz, Viviane Sousa, Janaína Coutinho, **Ítalo Linhares de Araújo**, Rossana M. C. Andrade, Joyce Costa, Rachel Barbosa, Marilia Marques	Revista ACTA Paulista de Enfermagem	21	2022
Refactoring Decision based on Measurements for IoHT (OLIVEIRA *et al.*, 2021)	Breno Oliveira, **Ítalo Linhares de Araújo**, Joseane Paiva, Evilasio Junior, Rossana M. C. ANdrade	SBSI	9	2021
Lessons Learned from the Development of Mobile Applications for Fall Detection (LINHARES *et al.*, 2020)	**Ítalo Linhares de Araújo**, Rossana M. C. Andrade, Evilasio Junior, Pedro Oliveira, Breno Oliveira, Paulo Aguilar	Global Health	-	2020
Dorsal: Ferramenta para Geração de Modelos de Dados para Aplicações voltadas a Saúde e Cuidado de Idosos (OLIVEIRA *et al.*, 2020)	Pedro Oliveira, **Ítalo Linhares de Araújo**, Evilasio Junior, Paulo Duarte, Ismayle Santos, Rossana M. C. Andrade, Ivana Barreto, Odorico Andrade	Workshop de Ferramentas e Aplicações do SBCAS	-	2020
Towards a Taxonomy for the Development of Older Adults Healthcare Applications (ARAÚJO *et al.*, 2020)	**Ítalo Linhares de Araújo**, Evilásio Júnior, Paulo Duarte, Ismayle Santos, Pedro Almir, Cícero Mendes, Rossana Andrade	HICSS (Conferência)	52	2020
An Algorithm for Fall Detection using Data from SmartWatch (ARAÚJO *et al.*, 2018)	**Ítalo Linhares de Araújo**, Lucas Dourado, Letícia Fernandes, Rossana Andrade, Paulo Armando Cavalcanete	IEEE SoSE (Conferência)	19	2018
Artefatos de Reuso para Internet das Coisas: Arquiteturas, Middlewares, Frameworks e Padrões (ARAÚJO, 2018)	**Ítalo Linhares de Araújo**	Relatório Técnico UFC	-	2018
Raspcare: A Telemedicine Platform for the Treatment and Monitoring of Patients with Chronic Diseases (ANDREAO *et al.*, 2018)	Rodrigo Varejão Andreão, Matheus Athayde, Jérôme Boudy, Paulo Aguilar, **Ítalo Linhares de Araújo**, Rossana M. C. Andrade	IntechOpen (Capítulo de Livro)	-	2018
Introdução ao Android Wear: desenvolvendo aplicações para smartwatch (MACEDO *et al.*, 2017)	Alysson Macedo, **Ítalo Linhares de Araújo**, Rossana Andrade	ERIPI (Capítulo de Livro)	-	2017
WatchAlert: Uma evolução do aplicativo fAlert para detecção de quedas em smartwatches (ALMEIDA *et al.*, 2016)	Rodrigo Almeida, Alysson Macedo, **Ítalo Linhares de Araújo**, Paulo Aguilar, Rossana Andrade	WFA - Webmedia (Conferência)	-	2016

Fonte: O autor.

monitoramento de idosos em um Ambient Assisted Living (AAL), que consiste de um ambiente de monitoramento na residência do idoso. O trabalho intitulado "*Mona: Solução com Internet das Coisas para detecção de atividades de idosos em um Ambient Assisted Living*" (BARROSO, 2022), consistia de uma aplicação que monitorava os passos através de uma bengala inteligente, o sono, as idas ao banheiro e a temperatura e o umidade da residência do idoso a fim de detectar situações desconfortáveis a ele.

Tabela 18 – Outras publicações durante o doutorado

Título	Autores	Conferência / Journal	h-index	Ano
Continuous Integration for Machine Learning Experiments Reproducibility: a Practical Study (ANDRADE *et al.*, 2021)	André Andrade, Marciel Pereira, Samuel Silveira, Francisco Linhares, Antônio Neto, Rossana M. C. Andrade, **Ítalo Linhares de Araújo**	ISE	-	2021
GreaTest: A Card Game to Motivate the Software Testing Learning (BEPPE *et al.*, 2018)	Thiago Beppe, **Ítalo Linhares de Araújo**, Bruno Sabóia, Ismayle Santos, Rossana Andrade	SBES - Trilha da Educação (Conferência)	12	2018
Towards Developing a Practical Tool to Assist UX Evaluation in the IoT Scenario (ALMEIDA *et al.*, 2018)	Rodrigo Almeida, Ticianne Darin, Rossana Andrade, **Ítalo Linhares de Araújo**	WFA - Webmedia (Conferência)	-	2018
Generating Test Cases and Procedures from Use Cases in Dynamic Software Product Lines (ARAÚJO *et al.*, 2017)	**Ítalo Linhares de Araújo**, Ismayle de Sousa Santos, João Bosco Ferreira, Rossana M. C. Andrade, Pedro Santos Neto	SAC (Conferência)	29	2017
What changes from Ubiquitous Computing to Internet of Things in Interaction Evaluation? (ANDRADE *et al.*, 2017a)	Rossana Andrade, Rainara Carvalho, **Ítalo Linhares de Araújo**, Káthia Marçal, Márcio Maia	HCI	31	2017
Retrospective for the Last 10 years of Teaching Software Engineering in UFC's Computer Department (ANDRADE *et al.*, 2017b)	Rossana Andrade, Ismayle Santos, **Ítalo Linhares de Araújo**, Bruno Sabóia, Fernanda Siewerdt	Fórum de Educação - SBES	-	2017
Uma Metodologia para o Ensino Teórico e Prático da Engenharia de Software (ANDRADE *et al.*, 2015)	Rossana Andrade, Ismayle Santos, **Ítalo Linhares de Araújo**, Rainara Carvalho	Fórum de Educação - SBES (Conferência)	-	2015

Fonte: O autor.

Tabela 19 – Lista de Projetos relacionados à área de saúde que o autor participou

Título	**Ano**	**Tipo**
Health Risk Detection: Smart Health Risk Monitoring and Detection System Solution for Ambient Assisted Living (AAL) using Internet of Things Devices	2021/2022	Lei de Informática/Pesquisa, Desenvolvimento e Inovação
Cloud AI Dev CE	2021	Lei de Informática/Pesquisa, Desenvolvimento e Inovação
Angel: IoT e-Health Platform to Monitor and Improve Quality of Life	2021/2022	CAPES/Stic-Amsud
Rise e-Well: Remote Intelligent Systems for Elderly and Chronic Patient Follow-up	2020/2021	CAPES/Stic-Amsud
Inteligência de Governança para Tomada de Decisão na Gestão de Sistemas de Saúde com Enfoque no Cuidado em Saúde do Idoso	2019-2020	Programa Fiocruz de Fomento a Inovação
GREat IoT - Internet das Coisas aplicada à Saúde	2019 - 2022	CNPq
eMONITOR: Chronic DIsease - Ambient Assisted Living and vital teleMONITORing for e-health	2017 - 2018	CAPES/Stic-Amsud
KIGB: Knowing and interacting while gaming for the blind	2014-2016	CAPES/Stic-Amsud

Fonte: O autor.

6.3 Revisitando a Hipótese e Comparação com Trabalhos Relacionados

No Capítulo 1 foi apresentada a hipótese de pesquisa desta tese e que com os resultados apresentados no Capítulo 4 e a avaliação descrita no Capítulo 5, é possível analisar a hipótese para aceitá-la ou rejeitá-la.

Hipótese: *Em um ambiente IoHT, o uso de um framework para o desenvolvimento de aplicações que coletam dados de múltiplos dispositivos vestíveis para monitorar o usuário com o objetivo de detectar quedas e possíveis causas reduz o tempo de desenvolvimento dessas aplicações e os desenvolvedores têm a percepção de que o framework é útil e fácil em comparação com o desenvolvimento sem essa solução?*

Resultado da análise da hipótese: **Aceita**. Diante dos resultados obtidos durante a avaliação, o tempo de desenvolvimento de uma aplicação cujo objetivo é detectar quedas e suas causas utilizando o Ágape foi reduzido em 98,09%. Além disso, os participantes da avaliação perceberam que o Ágape é útil e também é fácil de usar.

Quanto aos trabalhos relacionados apresentados no Capítulo 3, em comparação aos *frameworks* encontrados, este é o único cujo objetivo é detectar quedas e as possíveis causas. Além disso, ele é um *framework* de software, como o Google Fit (GOOGLE, 2022), o Apple HealtKit (APPLE, 2022a), o Withings (NOKIA, 2022) e o JEMF (MACHADO *et al.*, 2014). Ele também possui um modelo de dados orientado a objeto, assim como os três primeiros *frameworks* citados anteriormente. Quanto às funcionalidades comuns consideradas pelo Ágape, ele é o mais abrangente, pois foca em comunicação, heterogeneidade, interoperabilidade, processamento e armazenamento de dados. Essa análise está sumarizada na Tabela 20.

Tabela 20 – Análise do Ágape

Framework	**Objetivo**	**Tipo**	**Principal Característica**	**Tecnologia**	**Modelo de Dados**
Ágape	Detectar quedas e suas possíveis causas a partir de dados brutos oriundos de dispositivos vestíveis da IoHT.	Software	Comunicação, Heterogeneidade, Interoperabilidade, Processamento e Armazenamento de dados	Python	Orientado a objeto

Fonte: O autor.

Em comparação aos modelos de dados existentes, o modelo de dados do Ágape é orientado a objeto, e além dos três trabalhos relacionados citados anteriormente, outros dois modelos propostos em (YANG *et al.*, 2022) e (BEELER, 1998; BLOBEL *et al.*, 2006) também

são. Os modelos do Google Fit, Apple HealthKit e Withings visam a coleta de dados, enquanto os outros dois visam a interoperabilidade entre sistemas de saúde. Os outros seis modelos são ontologias e também possuem focos distintos. O modelo do Ágape é o único focado em detecção de quedas e suas causas a partir de dispositivos IoHT. O sumário dessa análise é feito na Tabela 21.

Tabela 21 – Modelo de Dados do Ágape em comparação com outros

Modelo	Objetivo	Tipo	IoHT
Modelo de dados do Ágape	Detectar quedas e suas causas a partir de episódios gerados dos dados brutos de dispositivos vestíveis da IoHT.	Orientado a objeto	Sim

Fonte: O autor.

6.4 Limitações

Este trabalho propôs um *framework*, o Ágape, para auxiliar no desenvolvimento de soluções de detecção de quedas e suas causas a partir de dados de dispositivos vestíveis que monitoram o usuário e, para isso, também foi definido um modelo de dados. Contudo, o modelo de dados pode evoluir a fim de permitir que o *framework* lide com outros tipos de dados, como de câmeras. Isso pode facilitar a ampliação do escopo do *framework* de quedas para atividades para, por exemplo, obter maior precisão nas ações identificadas.

O Ágape foi avaliado em um ambiente sintético e com poucos usuários. Assim, não há evidências que assegurem o uso do Ágape em um projeto que resulte em uma aplicação real de monitoramento dos usuários e com larga escala de dados sendo enviados simultaneamente, o que pode ser feito a partir de um estudo de caso. Dessa forma, é importante que sejam realizados novos estudos com mais participantes, utilizando dispositivos reais em um estudo de caso.

6.5 Trabalhos Futuros

Esta tese propôs o *framework* Ágape para detecção de quedas e possíveis causas usando dispositivos vestíveis. A partir dos resultados obtidos, surgem os trabalhos futuros listados a seguir:

– **Evolução do Ágape:** o Ágape pode ser evoluído a fim de ampliar o escopo para reconhecimento de atividades ou outros problemas de saúde dos usuários para que possa ser feito um monitoramento mais detalhado das ações do usuário, o que pode ajudar a aprimorar a

sensibilidade dos modelos de detecção de quedas;

- **Evolução da identificação das causas:** Gerar um modelo de Aprendizagem de Máquina preditivo para verificar o comportamento dos dados fisiológicos visando uma análise, como a glicemia que em uma medição pode estar dentro da normalidade, mas a tendência é ir diminuindo com o tempo, o que não é detectado na versão atual do modelo, a não ser por uma nova medição; e
- **Evolução do modelo de dados do Ágape:** o modelo de dados do Ágape pode ser estendido a fim de permitir que os desenvolvedores possam enriquecer a semântica dos episódios com informações que possam ser utilizadas em cada cenário de uso das aplicações desenvolvidas. Além disso, ele pode ser evoluído para lidar com dados de outros tipos monitoramento, como câmeras ou sensores do ambiente;
- **Integração com outros sistemas:** o Ágape pode evoluir para permitir a integração com outros sistemas para prover novos serviços para, por exemplo, criar serviços de AAL que utilizem os dados armazenados no Ágape;
- **Aumento da segurança dos dados:** embora os dados sejam salvos em banco de dados protegidos por senha e passem pela fila presente no *listener* que só aceita acessos via autenticação, os dados são enviados e armazenados sem criptografia. Assim, é importante adicionar mais uma camada de segurança e evoluir o Ágape para aceitar a criptografia dos dados.

REFERÊNCIAS

ACCU-CHEK. **Performa Connect | Brasil**. 2022. Disponível em: https://www.accu-chek.com.br/monitores-de-glicemia/performa-connect. (Acesso em: 19 mar. 2022).

ADA, A. D. A. P. P. C. 7. Diabetes Technology: Standards of Medical Care in Diabetes—2022. **Diabetes Care**, v. 45, n. Supplement$_1$, $p.S97 - -S112, January 2022. ISSN 0149 - 5992$.

AHAMED, F.; SHAHRESTANI, S.; CHEUNG, H. Intelligent fall detection with wearable iot. In: SPRINGER. **Conference on Complex, Intelligent, and Software Intensive Systems**. [*S. l.*], 2019. p. 391–401.

AL-FUQAHA, A.; GUIZANI, M.; MOHAMMADI, M.; ALEDHARI, M.; AYYASH, M. Internet of things: A survey on enabling technologies, protocols, and applications. **IEEE Communications Surveys & Tutorials**, IEEE, v. 17, n. 4, p. 2347–2376, 2015.

ALAMRI, A. Ontology middleware for integration of iot healthcare information systems in ehr systems. **Computers**, MDPI AG, v. 7, n. 4, p. 51, Oct 2018. ISSN 2073-431X. Disponível em: http://dx.doi.org/10.3390/computers7040051.

ALANAZI, M.; SOH, B. Behavioral intention to use iot technology in healthcare settings. **Engineering, Technology & Applied Science Research**, v. 9, n. 5, p. 4769–4774, 2019.

ALI, F.; ISLAM, S. R.; KWAK, D.; KHAN, P.; ULLAH, N.; YOO, S.-j.; KWAK, K. S. Type-2 fuzzy ontology–aided recommendation systems for iot–based healthcare. **Computer Communications**, Elsevier, v. 119, p. 138–155, 2018.

ALMEIDA, M. B. Uma introdução ao xml, sua utilização na internet e alguns conceitos complementares. **Ciência da informação**, SciELO Brasil, v. 31, p. 5–13, 2002.

ALMEIDA, R.; MACEDO, A.; ARAúJO, I. L. de; AGUILAR, P. A.; ANDRADE, R. M. de C. Watchalert: uma evolução do aplicativo falert para detecção de quedas em smartwatches. **Anais do XV Workshop de Ferramentas e Aplicações (WFA)**, 2016.

ALMEIDA, R. L.; DARIN, T. G.; ANDRADE, R. M.; ARAÚJO, I. L. de. Towards developing a practical tool to assist ux evaluation in the iot scenario. In: **Anais Estendidos do XXIV Simpósio Brasileiro de Sistemas Multimídia e Web**. Porto Alegre, RS, Brasil: SBC, 2018. p. 91–95. ISSN 2596-1683. Disponível em: https://sol.sbc.org.br/index.php/webmedia_estendido/article/view/4064.

ALSALIBI, A. I.; SHAMBOUR, M. K. Y.; ABU-HASHEM, M. A.; SHEHAB, M.; SHAMBOUR, Q. Internet of things in health care: A survey. In: ____. **Hybrid Artificial Intelligence and IoT in Healthcare**. Singapore: Springer Singapore, 2021. p. 165–200. ISBN 978-981-16-2972-3. Disponível em: https://doi.org/10.1007/978-981-16-2972-3_9.

ALZUBI, J.; NAYYAR, A.; KUMAR, A. Machine learning from theory to algorithms: an overview. In: IOP PUBLISHING. **Journal of physics: conference series**. [*S. l.*], 2018. v. 1142, n. 1, p. 012012.

ANDERSON, A.; NADALIN, A.; PARDUCCI, B.; ENGOVATOV, D.; LOCKHART, H.; KUDO, M.; HUMENN, P.; GODIK, S.; ANDERSON, S.; CROCKER, S. *et al.* extensible access control markup language (xacml) version 1.0. **OASIS**, 2003.

ANDRADE, A. M.; PEREIRA, M. B.; SILVEIRA, S. H. S.; LINHARES, F. I. F.; NETO, A. H. O.; ANDRADE, R. M. C.; ARAÚJO, I. L. Continuous integration for machine learning experiments reproducibility: a practical study. In: **Anais do I Workshop Brasileiro de Engenharia de Software Inteligente**. Porto Alegre, RS, Brasil: SBC, 2021. p. 25–28. ISSN 0000-0000. Disponível em: https://sol.sbc.org.br/index.php/ise/article/view/17279.

ANDRADE, R. M.; SANTOS, I. S.; ARAÚJO, I. L.; CARVALHO, R. M. Uma metodologia para o ensino teórico e prático da engenharia de software. In: **Fórum de Educaćão em Engenharia de Software (FEES). In VI Congresso Brasileiro de Software: Teoria e Prática (CBSoft)**. [*S. l.*: *s. n.*], 2015.

ANDRADE, R. M. C.; CARVALHO, R. M.; ARAÚJO, I. L. de; OLIVEIRA, K. M.; MAIA, M. E. F. What changes from ubiquitous computing to internet of things in interaction evaluation? In: ____. **Distributed, Ambient and Pervasive Interactions: 5th International Conference, DAPI 2017, Held as Part of HCI International 2017, Vancouver, BC, Canada, July 9–14, 2017, Proceedings**. Cham: Springer International Publishing, 2017. p. 3–21. ISBN 978-3-319-58697-7. Disponível em: https://doi.org/10.1007/978-3-319-58697-7_1.

ANDRADE, R. M. de C.; SANTOS, I. de S.; ARAÚJO, I. L. de; ARAGãO, B. S.; SIEWERDT, F. Retrospective for the last 10 years of teaching software engineering in ufc's computer department. In: **Proceedings of the 31st Brazilian Symposium on Software Engineering**. New York, NY, USA: ACM, 2017. (SBES'17), p. 358–367. ISBN 978-1-4503-5326-7. Disponível em: http://doi.acm.org/10.1145/3131151.3131179.

ANDREAO, R. V.; ATHAYDE, M.; BOUDY, J.; AGUILAR, P.; ANDRADE, R. M. de C.; ARAÚJO, I. L. de. Assistive technologies in smart cities. In: ____. [*S. l.*]: Intechopen, 2018. cap. Raspcare: A Telemedicine Platform for the Treatment and Monitoring of Patients with Chronic Diseases.

ANG, G. C.; LOW, S. L.; HOW, C. H. Approach to falls among the elderly in the community. **Singapore medical journal**, Singapore Medical Association, v. 61, n. 3, p. 116, 2020.

APPLE. **About the HealthKit Framework**. 2022. Disponível em: https://developer.apple.com/documentation/healthkit/about_the_healthkit_framework. (Acesso em: 09 abr 2022).

APPLE. **Usar a detecção de queda no Apple Watch - Suporte da Apple (BR)**. 2022. Disponível em: https://support.apple.com/pt-br/HT208944. (Acesso em: 14 abr. 2022).

APPLE WATCH. **Apple Watch Series 7 - Apple (BR)**. 2022. Disponível em: https://www.apple.com/br/watch/. (Acesso em: 18 mar. 2022).

ARAÚJO, I. L.; SANTOS, I. S.; FILHO, J. a. B. F.; ANDRADE, R. M. C.; NETO, P. S. Generating test cases and procedures from use cases in dynamic software product lines. In: **Proceedings of the Symposium on Applied Computing**. New York, NY, USA: ACM, 2017. (SAC '17), p. 1296–1301. ISBN 978-1-4503-4486-9. Disponível em: http://doi.acm.org/10.1145/3019612.3019790.

ARAÚJO, Í. L. D.; JUNIOR, E. C.; DUARTE, P.; SANTOS, I. D. S.; OLIVEIRA, P. A. M. D.; MENDES, C. M. O.; ANDRADE, R. M. D. C. Towards a taxonomy for the development of older adults healthcare applications. 2020.

ARAÚJO, I. L. de. **Artefatos de Reuso para Internet das Coisas: Arquiteturas, Middlewares, Frameworks e Padrões**. 2018.

ARAÚJO, I. L. de; Dourado, L.; Fernandes, L.; Andrade, R. M. d. C.; Aguilar, P. A. C. An algorithm for fall detection using data from smartwatch. In: **2018 13th Annual Conference on System of Systems Engineering (SoSE)**. [*S. l.*: *s. n.*], 2018. p. 124–131.

AZIZ, O.; PARK, E. J.; MORI, G.; ROBINOVITCH, S. N. Distinguishing the causes of falls in humans using an array of wearable tri-axial accelerometers. **Gait Posture**, v. 39, n. 1, p. 506–512, 2014. ISSN 0966-6362. Disponível em: https://www.sciencedirect.com/science/article/pii/S0966636213005924.

AZIZ, O.; ROBINOVITCH, S. N. An analysis of the accuracy of wearable sensors for classifying the causes of falls in humans. **IEEE Transactions on Neural Systems and Rehabilitation Engineering**, v. 19, n. 6, p. 670–676, 2011.

BAADER, F.; HORROCKS, I.; LUTZ, C.; SATTLER, U. **An Introduction to Description Logic**. [*S. l.*]: Cambridge University Press, 2017.

BAKER, S. B.; XIANG, W.; ATKINSON, I. Internet of things for smart healthcare: Technologies, challenges, and opportunities. **Ieee Access**, IEEE, v. 5, p. 26521–26544, 2017.

BARROSO, H. L. C. T. Trabalho de conclusão de concurso - graduação, **Mona: Solução com Internet das Coisas para detecção de atividades de idosos em um Ambient Assisted Living**. Fortaleza: [*S. n.*], 2022. 49 f.

BEELER, G. W. Hl7 version 3—an object-oriented methodology for collaborative standards development1presented at the international medical informatics association working group 16 conference on standardisation in medical informatics—towards international consensus and cooperation, bermuda, 12 september, 1997.1. **International Journal of Medical Informatics**, v. 48, n. 1, p. 151–161, 1998. ISSN 1386-5056. Disponível em: https://www.sciencedirect.com/science/article/pii/S1386505697001214.

BEPPE, T. A.; ARAúJO, I. L. de; aO, B. S. A.; SANTOS, I. de S.; XIMENES, D.; ANDRADE, R. M. C. Greatest: A card game to motivate the software testing learning. In: **Proceedings of the XXXII Brazilian Symposium on Software Engineering**. New York, NY, USA: ACM, 2018. (SBES '18), p. 298–307. ISBN 978-1-4503-6503-1. Disponível em: http://doi.acm.org/10.1145/3266237.3266254.

BHANDARI, S.; BABAR, N.; GUPTA, P.; SHAH, N.; PUJARI, S. A novel approach for fall detection in home environment. In: IEEE. **2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)**. [*S. l.*], 2017. p. 1–5.

BLOBEL, B.; ENGEL, K.; PHAROW, P. Semantic interoperability–hl7 version 3 compared to advanced architecture standards. **Methods of information in medicine**, v. 45 4, p. 343–53, 2006.

BOLEY, H. The rule markup language: Rdf-xml data model, xml schema hierarchy, and xsl transformations. In: BARTENSTEIN, O.; GESKE, U.; HANNEBAUER, M.; YOSHIE, O. (Ed.). **Web Knowledge Management and Decision Support**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 5–22. ISBN 978-3-540-36524-2.

BOUTELLAA, E.; KERDJIDJ, O.; GHANEM, K. Covariance matrix based fall detection from multiple wearable sensors. **Journal of Biomedical Informatics**, v. 94, p. 103189, 2019. ISSN 1532-0464. Disponível em: http://www.sciencedirect.com/science/article/pii/S1532046419301078.

BOUVIER, D. J. The state of html. **ACM SIGICE Bulletin**, ACM New York, NY, USA, v. 21, n. 2, p. 8–13, 1995.

BRADSHAW, S.; BRAZIL, E.; CHODOROW, K. **MongoDB: the definitive guide: powerful and scalable data storage**. [*S. l.*]: O'Reilly Media, 2019.

BRAR, P. S.; SHAH, B.; SINGH, J.; ALI, F.; KWAK, D. Using modified technology acceptance model to evaluate the adoption of a proposed iot-based indoor disaster management software tool by rescue workers. **Sensors**, MDPI, v. 22, n. 5, p. 1866, 2022.

BROUWER, M. D.; ONGENAE, F.; DANEELS, G.; MUNICIO, E.; FAMAEY, J.; LATRÉ, S.; TURCK, F. D. Personalized real-time monitoring of amateur cyclists on low-end devices: Proof-of-concept & performance evaluation. In: **Companion Proceedings of the The Web Conference 2018**. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018. (WWW '18), p. 1833–1840. ISBN 978-1-4503-5640-4. Disponível em: https://doi.org/10.1145/3184558.3191648.

BUKSMAN, S.; VILELA, A.; PEREIRA, S.; LINO, V.; SANTOS, V. Quedas em idosos: Prevenção. 2008.

BUTTERWORTH, P.; OTIS, A.; STEIN, J. The gemstone object database management system. **Communications of the ACM**, ACM New York, NY, USA, v. 34, n. 10, p. 64–77, 1991.

CARBONARO, A.; PICCININI, F.; REDA, R. Semantic description of healthcare devices to enable data integration. In: LATIFI, S. (Ed.). **Information Technology - New Generations**. Cham: Springer International Publishing, 2018. p. 627–630. ISBN 978-3-319-77028-4.

CATTELL, R. Scalable sql and nosql data stores. **SIGMOD Rec.**, ACM, New York, NY, USA, v. 39, n. 4, p. 12–27, maio 2011. ISSN 0163-5808. Disponível em: http://doi.acm.org/10.1145/1978915.1978919.

CHATTERJEE, P.; ARMENTANO, R. L. Internet of things for a smart and ubiquitous ehealth system. In: **2015 International Conference on Computational Intelligence and Communication Networks (CICN)**. [*S. l.*: *s. n.*], 2015. p. 903–907. ISSN 2472-7555.

CICIRELLI, G.; MARANI, R.; PETITTI, A.; MILELLA, A.; D'ORAZIO, T. Ambient assisted living: A review of technologies, methodologies and future perspectives for healthy aging of population. **Sensors**, v. 21, n. 10, 2021. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/21/10/3549.

CORALLO, A.; CRESPINO, A. M.; LAZOI, M.; LEZZI, M. Model-based big data analytics-as-a-service framework in smart manufacturing: A case study. **Robotics and Computer-Integrated Manufacturing**, v. 76, p. 102331, 2022. ISSN 0736-5845. Disponível em: https://www.sciencedirect.com/science/article/pii/S0736584522000205.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS quarterly**, JSTOR, p. 319–340, 1989.

DE, A.; SAHA, A.; KUMAR, P.; PAL, G. Fall detection approach based on combined two-channel body activity classification for innovative indoor environment. **Journal of Ambient Intelligence and Humanized Computing**, Springer, p. 1–12, 2022.

DINIZ, J. L.; SOUSA, V. F.; COUTINHO, J. F. V.; ARAÚJO, Í. L. d.; ANDRADE, R. M. d. C.; COSTA, J. d. S.; BARBOSA, R. G. B.; MARQUES, M. B. Gerontecnologias e internet das coisas para prevenção de quedas em idosos: revisão integrativa. **Acta Paulista de Enfermagem**, SciELO Brasil, v. 35, 2022.

Dou, D.; Wang, H.; Liu, H. Semantic data mining: A survey of ontology-based approaches. In: **Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)**. [*S. l.*: *s. n.*], 2015. p. 244–251.

DUNCAN, J.; EILBECK, K.; NARUS, S. P.; CLYDE, S.; THORNTON, S.; STAES, C. Building an ontology for identity resolution in healthcare and public health. **Online journal of public health informatics**, University of Illinois at Chicago Library, v. 7, n. 2, 2015.

ERGEN, S. C. Zigbee/ieee 802.15. 4 summary. **UC Berkeley, September**, v. 10, n. 17, p. 11, 2004.

EVANS, M.; NOBLE, J.; HOCHENBAUM, J. **Arduino em ação**. [*S. l.*]: Novatec Editora, 2013.

FANG, R.; POUYANFAR, S.; YANG, Y.; CHEN, S.-C.; IYENGAR, S. S. Computational health informatics in the big data age: A survey. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 49, n. 1, jun 2016. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/2932707.

FAYAD, M.; SCHMIDT, D. C. Object-oriented application frameworks. **Communications of the ACM**, ACM New York, NY, USA, v. 40, n. 10, p. 32–38, 1997.

FREITAS, E.; COSTA, E.; GALERA, C. **Freitas EV, Py L. Tratado de geriatria e gerontologia. 4**. [*S. l.*]: Rio de Janeiro: Guanabara Koogan, 2016.

GJORESKI, H.; STANKOSKI, S.; KIPRIJANOVSKA, I.; NIKOLOVSKA, A.; MLADENOVSKA, N.; TRAJANOSKA, M.; VELICHKOVSKA, B.; GJORESKI, M.; LUŠTREK, M.; GAMS, M. Wearable sensors data-fusion and machine-learning method for fall detection and activity recognition. In: **Challenges and Trends in Multimodal Fall Detection for Healthcare**. [*S. l.*]: Springer, 2020. p. 81–96.

GLIEM, J. A.; GLIEM, R. R. Calculating, interpreting, and reporting cronbach's alpha reliability coefficient for likert-type scales. In: MIDWEST RESEARCH-TO-PRACTICE CONFERENCE IN ADULT, CONTINUING, AND COMMUNITY .... [*S. l.*], 2003.

GOOGLE. **Overview of Google Fit**. 2022. Disponível em: https://developers.google.com/fit/overview. (Acesso em: 09/04/2022).

GROUP, T. H. **Getting Started - Gobot - Golang framework for robotics, physical computing, and the Internet of Things (IoT)**. 2017. Disponível em: https://gobot.io/documentation/getting-started/. (Acesso em: 22/10/2017).

GROUP, T. H. **GitHub - hybridgroup/gobot: Golang framework for robotics, drones, and the Internet of Things (IoT)**. 2022. Disponível em: https://github.com/hybridgroup/gobot/pulls. (Acesso em: 13/04/2022).

GRUBER, T. R. A translation approach to portable ontology specifications. **Knowledge acquisition**, Elsevier, v. 5, n. 2, p. 199–220, 1993.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of things (iot): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, n. 7, p. 1645 – 1660, 2013. ISSN 0167-739X. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond. Disponível em: http://www.sciencedirect.com/science/article/pii/S0167739X13000241.

Haghi Kashani, M.; MADANIPOUR, M.; NIKRAVAN, M.; ASGHARI, P.; MAHDIPOUR, E. A systematic review of iot in healthcare: Applications, techniques, and trends. **Journal of Network and Computer Applications**, v. 192, p. 103164, 2021. ISSN 1084-8045. Disponível em: https://www.sciencedirect.com/science/article/pii/S1084804521001764.

HEMMATPOUR, M.; FERRERO, R.; GANDINO, F.; MONTRUCCHIO, B.; REBAUDENGO, M. Internet of things for fall prediction and prevention. **Journal of Computational Methods in Sciences and Engineering**, IOS Press, n. Preprint, p. 1–8, 2018.

HIMSS. **About Continua | Personal Connected Health Alliance**. 2017. Disponível em: https://www.pchalliance.org/about-continua. (Acesso em: 29/08/2019).

HIREMATH, S.; YANG, G.; MANKODIYA, K. Wearable internet of things: Concept, architectural components and promises for person-centered healthcare. In: **2014 4th International Conference on Wireless Mobile Communication and Healthcare - Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH)**. [*S. l.*: *s. n.*], 2014. p. 304–307.

HORTA, E. T.; LOPES, I. C.; RODRIGUES, J. J. P. C. Ubiquitous mhealth approach for biofeedback monitoring with falls detection techniques and falls prevention methodologies. In: ____. **Mobile Health: A Technology Road Map**. Cham: Springer International Publishing, 2015. p. 43–75. ISBN 978-3-319-12817-7. Disponível em: https://doi.org/10.1007/978-3-319-12817-7_3.

HSIEH, S. L.; CHEN, C. C.; WU, S. H.; YUE, T. W. A wrist -worn fall detection system using accelerometers and gyroscopes. In: **Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control**. [*S. l.*: *s. n.*], 2014. p. 518–523.

HUANG, J.; WU, X.; HUANG, W.; WU, X.; WANG, S. Internet of things in health management systems: A review. **International Journal of Communication Systems**, Wiley Online Library, v. 34, n. 4, p. e4683, 2021.

HUAWEI. **HUAWEI WATCH GT 3 - HUAWEI Brazil**. 2022. Disponível em: https://consumer.huawei.com/br/wearables/watch-gt3/. (Acesso em: 19 mar. 2022).

HUSSAIN, F.; HUSSAIN, F.; HAQ, M. Ehatisham-ul; AZAM, M. A. Activity-aware fall detection and recognition based on wearable sensors. **IEEE Sensors Journal**, IEEE, v. 19, n. 12, p. 4528–4536, 2019.

IBGE. **Projeções da População - IBGE :: Instituto Brasileiro de Geografia e Estatística**. 2018. Disponível em: https://www.ibge.gov.br/estatisticas-novoportal/sociais/populacao/9109-projecao-da-populacao.html?=&t=o-que-e. (Acesso em: 18 mar. 2022).

IMRAN; GHAFFAR, Z.; ALSHAHRANI, A.; FAYAZ, M.; ALGHAMDI, A. M.; GWAK, J. A topical review on machine learning, software defined networking, internet of things applications:

Research limitations and challenges. **Electronics**, v. 10, n. 8, 2021. ISSN 2079-9292. Disponível em: https://www.mdpi.com/2079-9292/10/8/880.

INFLUXDATA. **Get InfluxDB | #1 Ranked Time Series Database | InfluxData**. 2022. Disponível em: https://www.influxdata.com/get-influxdb/. (Acesso em: 26 mar. 2022).

International Telecommunication Union. Overview of the Internet of things. **Series Y: Global information infrastructure, internet protocol aspects and next-generation networks - Frameworks and functional architecture models**, p. 22, 2012.

IOTIVITY. **architecture [Wiki]**. 2017. Disponível em: https://wiki.iotivity.org/architecture. (Acesso em 22 out. 2017).

IOTIVITY. **IoTivity**. 2021. Disponível em: https://iotivity.org/. (Acesso em: 13 abr. 2022).

IPIÑA, D. López-de; LAISECA, X.; BARBIER, A.; AGUILERA, U.; ALMEIDA, A.; ORDUÑA, P.; VAZQUEZ, J. I. Infrastructural support for ambient assisted living. In: SPRINGER. **3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008**. [*S. l.*], 2009. p. 66–75.

ISLAM, S. M. R.; KWAK, D.; KABIR, M. H.; HOSSAIN, M.; KWAK, K.-S. The internet of things for health care: A comprehensive survey. **IEEE Access**, v. 3, p. 678–708, 2015. ISSN 2169-3536.

ISO/IEC. **Internet of Things: Preliminary Report**. [*S. l.*], 2014.

ISTEPANIAN, R. S. H.; JOVANOV, E.; ZHANG, Y. T. Guest editorial introduction to the special section on m-health: Beyond seamless mobility and global wireless health-care connectivity. **IEEE Transactions on Information Technology in Biomedicine**, v. 8, n. 4, p. 405–414, Dec 2004. ISSN 1089-7771.

JANOWICZ, K.; HALLER, A.; COX, S. J.; PHUOC, D. L.; LEFRANçOIS, M. Sosa: A lightweight ontology for sensors, observations, samples, and actuators. **Journal of Web Semantics**, 2018. ISSN 1570-8268. Disponível em: http://www.sciencedirect.com/science/article/pii/S1570826818300295.

JOSHI, A.; KALE, S.; CHANDEL, S.; PAL, D. K. Likert scale: Explored and explained. **British journal of applied science & technology**, SCIENCEDOMAIN International, v. 7, n. 4, p. 396, 2015.

KHOJASTEH, S. B.; VILLAR, J. R.; CHIRA, C.; GONZÁLEZ, V. M.; CAL, E. de la. Improving fall detection using an on-wrist wearable accelerometer. **Sensors (Basel, Switzerland)**, Multidisciplinary Digital Publishing Institute (MDPI), v. 18, n. 5, 2018.

KITCHENHAM, B.; BRERETON, O. P.; BUDGEN, D.; TURNER, M.; BAILEY, J.; LINKMAN, S. Systematic literature reviews in software engineering–a systematic literature review. **Information and software technology**, Elsevier, v. 51, n. 1, p. 7–15, 2009.

KOSKIMIES, K.; MÖSSENBÖCK, H. Designing a framework by stepwise generalization. In: SPRINGER. **European Software Engineering Conference**. [*S. l.*], 1995. p. 479–498.

KUMAR, B. A. Ontology based data model for context aware mhealth application. In: **2015 1st International Conference on Next Generation Computing Technologies (NGCT)**. [*S. l.*: *s. n.*], 2015. p. 274–279.

KUMAR, P. M.; Devi Gandhi, U. A novel three-tier internet of things architecture with machine learning algorithm for early detection of heart diseases. **Computers Electrical Engineering**, v. 65, p. 222–235, 2018. ISSN 0045-7906. Disponível em: https://www.sciencedirect.com/science/article/pii/S0045790617328410.

KUMAR, V. *et al.* Ontology based public healthcare system in internet of things (iot). **Procedia Computer Science**, Elsevier, v. 50, p. 99–102, 2015.

K'WATCH. **K'Watch Glucose - PKVitality**. 2022. Disponível em: https://www.pkvitality.com/ktrack-glucose/. (Acesso em: 19 mar. 2022).

LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. **Biometrics**, [Wiley, International Biometric Society], v. 33, n. 1, p. 159–174, 1977. ISSN 0006341X, 15410420. Disponível em: http://www.jstor.org/stable/2529310.

LI, J.; XU, Y.; CUI, L.; WEI, F. Markuplm: Pre-training of text and markup language for visually-rich document understanding. **arXiv preprint arXiv:2110.08518**, 2021.

LINHARES, I.; ANDRADE, R.; JUNIOR, E. C.; ALMIR, P.; OLIVEIRA, B.; AGUILAR, P. Lessons learned from the development of mobile applications for fall detection. In: **The Ninth International Conference on Global Health Challenges**. [*S. l.*: *s. n.*], 2020. p. 18–25.

MACEDO, A.; ARAÚJO, I. L. de; ANDRADE, R. M. de C. Introdução ao android wear: desenvolvendo aplicações para smartwatch. In: **III Escola Regional de Informática do Piauí**. [*S. l.*: *s. n.*], 2017.

MACHADO, M. F. T.; NASCIMENTO, B. S.; VIVACQUA, A. S.; BORGES, M. R. S. Jemf: A framework for the development of mobile systems for emergency management. In: BALOIAN, N.; BURSTEIN, F.; OGATA, H.; SANTORO, F.; ZURITA, G. (Ed.). **Collaboration and Technology**. Cham: Springer International Publishing, 2014. p. 239–254. ISBN 978-3-319-10166-8.

MALHOTRA, R. **Empirical research in software engineering: concepts, analysis, and applications**. [*S. l.*]: CRC press, 2016.

MARINESCU, D. C. **Cloud computing: theory and practice**. [*S. l.*]: Morgan Kaufmann, 2022.

MCROBERTS, M. **Arduino básico**. [*S. l.*]: Novatec Editora, 2018.

MEDDRA. **Medical Dictionary for Regulatory Activities Terminology (MedDRA) - Cardiac disorder - Classes | NCBO BioPortal**. 2021. Disponível em: http://bioportal.bioontology.org/ontologies/MEDDRA/?p=classes&conceptid=http%3A%2F%2Fpurl.bioontology.org%2Fontology%2FMEDDRA%2F10061024. (Acesso em: 13 abr. 2022).

MEDICAL, A. **Premium Wireless Blood Pressure Monitor | A&D Medical**. 2022. Disponível em: https://medical.andonline.com/product/premium-wireless-blood-pressure-monitor/ua-651ble. (Acesso em: 19 mar. 2022).

MISHRA, N.; PANDYA, S. Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. **IEEE Access**, v. 9, p. 59353–59377, 2021.

MOKHTARI, G.; AMINIKHANGHAHI, S.; ZHANG, Q.; COOK, D. J. Fall detection in smart home environments using uwb sensors and unsupervised change detection. **Journal of Reliable Intelligent Environments**, Springer, v. 4, n. 3, p. 131–139, 2018.

MONEKOSSO, D.; FLOREZ-REVUELTA, F.; REMAGNINO, P. Ambient assisted living [guest editors' introduction]. **IEEE Intelligent Systems**, IEEE, v. 30, n. 4, p. 2–6, 2015.

MORIN, B.; Mé, L.; DEBAR, H.; DUCASSé, M. A logic-based model to support alert correlation in intrusion detection. **Information Fusion**, v. 10, n. 4, p. 285 – 299, 2009. ISSN 1566-2535. Special Issue on Information Fusion in Computer Security. Disponível em: http://www.sciencedirect.com/science/article/pii/S1566253509000177.

MUBASHIR, M.; SHAO, L.; SEED, L. A survey on fall detection: Principles and approaches. **Neurocomputing**, Elsevier, v. 100, p. 144–152, 2013.

NAZARI, E.; SHAHRIARI, M. H.; TABESH, H. Applications of framework in health care: A survey. **Frontiers in Health Informatics**, v. 8, n. 1, p. 16, 2019.

Nações Unidas. **World Population Prospects**. 2017. Disponível em: https://www.un.org/development/desa/pd/sites/www.un.org.development.desa.pd/files/files/documents/2020/Jan/un_2017_world_population_prospects-2017_revision_databooklet.pdf. (Acesso em: 25/04/2022).

NGUYEN, H.; MIRZA, F.; NAEEM, M. A.; BAIG, M. M. Falls management framework for supporting an independent lifestyle for older adults: a systematic review. **Aging Clinical and Experimental Research**, v. 30, n. 11, p. 1275–1286, Nov 2018. ISSN 1720-8319. Disponível em: https://doi.org/10.1007/s40520-018-1026-6.

NIZAM, Y.; JAMIL, M. M. A. Classification of daily life activities for human fall detection: A systematic review of the techniques and approaches. In: **Challenges and Trends in Multimodal Fall Detection for Healthcare**. [*S. l.*]: Springer, 2020. p. 137–179.

NOGUEIRA, T. P.; BRAGA, R. B.; OLIVEIRA, C. T. de; MARTIN, H. Framestep: A framework for annotating semantic trajectories based on episodes. **Expert Systems with Applications**, v. 92, p. 533 – 545, 2018. ISSN 0957-4174. Disponível em: http://www.sciencedirect.com/science/article/pii/S0957417417306796.

NOKIA. **Withings API**. 2022. Disponível em: http://developer.withings.com/oauth2/#tag/glossary. (Acesso em: 09 abr. 2022).

NORD, J. H.; KOOHANG, A.; PALISZKIEWICZ, J. The internet of things: Review and theoretical framework. **Expert Systems with Applications**, Elsevier, v. 133, p. 97–108, 2019.

NORMAN, A. **Aprendizagem De Máquina Em Ação**. , 2022. ISBN 9785042765827. Disponível em: https://books.google.com.br/books?id=Zur3DwAAQBAJ.

NOURA, M.; ATIQUZZAMAN, M.; GAEDKE, M. Interoperability in internet of things: Taxonomies and open challenges. **Mobile Networks and Applications**, Springer, v. 24, n. 3, p. 796–809, 2019.

OASIS Standard Incorporating Approved. Mqtt version 3.1. 1 plus errata 01. 2015.

OLIVEIRA, B. S.; ARAÚJO, Í. L.; PAIVA, J. O.; JUNIOR, E. C.; ANDRADE, R. M. Refactoring decision based on measurements for ioht apps. In: **XVII Brazilian Symposium on Information Systems**. [*S. l.*: *s. n.*], 2021. p. 1–9.

OLIVEIRA, P.; ARAÚJO, I.; JUNIOR, E.; DUARTE, P.; SANTOS, I. S.; ANDRADE, R. M.; BARRETO, I. C. H.; ANDRADE, L. O. M. Dorsal: Ferramenta para geração de modelos de dados para aplicações voltadas a saúde e cuidado de idosos. In: SBC. **Anais Estendidos do XX Simpósio Brasileiro de Computação Aplicada à Saúde**. [*S. l.*], 2020. p. 1–6.

OLIVEIRA, S. de. **Internet das coisas com ESP8266, Arduino e Raspberry PI**. [*S. l.*]: Novatec Editora, 2017.

OMRON. **Monitor de Pressão Arterial de Pulso com Bluetooth CONNECT | Omron Healthcare Brasil**. 2022. Disponível em: https://www.omronbrasil.com/monitor-pressao-arterial-bluetooth-connect-hem6232t/p. (Acesso em: 19 mar. 2022).

OMS. **Global Report on Falls Prevention in Older Age**. 2007. Disponível em: https://www.who.int/ageing/publications/Falls_prevention7March.pdf. (Acesso em: 18 mar. 2022).

OMS. **Falls - World Health Organization**. 2021. Disponível em: http://www.who.int/news-room/fact-sheets/detail/falls. (Acesso em: 16 mar. 2022).

PALIPANA, S.; ROJAS, D.; AGRAWAL, P.; PESCH, D. Falldefi: Ubiquitous fall detection using commodity wi-fi devices. **Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies**, ACM New York, NY, USA, v. 1, n. 4, p. 1–25, 2018.

PATERSON, J.; EDLICH, S.; HÖRNING, H.; HÖRNING, R. db4o. Springer, 2006.

PATHAK, A.; KALAIARASAN, C. Rabbitmq queuing mechanism of publish subscribe model for better throughput and response. In: IEEE. **2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT)**. [*S. l.*], 2021. p. 1–7.

PINTO, A.; ASSIS, G. A. de; TORRES, L. C.; BELTRAME, T.; DOMINGUES, D. M. Wearables and detection of falls: A comparison of machine learning methods and sensors positioning. **Neural Processing Letters**, Springer, p. 1–15, 2022.

PIVA, L. S.; FERREIRA, A. B.; BRAGA, R. B.; ANDRADE, R. falert: An android system for monitoring falls in people with special care. In: **Workshop on Tools and Applications of the 20th Brazilian Symposium on Multimedia and Web Systems**. [*S. l.*: *s. n.*], 2014.

POKLUDA, A.; SUN, W. Benchmarking failover characteristics of large-scale data storage applications: Cassandra and voldemort. **Alexanderpokluda. Ca**, 2013.

POLAR. **Accessories for heart rate monitors, fitness trackers and cycling sensors | Polar Global**. 2022. Disponível em: https://www.polar.com/en/products/accessories#pf6=1. (Acesso em: 19 mar. 2022).

PREE, W. Framework development and reuse support. **Burnett et al.[5]**, Citeseer, 1995.

PRIYADARSHINI, S. B. B.; SHARMA, D. K.; SHARMA, R.; CENGIZ, K. **The Role of the Internet of Things (IoT) in Biomedical Engineering: Present Scenario and Challenges**. [*S. l.*]: Apple Academic Press, 2022.

PSF, P. S. F. **Welcome to Python.org**. 2022. Disponível em: https://www.python.org/. (Acesso em: 26 mar. 2022).

RAEVE, N. D.; SHAHID, A.; SCHEPPER, M. de; POORTER, E. D.; MOERMAN, I.; VERHAEVERT, J.; TORRE, P. V.; ROGIER, H. Bluetooth-low-energy-based fall detection and warning system for elderly people in nursing homes. **Journal of Sensors**, Hindawi, v. 2022, 2022.

RAGHUPATHI, W.; RAGHUPATHI, V. Big data analytics in healthcare: promise and potential. **Health information science and systems**, Springer, v. 2, n. 1, p. 1–10, 2014.

RAHMANI, A. M.; BAYRAMOV, S.; KALEJAHI, B. K. Internet of things applications: Opportunities and threats. **Wireless Personal Communications**, Springer, v. 122, n. 1, p. 451–476, 2022.

RAZZAQUE, M. A.; MILOJEVIC-JEVRIC, M.; PALADE, A.; CLARKE, S. Middleware for internet of things: A survey. **IEEE Internet of Things Journal**, v. 3, n. 1, p. 70–95, Feb 2016. ISSN 2327-4662.

REDA, R.; PICCININI, F.; CARBONARO, A. Towards consistent data representation in the iot healthcare landscape. In: **Proceedings of the 2018 International Conference on Digital Health**. New York, NY, USA: ACM, 2018. (DH '18), p. 5–10. ISBN 978-1-4503-6493-5. Disponível em: http://doi-acm-org.ez11.periodicos.capes.gov.br/10.1145/3194658.3194668.

RICHARDSON, M.; WALLACE, S. Primeiros passos com o raspberry pi. **Primeira Edição. Novatec Editora Ltda**, v. 20, 2013.

RUENIN, P.; TECHAKAEW, S.; TOWATRAKOOL, P.; CHAWACHAT, J. A fall alert system with prior-fall activity identification. **CoRR**, abs/2201.02803, 2022. Disponível em: https://arxiv.org/abs/2201.02803.

RUGGERI, G.; BRIANTE, O. A framework for iot and e-health systems integration based on the social internet of things paradigm. In: **2017 International Symposium on Wireless Communication Systems (ISWCS)**. [*S. l.*: *s. n.*], 2017. p. 426–431. ISSN 2154-0225.

RUSSELL, S.; NORVIG, P.; INTELLIGENCE, A. Artificial inteligence: A modern approach. **Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs**, v. 25, p. 27, 2009.

SAHU, M. L.; ATULKAR, M.; AHIRWAL, M. K. Iot-based smart healthcare system: A review on constituent technologies. **Journal of Circuits, Systems and Computers**, v. 30, n. 11, p. 2130008, 2021. Disponível em: https://doi.org/10.1142/S0218126621300087.

SALEH, M.; JEANNÈS, R. L. B. Elderly fall detection using wearable sensors: A low cost highly accurate algorithm. **IEEE Sensors Journal**, IEEE, v. 19, n. 8, p. 3156–3164, 2019.

SAMSUNG. **Galaxy Watch4 BT 44mm - Preto | Samsung Brasil**. 2022. Disponível em: https://www.samsung.com/br/watches/galaxy-watch/galaxy-watch4-black-bluetooth-sm-r870nzkpzto/. (Acesso em: 19 mar. 2022).

SARKER, I. H. Machine learning: Algorithms, real-world applications and research directions. **SN Computer Science**, Springer, v. 2, n. 3, p. 1–21, 2021.

SASAKI, S.; YAMAMOTO, H.; KITAGAWA, K.; WADA, C. Identification of the cause of fall during the pre-impact fall period. **Journal of Physical Therapy Science**, v. 34, n. 4, p. 320–326, 2022.

SHAHZAD, S. K.; AHMED, D.; NAQVI, M. R.; MUSHTAQ, M. T.; IQBAL, M. W.; MUNIR, F. Ontology driven smart health service integration. **Computer Methods and Programs in Biomedicine**, v. 207, p. 106146, 2021. ISSN 0169-2607. Disponível em: https://www.sciencedirect.com/science/article/pii/S0169260721002200.

SHELBY, Z.; HARTKE, K.; BORMANN, C. **The Constrained Application Protocol (CoAP)**. RFC Editor, 2014. RFC 7252. (Request for Comments, 7252). Disponível em: https://www.rfc-editor.org/info/rfc7252.

SILVA, M. S. **HTML5: a linguagem de marcação que revolucionou a web**. [*S. l.*]: Novatec Editora, 2019.

STANDARD, O. Mqtt version 5.0. **Retrieved June**, v. 22, p. 2020, 2019.

STANOJEVIć, V.; VLAJIć, S.; MILIć, M.; OGNJANOVIć, M. Guidelines for framework development process. In: **2011 7th Central and Eastern European Software Engineering Conference (CEE-SECR)**. [*S. l.*: *s. n.*], 2011. p. 1–9.

STONEBRAKER, M. Sql databases v. nosql databases. **Communications of the ACM**, ACM New York, NY, USA, v. 53, n. 4, p. 10–11, 2010.

SUNDMAEKER, H.; GUILLEMIN, P.; FRIESS, P.; WOELFFLÉ, S. Vision and challenges for realising the internet of things. **Cluster of European Research Projects on the Internet of Things, European Commision**, 2010.

TAHIR, M.; KHAN, F.; BABAR, M.; ARIF, F.; KHAN, F. Framework for better reusability in component based software engineering. **the Journal of Applied Environmental and Biological Sciences (JAEBS)**, v. 6, n. 4S, p. 77–81, 2016.

TANWAR, R.; NANDAL, N.; ZAMANI, M.; MANAF, A. A. Pathway of trends and technologies in fall detection: A systematic review. **Healthcare**, v. 10, n. 1, 2022. ISSN 2227-9032. Disponível em: https://www.mdpi.com/2227-9032/10/1/172.

TSAI, J.-C.; LEU, J.-S.; PRAKOSA, S. W.; HSIAO, L.-C.; HUANG, P.-C.; YANG, S.-Y.; HUANG, Y.-T. Design and implementation of an internet of healthcare things system for respiratory diseases. **Wireless Personal Communications**, Springer, v. 117, n. 2, p. 337–353, 2021.

USAK, M.; KUBIATKO, M.; SHABBIR, M. S.; DUDNIK, O. V.; JERMSITTIPARSERT, K.; RAJABION, L. Health care service delivery based on the internet of things: A systematic and comprehensive study. **International Journal of Communication Systems**, Wiley Online Library, v. 33, n. 2, p. e4179, 2020.

VALIANTE, M.; GUIDA, D.; SETA, M. D.; BOZZANO, F. A spatiotemporal object-oriented data model for landslides (loom). **Landslides**, Springer, v. 18, n. 4, p. 1231–1244, 2021.

VASSEUR, J.-P.; DUNKELS, A. **Interconnecting Smart Objects with IP: The Next Internet**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010. ISBN 0123751659, 9780123751652.

W3C. **SOSA Ontology - Spatial Data on the Web Working Group**. 2016. Disponível em: https://www.w3.org/2015/spatial/wiki/SOSA_Ontology. (Acesso em: 09 jul. 2019).

W3C. **Semantic Sensor Network Ontology**. 2018. Disponível em: https://www.w3.org/TR/vocab-ssn/. (Acesso em: 07 dez. 2018).

WANG, P.; LI, Q.; YIN, P.; WANG, Z.; LING, Y.; GRAVINA, R.; LI, Y. A convolution neural network approach for fall detection based on adaptive channel selection of uwb radar signals. **Neural Computing and Applications**, Springer, p. 1–14, 2022.

WARRENS, M. J. Equivalences of weighted kappas for multiple raters. **Statistical Methodology**, v. 9, n. 3, p. 407 – 422, 2012. ISSN 1572-3127. Disponível em: http://www.sciencedirect.com/science/article/pii/S1572312711001171.

WILSON, D.; WILSON, S. Writing frameworks-capturing your expertise about a problem domain. **Tutorial Notes, OOPSLA'93**, 1993.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in software engineering**. [*S. l.*]: Springer Science & Business Media, 2012.

WOLGAST, G.; EHRENBORG, C.; ISRAELSSON, A.; HELANDER, J.; JOHANSSON, E.; MANEFJORD, H. Wireless body area network for heart attack detection [education corner]. **IEEE antennas and propagation magazine**, IEEE, v. 58, n. 5, p. 84–92, 2016.

WU, T.; GU, Y.; CHEN, Y.; XIAO, Y.; WANG, J. A mobile cloud collaboration fall detection system based on ensemble learning. **arXiv preprint arXiv:1907.04788**, 2019.

Xu, B.; Xu, L. D.; Cai, H.; Xie, C.; Hu, J.; Bu, F. Ubiquitous data accessing method in iot-based information system for emergency medical services. **IEEE Transactions on Industrial Informatics**, v. 10, n. 2, p. 1578–1586, May 2014. ISSN 1551-3203.

YANG, L.; ZOU, W.; WANG, J.; TANG, Z. Edgeshare: A blockchain-based edge data-sharing framework for industrial internet of things. **Neurocomputing**, v. 485, p. 219–232, 2022. ISSN 0925-2312. Disponível em: https://www.sciencedirect.com/science/article/pii/S0925231221016350.

YANG, Y. J.; KIM, S. Y.; CHOI, G. J.; CHO, E. S.; KIM, C. J.; KIM, S. D. A uml-based object-oriented framework development methodology. In: IEEE. **Proceedings 1998 asia pacific software engineering conference (Cat. No. 98EX240)**. [*S. l.*], 1998. p. 211–218.

YANG, Z.; JIANG, K.; LOU, M.; GONG, Y.; ZHANG, L.; LIU, J.; BAO, X.; LIU, D.; YANG, P. Defining health data elements under the hl7 development framework for metadata management. **Journal of Biomedical Semantics**, BioMed Central, v. 13, n. 1, p. 1–15, 2022.

ZELKOWITZ, M. V.; WALLACE, D. R.; BINKLEY, D. W. Experimental validation of new software technology. In: **Lecture notes on empirical software engineering**. [*S. l.*]: World Scientific, 2003. p. 229–263.

APÊNDICE A – FORMULÁRIO PRÉ-EXPERIMENTO

1. Nome
2. E-mail
3. Escolaridade
   - Graduando
   - Graduado
   - Mestrando
   - Mestre
   - Doutorando
   - Doutor
4. Você já desenvolveu para Android?
   - Sim, já desenvolvi.
   - Não, não desenvolvi.
5. Caso a resposta seja "Sim", informe há quanto tempo você desenvolve com Android.
6. Você já desenvolveu um software (ou scripts) na linguagem de programação Python?
   - Sim, já desenvolvi.
   - Não, não desenvolvi.
7. Caso a resposta seja "Sim", informe há quanto tempo você desenvolve em Python.
8. Você desenvolveu algum software utilizando um framework?
   - Sim, já desenvolvi.
   - Não, não desenvolvi.
9. Caso a resposta seja "Sim", informe quais frameworks você utilizou.
10. Você conhece o termo "Internet das Coisas"(IoT)?
    - Sim, conheço o termo e já desenvolvi um aplicativo de IoT.
    - Sim, conheço o termo, mas nunca desenvolvi um aplicativo IoT.
    - Não, não conheço.
11. Você já desenvolveu um aplicativo que utiliza dados de sensores?
    - Sim, já desenvolvi.
    - Não, não desenvolvi.
12. Caso a resposta da pergunta anterior seja "Sim", informe os sensores com que já trabalhou.
13. Você desenvolveu algum software para saúde?
    - Sim, já desenvolvi.

   - Não, não desenvolvi.

14. Caso a resposta seja "Sim", informe o cenário do software desenvolvido.
15. Você desenvolveu algum software de IoT para saúde?
   - Sim, já desenvolvi.
   - Não, não desenvolvi.
16. Caso a resposta seja "Sim", informe o cenário da aplicação desenvolvida.
17. Você utilizou frameworks para o desenvolvimento de um software de IoT para saúde?
   - Sim, já utilizei.
   - Não, não utilizei.
18. Se a resposta anterior for "Sim", informe quais frameworks utilizou.

APÊNDICE B – FORMULÁRIO DE AVALIAÇÃO DO ÁGAPE

– *Nome*

– *Percepção de facilidade de uso*

1. Eu me senti confuso ao usar o framework.
   – Concordo Totalmente
   – Concordo
   – Neutro
   – Discordo
   – Discordo Totalmente
2. Eu cometi erros quando usei o framework, pois não sabia utilizá-lo.
   – Concordo Totalmente
   – Concordo
   – Neutro
   – Discordo
   – Discordo Totalmente
3. Eu me senti frustrado frequentemente.
   – Concordo Totalmente
   – Concordo
   – Neutro
   – Discordo
   – Discordo Totalmente
4. Eu precisei consultar o tutorial para usar o framework.
   – Concordo Totalmente
   – Concordo
   – Neutro
   – Discordo
   – Discordo Totalmente
5. Eu achei fácil recuperar de erros que aconteceram no framework.
   – Concordo Totalmente
   – Concordo
   – Neutro
   – Discordo

    - Discordo Totalmente

6. Eu achei fácil fazer o framework executar o que eu queria.
    - Concordo Totalmente
    - Concordo
    - Neutro
    - Discordo
    - Discordo Totalmente
7. O framework se comportou de modo inesperado frequentemente.
    - Concordo Totalmente
    - Concordo
    - Neutro
    - Discordo
    - Discordo Totalmente
8. Eu achei desconfortável utilizar o framework.
    - Concordo Totalmente
    - Concordo
    - Neutro
    - Discordo
    - Discordo Totalmente
9. Foi fácil lembrar como executar uma tarefa no framework (e.g., adicionar um algoritmo).
    - Concordo Totalmente
    - Concordo
    - Neutro
    - Discordo
    - Discordo Totalmente

- *Percepção de utilidade*

1. O uso de um framework facilitou o desenvolvimento de um software IoHT.
    - Concordo Totalmente
    - Concordo
    - Neutro
    - Discordo

    - Discordo Totalmente

2. O framework reduziu o tempo de desenvolvimento de um software IoHT.
    - Concordo Totalmente
    - Concordo
    - Neutro
    - Discordo
    - Discordo Totalmente
3. O uso do framework contribui para melhorar qualidade do meu trabalho.
    - Concordo Totalmente
    - Concordo
    - Neutro
    - Discordo
    - Discordo Totalmente
4. O framework ajuda a processar os dados e melhor a detecção de quedas.
    - Concordo Totalmente
    - Concordo
    - Neutro
    - Discordo
    - Discordo Totalmente
5. O framework é útil no desenvolvimento de software.
    - Concordo Totalmente
    - Concordo
    - Neutro
    - Discordo
    - Discordo Totalmente

- *Se desejar, escreva comentários ou sugestões*