UNIVERSIDADE FEDERAL DO MARANHÃO

Programa de Pós-Graduação em Ciência da Computação

Jean Pablo Marques Mendes

Um Framework para Facilitar o Desenvolvimento de Aplicações Móveis de Fenotipagem Digital

São Luís
2022

Jean Pablo Marques Mendes

Um Framework para Facilitar o Desenvolvimento de Aplicações Móveis de Fenotipagem Digital

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação - da Universidade Federal do Maranhão, como requisito para obtenção do título de Mestre em Ciência da Computação.

Universidade Federal do Maranhão – UFMA

Programa de Pós-Graduação em Ciência da Computação

Orientador: Prof. Dr. Ariel Soares Teles

Coorientador: Prof. Dr. Francisco José da Silva e Silva

São Luís - MA

2022

Jean Pablo Marques Mendes

Um Framework para Facilitar o Desenvolvimento de Aplicações Móveis de Fenotipagem Digital/ Jean Pablo Marques Mendes. – São Luís - MA, 2022.

91 f.

Orientador: Prof. Dr. Ariel Soares Teles

Dissertação (Mestrado) – Universidade Federal do Maranhão – UFMA
Programa de Pós-Graduação em Ciência da Computação, 2022.

1. Saúde Mental. 2. Fenotipagem Digital. 3. Aplicativos de Sensoriamento. 4. Datasets. 5. Dados de sensores I. Soares Teles, Ariel, orient. II. Título.

CDU XXX

Jean Pablo Marques Mendes

Um Framework para Facilitar o Desenvolvimento de Aplicações Móveis de Fenotipagem Digital

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação - da Universidade Federal do Maranhão, como requisito para obtenção do título de Mestre em Ciência da Computação.

Trabalho Aprovado. São Luís - MA, 06 de Maio de 2022:

Prof. Dr. Ariel Soares Teles
Orientador
Instituto Federal do Maranhão

Prof. Dr. Francisco José da Silva e Silva
Coorientador
Universidade Federal do Maranhão

Prof. Dr. Luciano Reis Coutinho
Examinador Interno
Universidade Federal do Maranhão

Prof. Dr. Joel José Puga Coelho Rodrigues
Examinador Externo
Universidade Federal do Piauí

São Luís - MA
2022

Em memória ao Prof. Zair Abdelouahab, com quem tive o privilégio de conviver e de aprender com seus exemplos.

Agradecimentos

A Deus, o que seria de mim sem a fé que eu tenho nele.

Aos meus pais, José Pedro e Maria Ivoneide, aos meus irmãos e toda minha família que, com muito carinho e apoio, não mediram esforços para eu chegar até esta etapa de minha vida.

Ao Professor Ariel Teles pela paciência na orientação e incentivo que tornaram possível a conclusão desta dissertação.

Ao meu Co-orientador Francisco Silva que com sua ajuda me propiciou elaboração, criação e desenvolvimento deste trabalho.

Aos colegas de laboratório, em especial a Ivan Moura, Raimundo Castro, José Daniel, Luís Eduardo, André, Reinaldo pela ajuda e apoio que todos prestaram na produção deste trabalho.

A todos os professores do mestrado, que foram tão importantes na minha vida acadêmica e no desenvolvimento desta dissertação.

Aos amigos e colegas, pelo incentivo e pelo apoios constantes.

"E bonito e virtuoso, que tanto valor merece,
quem tem viva fé em Deus, nada de mal lhe acontece".
Valdemor Uchoa Mendes

Resumo

Os transtornos de saúde mental têm alta prevalência na população mundial. Com a pandemia do coronavírus (COVID-19), houve a exigência de distanciamento social, agravando os problemas relacionados a saúde mental e bem-estar. A proliferação dos *smartphones* apresenta oportunidades para a coleta de dados para estudar o comportamento e saúde humana. Eles têm sido utilizados em estudos na saúde mental, por possuírem vários sensores embutidos que podem capturar medições no dia-a-dia das pessoas. Tradicionalmente, os transtornos mentais são diagnosticados por profissionais de saúde mental (e.g., psiquiatras, psicólogos) apenas com base nos sintomas identificados a partir de entrevistas com pacientes e experiências autor-relatadas. No entanto, os pacientes costumam recorrer a eventos ocorridos dias, semanas ou meses atrás, o que pode comprometer o diagnóstico, devido ao vieses de memória e desejabilidade. Para mitigar esses vieses, surge a fenotipagem digital, coletando dados de forma passiva (sem interação direta do usuário) com uso de dispositivos móveis. Este trabalho visa apresentar um framework para facilitar o desenvolvimento de aplicativos móveis de fenotipagem digital, chamado *OpenDP*. A solução proposta é extensível e reusável, por permitir a inclusão de módulos de processamento de dados brutos de contexto, e podendo ser aplicada a diferentes transtornos mentais e ao monitoramento do bem-estar humano. Além disso, o framework foi desenvolvido sobre o *middleware* Mobile Hub (M-Hub)/*Camada de Distribuição de Dados de Contexto* (CDDL) para coletar os dados de sensores (físicos e virtuais) e distribuí-los entre os componentes internos do framework e com o *broker* externo. Estudos de caso foram conduzidos para demonstrar que a solução proposta conseguiu compor fenótipos digitais a partir da inferência de informações de alto nível geradas pelos módulos de processamento de dados. Também foi demonstrado a capacidade da solução móvel em adicionar módulos de processamento de dados (*plugins*). Visando desenvolvedor uma solução móvel para dispositivos móveis que não cause um grande impacto no consumo de energia, foi realizada uma avaliação experimental analisando o impacto no consumo de energia do *smartphone* do usuário. Os resultados foram satisfatórios, mostrando através da avaliação experimental que o consumo de bateria foi pequeno.

Palavras-chave: Saúde Mental, Fenotipagem Digital, Monitoramento Remoto, Aplicações de Sensoriamento, Framework.

Abstract

Mental health disorders have a high prevalence in the world population. With the coronavirus (COVID-19) pandemic, there was a requirement for social distancing, aggravating problems related to mental health and well-being. The proliferation of smartphones presents opportunities for data collection to study human behavior and health. They have been used in mental health studies, as they have several embbeded sensors that can capture measurements in people's daily lives. Traditionally, mental disorders are diagnosed by mental health professionals (e.g., psychiatrists, psychologists) on the basis of symptoms identified from patient interviews and self-reported experiences. However, patients often resort to events that occurred days, weeks or months ago, which can compromise the diagnosis, due to memory and desirability biases. To mitigate these biases, digital phenotyping appears, collecting data passively (without direct user interaction) using mobile devices. This work aims to present a framework to facilitate the development of digital phenotyping mobile applications, called *OpenDP*. The proposed solution is extensible and reusable, as it allows the inclusion of modules for processing raw context data, and can be applied to different mental disorders and to the monitoring of human well-being. In addition, the framework was developed using the middleware M-Hub/CDDL to collect data from sensors (physical and virtual) and distribute them among the internal components of the framework and with the broker external. Case studies were conducted to demonstrate that the proposed solution was able to compose digital phenotypes from the inference of high-level information generated by the data processing modules. It was also demonstrated the ability of the mobile solution to add data processing modules (plugins). By aiming at developing a mobile solution for mobile devices that does not cause a great impact on energy consumption, an experimental evaluation was carried out analyzing the impact on energy consumption of the smartphone user. The results were satisfactory, showing through the experimental evaluation that the battery consumption was small.

Keywords: Mental Health, Digital Phenotyping, Remote Monitoring, Sensing Applications, Framework.

Lista de ilustrações

Lista de tabelas

Lista de Siglas

API Application Programming Interface.

AWS Amazon Web Services.

BLE Bluetooh Low Energy.

CA Autoridade Certificado.

CDDL *Camada de Distribuição de Dados de Contexto.*

CEP Complex Event Processing.

DPMH *Digital Phenotyping Mental Health.*

EMA *Ecological Momentary Assessment.*

EMI *Ecological Momentary Intervention.*

EPL Event Processing Language.

FDD Frequência de Distribuição de Dados.

FGD Frequência de Geração de Dados.

GPS Global Positioning System.

IoMT Internet of Mobile Things.

IoT Internet of Things.

LGPL Lesser General Public License.

M-Hub Mobile Hub.

MEPA Mobile Event Processing Agent.

ML Machine Learning.

MQTT Message Queuing Telemetry Transport.

OMS Organização Mundial de Saúde.

QoC Quality of Context.

QoI Quality of Information.

QoS Quality of Service.

RSL Revisão sistemática da literatura.

S2PA Short-range Sensing Presence Actuation.

SecS Security Service.

SMS Short Message Service.

TLS Transport Layer Security.

WLAN Wireless Local Area Network.

WPAN Wireless Personal Area Network.

XML eXtensible Markup Language.

Sumário

1 Introdução

1.1 Contexto Geral

Os problemas de saúde mental têm alta prevalência na população mundial: 1 em cada 10 pessoas vive com um transtorno mental (DATTANI; RITCHIE; ROSER, 2021). Com o surgimento da pandemia de coronavírus (COVID-19), a população ficou mais exposta aos transtornos mentais, pois exigiu distanciamento social e ordens de permanência em casa (CONTROL; PREVENTION, 2021). Uma ferramenta tecnológica que está sendo utilizada em estudos na saúde mental e bem-estar são os *smartphones* (INSEL, 2017). Existiam cerca de 5 bilhões de dispositivos móveis em uso no mundo em 2020 (ASSOCIATION, 2021). Por exemplo, os *smartphones* foram usados no trabalho de Henson (HENSON *et al.*, 2021) através da plataforma *Beiwe* para prever recaídas de esquizofrenia, com base na detecção de anomalias de sociabilidade. Essa proliferação na adoção de *smartphones* apresenta oportunidades para a coleta de dados para estudar comportamentos e saúde de seres humanos (STRACZKIEWICZ; ONNELA, 2019).

Os transtornos mentais são geralmente diagnosticados por profissionais de saúde mental (e.g., psiquiatras, psicólogos) apenas com base nos sintomas identificados a partir de entrevistas com pacientes e experiências autor-relatadas em ambientes clínicos. No entanto, os pacientes costumam recorrer a eventos ocorridos dias, semanas ou meses antes das sessões, o que pode comprometer o diagnóstico. Quando os pacientes não lembram de eventos ou experiências prévias de forma precisa, trata-se viés de memória (GARCIA-CEJA *et al.*, 2018), ou quando mudam a verdade dos autorrelatos, consciente ou inconscientemente, ou omitem para alcançar um resultado socialmente desejável, trata-se do viés de desejabilidade (GARCIA-CEJA *et al.*, 2018). Além dessas limitações, o tratamento ocorre em ambientes clínicos, visto que o ambiente clínico difere do contexto natural das pessoas (TRULL; EBNER-PRIEMER, 2013). Uma forma de resolver essas limitações, é através da Fenotipagem Digital, do inglês *Digital Phenotyping Mental Health* (DPMH) (LIANG; ZHENG; ZENG, 2019), a qual visa coletar dados de forma passiva (i.e., sem interação direta do usuário) com uso de dispositivos ubíquos.

A fenotipagem digital é a "quantificação *in situ* momento a momento do fenótipo humano em um nível individual usando dados de *smartphones* e outros dispositivos digitais pessoais" (TOROUS *et al.*, 2016). As aplicações móveis de fenotipagem digital facilitam o diagnóstico, o tratamento de problemas de saúde, e o monitoramento do bem-estar a partir das interações das pessoas com as tecnologias digitais no ambiente cotidiano (e.g., em casa, no trabalho). Essas aplicações visam facilitar o monitoramento remoto e também identificação de problemas de saúde, pois há um processo de inferência dos

comportamentos e hábitos dos indivíduos, exigindo pouca ou nenhuma interação para a coleta de dados. Uma das áreas mais proeminentes usando a fenotipagem digital é a saúde mental, em que dispositivos móveis e vestíveis são usados para facilitar o diagnóstico e o tratamento de não somente transtornos mentais (e.g., depressão, esquizofrenia), mas também o bem-estar humano.

1.2 Caracterização do Problema

O problema central desta pesquisa consiste no fato de não haver um framework para o desenvolvimento de aplicações móveis de fenotipagem digital que seja extensível e reutilizável, ou seja, uma solução que permita monitorar uma ampla variedade de estados mentais, incluindo o monitoramento do bem-estar humano, e também uma solução que permita adaptar a outros estudos/pesquisas que está sendo feito ou o tipo de monitoramento requerido.

Outro problema que este trabalho vem solucionar é o fato de não haver soluções capazes de coletar dados de vários sensores e, a partir da inferência de informações de alto nível, caracterizar os comportamentos e hábitos de usuários, chamados neste trabalho de mestrado de fenótipos digitais.

1.3 Hipótese de Pesquisa

Este trabalho possui a seguinte hipótese: *é possível prover um arcabouço de software (i.e., um framework) reutilizável e extensível para facilitar o desenvolvimento de novas aplicações móveis de fenotipagem digital.*

1.4 Relevância do Trabalho

A saúde mental afeta grande parte da população mundial, independente de raça, sexo ou idade. Com o desenvolvimento de novas soluções de fenotipagem digital, elas utilizam dados gerados passivamente por dispositivos móveis, como *smartphones*, geram informações que podem caracterizar o comportamento e hábitos humanos, podendo resultar na diminuição do tempo de diagnósticos dos transtornos mentais, adequar o melhor tratamento específico para o paciente e monitoramento do bem-estar humano (HUCKVALE; VENKATESH; CHRISTENSEN, 2019).

Segundo a Organização Mundial de Saúde (OMS) (ORGANIZATION, 2004), estima-se que os transtornos mentais afetam 25% da população mundial em algum momento da vida. Os altos custos no tratamento da saúde mental, levam muitas pessoas a não procurarem um profissional de saúde mental. Com o desenvolvimento de soluções de

fenotipagem digital, com custos mais acessíveis, vem alcançar um maior número de pessoas a buscarem o diagnóstico, o tratamento do transtorno mental ou o monitoramento do bem-estar.

Outra relevância deste trabalho, é a solução pode servir de base para os desenvolvedores/pesquisadores/profissionais de saúde para construção/aprimoramento desta ou de novas ferramentas de fenotipagem digital. Por ser uma solução aberta, extensível e reutilizável, o desenvolvedor/profissional de saúde pode estender a outras demandas da saúde mental.

1.5 Objetivos

O objetivo geral desta pesquisa é conceber e implementar um framework para facilitar o desenvolvimento de aplicações móveis de fenotipagem digital.

Para tanto, consideram-se os seguintes objetivos específicos:

- Tornar o framework capaz de coletar dados de sensores virtuais;
- Implementar um mecanismo que adicione novos módulos de processamento de dados brutos de contexto, chamados *plugins*;
- Permitir que o framework seja extensível adicionando novos módulos de processamento de dados (*plugins*);
- Implementar o framework capaz de compor fenótipos digitais a partir de informações de alto nível geradas pelos módulos de processamento de dados;
- Testar a solução proposta em estudos de caso;
- Avaliar a solução proposta considerando como métrica o consumo de energia.

1.6 Contribuições Científicas e Tecnológicas

Considera-se que as principais contribuições científicas desta pesquisa são:

- A condução de uma revisão sistemática abrangente que apresentou o estado da arte existente sobre aplicações de sensoriamento para fenotipagem digital de saúde mental;
- Criação de uma ferramenta capaz de abstrair as necessidades dos profissionais de saúde de caracterizar os comportamentos sociais e hábitos dos usuários através da coleta de dados de contexto resultando na composição dos fenótipos digitais;

- Modelo arquitetural base para o desenvolvimento de aplicações de sensoriamento de dados de contexto para fenotipagem digital;
- Framework para facilitar o desenvolvimento de aplicações de fenotipagem digital, contribuindo para o cuidado de saúde mental e bem-estar humano.

Já as principais contribuições tecnológicas desta pesquisa são:

- Atualização do *middleware* M-Hub com a capacidade de coletar dados de sensores virtuais como: chamadas telefônicas, Short Message Service (SMS), toque de tela, tela ligada/desligada;
- A implementação de um framework que pode ser reutilizada e estendida para outras demandas;
- Um framework para criar aplicações móveis de fenotipagem digital que, por ser *Open Source*, pode ser atualizado e incrementado em novas versões.

1.7 Organização do Trabalho

O restante da dissertação está organizado da seguinte forma:

- O **capitulo 2** contém a fundamentação teórica, apresentando os conceitos relacionados à fenotipagem digital, além de apresentar as soluções CDDL e M-Hub.
- O **capitulo 3** contém os trabalhos relacionados, os quais são detalhados de maneira a apresentar aplicações de fenotipagem digital e suas características, bem como analisados comparativamente.
- O **capitulo 4** apresenta os detalhes da solução proposta, incluindo uma visão geral do framework implementado, os componentes e sua arquitetura.
- O **capitulo 5** contém os estudos de caso e a avaliação do consumo de energia.
- No **capitulo 6** são descritas as considerações finais, as limitações do trabalho e as publicações.

2 Fundamentação Teórica

Este capítulo contém a fundamentação teórica, abordando conceitos de fenotipagem digital, quando o termo surgiu, sua taxonomia, como funciona o processo de fenotipagem digital desde a coleta e processamento dos dados até a detecção dos padrões comportamentais. Este capítulo também apresenta conceitos sobre o *middleware* CDDL (GOMES *et al.*, 2017) e M-Hub (GOMES *et al.*, 2016; SILVA *et al.*, 2020).

2.1 Fenotipagem Digital

Antes de explicar o termo "fenotipagem digital" neste trabalho, é importante saber como ele surgiu. Cerca de duas décadas atrás, os cientistas Andy Clarck e David Chalmers (CLARK; CHALMERS, 1998) propuseram a noção de mente estendida. Para os autores, a separação entre mente, corpo e ambiente é, em princípio, inseparável, enquanto objetos do mundo externo podem fazer parte de processos cognitivos ou da identidade pessoal e, dessa forma, funcionam como elementos integrantes do funcionamento mental, tornando-se um único elemento (mente, corpo e ambiente). É exatamente o que testemunham no cotidiano atual com o uso cada vez mais intenso dos dispositivos digitais que mediam a relação com o ambiente e com os outros.

A ideia de mente estendida tem origem feita por Richard Dawkins (PASZTOR, 1991) anos antes dos cientistas Andy Clarck e David Chalmers começar a falar sobre o assunto. Richard Dawkins apresenta o conceito de fenótipo estendido: a ideia de que fenótipos não deveriam ser vistos apenas como processos biológicos, mas estendidos a todos os efeitos que um gene tem sobre seu ambiente, dentro ou fora do organismo individual. No exemplo que ele fala dos diques construídos por castores comporiam seu fenótipo estendido. Os castores constroem diques para criar reservatórios de água profundos onde podem se abrigar de predadores, fazendo fluir o próprio alimento (i.e., da casca, dos ramos e das folhas das árvores) e os materiais de construção que utilizam. O roedor faz uma represa num trecho de rio, criando um lago artificial. Em seguida, constrói perto dali sua toca, usando galhos, troncos, lama e pedras. Ele tem hábitos noturnos e usa os dentes para cortar a madeira com que recobre sua toca.

Em 2015, um artigo publicado por Jain (JAIN *et al.*, 2015), propõe uma ampliação desse conceito de Dawkins (PASZTOR, 1991), chamando fenótipo digital. Com o crescimento e evolução dos dispositivos digitais (e.g., relógios, *smartphones*), a interação dos indivíduos entre si e com os dispositivos digitais produz de forma constante uma massa de dados, podendo conter informações sobre comportamentos, hábitos dos usuários que pode contribuir do diagnóstico ao tratamento, passando pela prevenção, detecção de fatores

de risco e monitoramento do bem-estar das pessoas. O rastreamento contínuo e em larga escala dos dados produzidos de forma passiva, constante e remota pode permitir sobretudo a detecção precoce de problemas, identificando sintomas muito antes de sua expressão fenotípica clássica, e potencialmente abrindo caminho para intervenções terapêuticas mais rápidas e bem dirigidas (BEZERRA BENILTON, 2020).

Em 2016, o termo fenotipagem digital foi introduzido por Torous (TOROUS *et al.*, 2016), e ganhou destaque com o desenvolvimento de diversas plataformas, tais como o *Beiwe* (TOROUS *et al.*, 2016) e o *StudentLife* (WANG *et al.*, 2014), as quais têm foco em saúde mental. A literatura apresenta uma variedade de estudos apontando que a área da saúde mental pode se beneficiar da fenotipagem digital (MENDES *et al.*, 2022; MOURA *et al.*, 2020). Como já informado neste trabalho, a fenotipagem digital é definida formalmente como a "quantificação momento a momento do fenótipo humano ao nível individual *in situ* usando dados de dispositivos digitais pessoais" (TOROUS *et al.*, 2016).

2.1.1 Caracterizando a Fenotipagem Digital

Segundo o Coghlan et al. (COGHLAN; D'ALFONSO, 2021), analisando a etimologia da palavra "Phenotype", *pheno* significa manifestação, o fenótipo de um organismo são suas características biológicas. Um "fenótipo da doença" é a manifestação de uma doença, enquanto o "endótipo" é o mecanismo da doença. Tal endótipo não pode ser diferenciado do fenótipo, mas está oculto, e requer esclarecimento através de investigação e análise. De maneira análoga, dados digitais derivados de vários dispositivos digitais estão ligados às condições e estados comportamentais humanos. Determinar exatamente quais sinais de dados digitais (o que, pode-se dizer) requer investigação e análise, é aí que entra a fenotipagem digital.

A característica mais promissora da fenotipagem digital é a possibilidade de medições contínuas. O uso de aplicações, chamadas telefônicas, velocidade de digitação e recursos de voz podem ser monitorados discretamente a cada segundo ao longo da vida, com algoritmos em tempo real. Outra característica é usar dados ativos e passivos. Dados ativos são obtidos por entrada direta de usuários em resposta a solicitações para esses dados. Por exemplo, usuários que respondem a um *prompt* para pesquisa. Dados passivos incluem dados não solicitados recebidos de sensores. Por exemplo, sensores incorporados em dispositivos onipresentes como acelerômetro, rastreamento por Global Positioning System (GPS).

2.1.2 Taxonomia da Fenotipagem Digital

A fenotipagem digital pode envolver diversas inferências lógicas e várias relações existentes entre dados e informações, por um lado, e propriedades psicológicas, por

outro (COGHLAN; D'ALFONSO, 2021). Os sinais de dados derivados de dispositivos e sensores podem ser usados para fazer previsões e determinações sobre indivíduos e grupos. Tais determinações podem ocorrer de diferentes maneiras. É importante frisar que os dados digitais capturados são associados e usados para derivar recursos de informações (*information features*). Por exemplo, os dados de GPS podem gerar informações sobre frequência e a duração das visitas de alguém a vários locais. Dados digitais e recursos de informações podem ser usados para inferir propriedades psicológicas e vice-versa. As propriedades psicológicas incluem personalidades humanas, humores, comportamentos, sentimentos, hábitos.

A Figura 1 mostra uma taxonomia de possibilidades de fenotipagem digital que dizem respeito à determinação de propriedades psicológicas definidas por Coghlan et al. (COGHLAN; D'ALFONSO, 2021). A taxonomia identifica os seguintes cenários e caraterísticas: (a) propriedade psicológica causa recursos de dados/informações; (b) recursos de informações causam propriedade psicológica; (c) recursos de dados/informações estão correlacionadas com propriedades psicológicas; e (d) recursos de informações constituem propriedade psicológica.

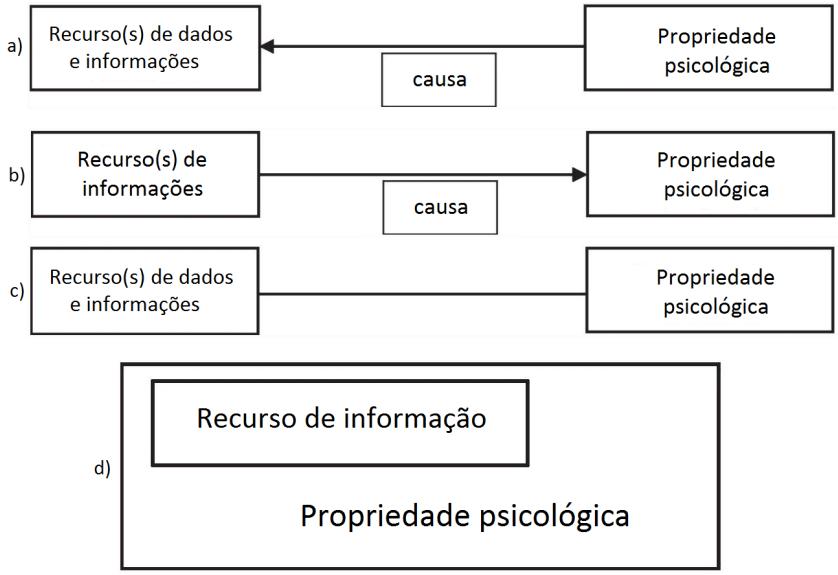

Figura 1 – Tetra-taxonomia da Fenotipagem Digital (COGHLAN; D'ALFONSO, 2021).

(a) A propriedade psicológica afeta causalmente o(s) recurso(s) de dados/ informações: nesse primeiro cenário a direção causal vai da propriedade psicológica à recurso de informação. Por exemplo, a propriedade psicológica (insônia) pode causar o recurso de informação (uso repentino do smartphone à meia-noite). Haverá, sem dúvida, geralmente mais de uma causa possível para um recurso de informação. O objetivo ao fazer essa inferência é coletar o máximo de informações relevantes de dados/informações para a propriedade psicológica que o causou. No entanto, às vezes a conexão causal que vai da propriedade ao recurso pode ser forte ou confiável

o suficiente para fazer inferências com certeza prática. Por exemplo, se um certo distúrbio psicomotor (quase) sempre causa um certo padrão de interação com a tela do *smartphones*, a ausência desse padrão nos dados de interação da tela do usuário implicaria (dedutivamente) que o usuário não possui esse distúrbio;

(b) Recurso(s) de informação afetam causalmente a propriedade psicológica: nessa segunda possibilidade, a direção causal é invertida. Aqui, o recurso de informação ou o que ele representa causa a propriedade. Por exemplo, a maneira como alguém usa um determinada aplicação pode ter um efeito causal em seu estado psicológico. O recurso de informação extraído dos dados neste exemplo é a maneira e a extensão do uso do smartphone, e é isso que causa (ou afeta causalmente) a propriedade psicológica. Assim, os recursos de dados/informações podem, não ser apenas "manifestações de doenças biológicas", mas às vezes causas delas (JAIN *et al.*, 2015);

(c) Existem correlações entre recurso de informação e propriedades psicológicas: nessa terceira possibilidade, há a presença de correlação, mas nenhuma causalidade direta entre a recurso de informação e a propriedade psicológica. Por exemplo, muitas pessoas que vivem em uma determinada área podem ter uma determinada condição. Se essa condição seja realmente causada por algum outro fator, digamos, envenenamento por chumbo da água. Viver na área não causa (diretamente) a condição, e ter a condição não causa viver na área. No entanto, os dados de geolocalização podem ser usados para inferir que esses indivíduos têm a condição;

(d) Recurso de informação são constitutivas de propriedade psicológica: nessa quarta possibilidade, os recursos de informações são ou representam o próprio comportamento, ou estado psicológico. Aqui, certas condições podem ser definidas por recursos de informações extraídas de dados digitais. Por exemplo, suponha-se que o uso problemático de *smartphones* fosse um distúrbio. Esse distúrbio pode então ser medido e diagnosticado, quantificando o uso do *smartphones* determinando um limite (tempo) de uso como condição, dado que esse limite de uso seja excedido indicaria a presença do distúrbio. Outro exemplo envolve testes cognitivos ou psicomotores (e.g., jogo eletrônicos (WISNIEWSKI; HENSON; TOROUS, 2019)) Aqui, certas formas de teste (dados ativos) podem ser criadas em dispositivos de *smartphones*, de modo que, ter uma determinada condição é parcialmente definida pelos dados resultantes que eles representam.

2.1.3 Processo de Fenotipagem Digital

O processo de fenotipagem digital organizado em camadas pode ser observado na Figura 2.

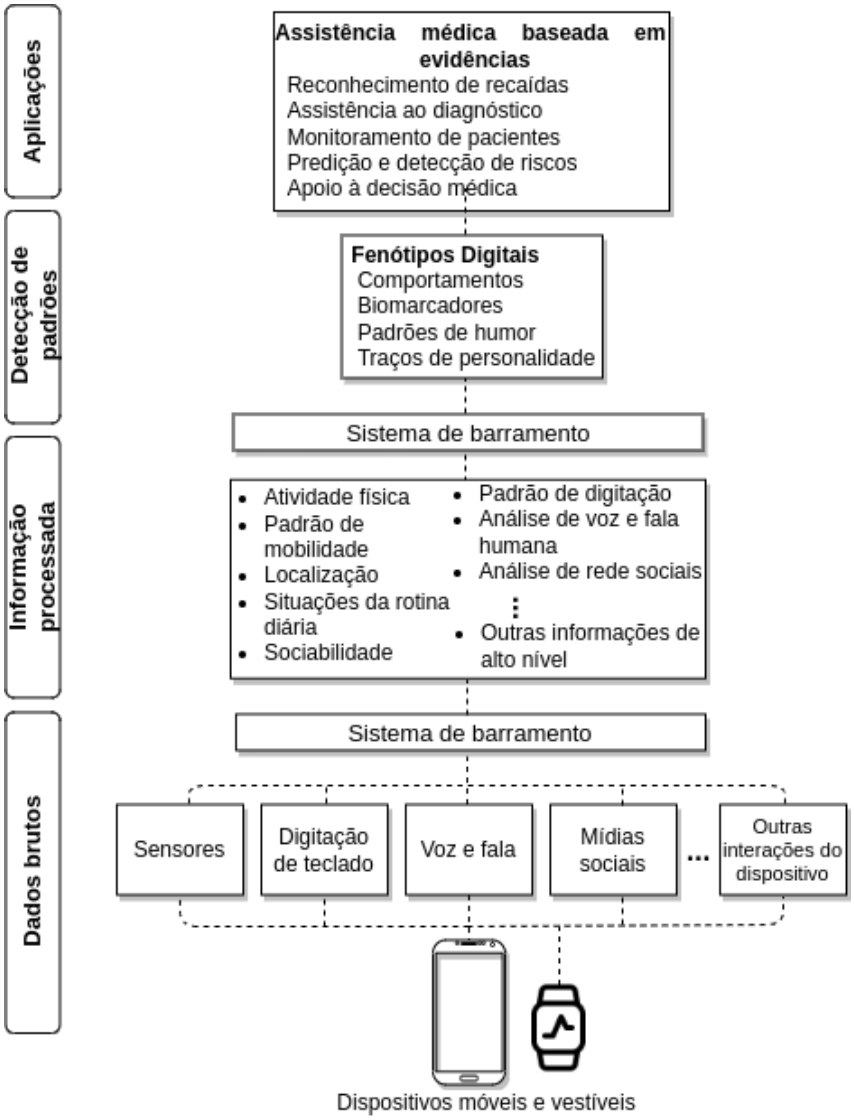

Figura 2 – O Processo de fenotipagem digital adaptado de (MENDES *et al.*, 2022).

2.1.4 Camada de Dados Brutos

O processo começa na primeira **camada de dados brutos** com a coleta de dados obtidos por diferentes fontes (e.g., sensores como GPS, entradas de teclado e dados de mídia social) em dispositivos móveis e vestíveis. Esses dados podem ser coletados ativamente, em que as entradas do usuário são explicitamente solicitadas, e de forma passiva, exigindo apenas a permissão do usuário para acessar os dados de contexto.

Na camada de dados brutos, os dispositivos móveis e vestíveis apresentam milhares de sensores, os quais vêm usualmente instalados de fábrica. Os sensores são utilizados para obter informações diárias dos usuários, de modo a encontrar um padrão comportamental, na literatura classificam em sensores físicos e virtuais. Em (MENDES *et al.*, 2022) define sensores físicos como componentes de *hardware* incorporados ou conectados a dispositivos responsáveis por coletar dados de contexto. Alguns exemplos são acelerômetros para medir atividade do usuário, sensores de luz para medir os níveis de luz ambiente e GPS

para coletar as localizações do usuário. Sensores virtuais (MENDES *et al.*, 2022) são definidos como componentes de *software* capazes de registrar interações de indivíduos com dispositivos ou usando vários sensores físicos (ou outros virtuais) para construir um recurso de nível superior. Exemplos de tais sensores são os sensores de interação social que pode usar encontro *Bluetooth* (ou seja, informações de co-localização entre indivíduos ou lugares), rede Wi-Fi.

Nesse trabalho, foi usado a classificação de Palaghias (PALAGHIAS *et al.*, 2016) para categorizar os sensores, abaixo está as categorias:

- Ambiente: microfone, câmera, magnetômetro;
- Posicionamento: *Bluetooth*, sinal de torre celular, GPS, Wi-Fi;
- Inercial: Acelerômetro, giroscópio;
- Virtual: Chamadas telefônica, SMS.

2.1.5 Camada de Informações Processadas

A medida que os dados de contexto vão sendo coletados, quando passam pela **camada de informações processada**, eles são processados com a intenção de gerar informações de alto nível. Para exemplificar este processo, a solução *SituMan* (TELES *et al.*, 2017) utiliza dados de acelerômetro e coordenadas para inferir situações de rotina diária do usuário (e.g., trabalhando, estudando). Outro exemplo é a aplicação móvel *eB2* (BERROUIGUET *et al.*, 2018), que identifica alterações nos padrões de mobilidade dos usuários que sofrem de transtorno de humor e depressão com base nos dados de localização. Em algumas categorias de sensores, por exemplo, o acelerômetro, é necessário atribuir uma taxa de amostragem ou filtrar os dados, pois gera uma grande massa de dados.

2.1.6 Camada de Detecção de Padrões

Na **camada de detecção de padrões**, de posse das informações de alto nível geradas pela camada de informações processadas, o desenvolvedor recorre a ferramentas computacionais (e.g., aprendizado de máquina, mineração de dados), modelos estatísticos para detectar padrões comportamentais, biomarcadores, de humor e traços de personalidade. A medida que as informações de alto nível são agrupadas, tem-se um conjunto de fenótipos digitais. O fenótipo digital representa o comportamento social diário do usuário em seu ambiente natural, que vai servir para dar apoio ao profissional de saúde no diagnóstico ao tratamento até o monitoramento do bem-estar das pessoas. Por exemplo, o trabalho de Berrouiguet (BERROUIGUET *et al.*, 2018) identificou alterações nos padrões de mobilidade

dos usuários que tinham transtornos de humor e ansiedade com base nos dados dos sensores do *smartphone* (e.g., localização, chamadas telefônicas, SMS) capturados pela aplicação *eB2* (BERROUIGUET *et al.*, 2018), processados por ferramentas computacionais.

2.1.7 Camada de Aplicação

Por fim, observa-se na **camada de aplicação**, o papel de identificar os fenótipos digitais para encontrar evidências de transtornos mentais ou o monitoramento do bem-estar das pessoas. Neste contexto, o objetivo das aplicações é absorver os fenótipos digitais que vão sendo construídos para dar assistência médica com reconhecimento de recaídas, assistência ao diagnóstico, monitoramento de usuários, predição e detecção de riscos e apoio à decisão médica. Existem vários trabalhos com intenções de fornecer apoio a decisão médica. O *Beiwe* (TOROUS *et al.*, 2016) e o *Lamp* (WISNIEWSKI; HENSON; TOROUS, 2019) são exemplos de plataformas que focam em vários transtornos mentais sendo utilizadas em pesquisas científicas em fenotipagem digital.

2.1.8 O Cenário da Pandemia do Coronavírus (COVID-19)

O surto de infecção pelo coronavírus (COVID-19) entre os humanos pelo mundo tem impactado drasticamente a saúde global e saúde mental (TORALES *et al.*, 2020). Um dos efeitos dessa pandemia é o isolamento social, com mudanças no padrão de sociabilidade dos indivíduos, dado que as pessoas devem se manter afastadas fisicamente na tentativa de reduzir o avanço da contaminação do vírus. Este surto leva a problemas como estresse e sintomas depressivos causados pelo isolamento social (TORALES *et al.*, 2020). Com o isolamento social, houve mudanças na maneira das pessoas se socializarem, com poucas interações físicas entre elas. Por outro lado, aumentaram as interações virtuais (e.g., videoconferência e uso de redes sociais *online*).

As aplicações de fenotipagem digital podem ser capazes de monitorar e identificar padrões de sociabilidade e proximidade física das pessoas também nesse contexto de pandemia. Dessa forma, elas podem ajudar a monitorar os usuários para evitarem contaminação com o vírus e também os profissionais na linha de frente (Sear *et al.*, 2020), tem enfrentando uma alta carga de estresse em decorrência do excesso de trabalho.

2.2 CDDL

O CDDL (GOMES *et al.*, 2017) é um *middleware* com recursos para processamento e distribuição de dados de contexto, que oferece suporte ao desenvolvimento de aplicações de Internet of Things (IoT)/Internet of Mobile Things (IoMT). Ele é utilizado na solução proposta para criar um canal de comunicação entre os componentes do framework e um canal de comunicação e distribuição de dados com o servidor.

O *middleware* possui três versões: móvel (*android*), *desktop* e *web*, conforme pode ser obervado na Figura 3. A Figura 3 mostra às três formas do CDDL interagindo no ambiente para publicar e subscrever os dados.

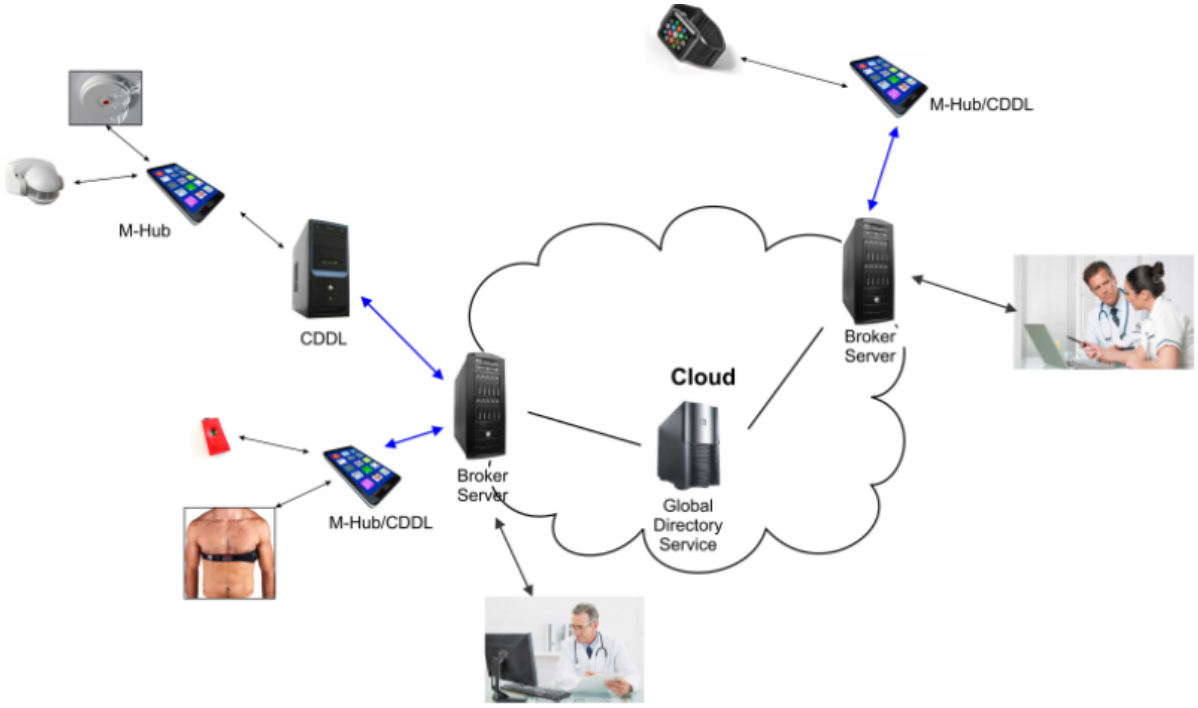

Figura 3 – CDDL (GOMES *et al.*, 2017).

O CDDL é responsável por promover um mecanismo de comunicação *publisher-subscriber* baseado em tópicos. Além disso, ele provê a comunicação entre aplicações consumidoras e objetos inteligentes descoberto pelo M-Hub. Ele usa o protocolo Message Queuing Telemetry Transport (MQTT) (CONTRIBUTORS, 2022), permite a publicação de dados de contexto. Ele disponibiliza uma Application Programming Interface (API) para o desenvolvimento de aplicações de IoT/IoMT.

A grande vantagem de usar o *middleware* CDDL em aplicações IoT/IoMT são os recursos que Quality of Context (QoC)/Quality of Service (QoS)/Quality of Information (QoI) fornece. No trabalho de Gomes (GOMES *et al.*, 2017), mostra QoC como sendo qualquer informação que descreva o QoI sendo usado como informação de contexto. Tem-se então que o QoI define um conjunto inicial (ou padrão) de atributos que fazem parte do seu modelo de contexto, mais precisamente do modelo de mensagem que o CDDL pode publicar. O CDDL possui uma classe *Message* usada na distribuição de dados, carrega informações de acurácia, confiança, localização de fonte (latitude/longitude), tempo de medição, tempo de chegada, tempo de validade, idade, intervalo de medição, lista de atributos, atributos disponíveis, completude, resolução, atraso e tempo total de entrega. O QoS define um conjunto extensível de parâmetros e políticas. Uma política de QoS é um mecanismo transparente que controla o envio e o recebimento de dados para aplicações produtoras

e consumidoras de informações. Por um lado, os parâmetros de QoS informados pelo produtor controlarão como os dados são enviados para o broker. Por outro, os parâmetros de QoS informados pelo subscritor vão controlar a maneira como os dados são recebidos do broker e entregues para a camada de aplicação. Os parâmetros e políticas definidos pelo QoS são: prazo de entrega, taxa de atualização, controle de latência, histórico, ordem de destino, vida útil, retenção, vivacidade, confiabilidade e sessão.

O CDDL dá assistência a políticas de QoS de distribuição dos dados, podendo especificar diversos parâmetros de QoC, tanto para o envio como para o recebimento dos dados. Além disso, toda mensagem publicada pelo CDDL utiliza metadados de QoC. Outra funcionalidade do CDDL é a descoberta de serviços através do uso de diretórios de serviços (*Global Directory Service*). As aplicações podem consultar estes diretórios especificando requisitos de QoC em seus critérios de busca. Um mecanismo de filtragem dos dados é fornecido com o qual as aplicações podem especificar filtros para os dados publicados e/ou recebidos, de modo a diminuir a quantidade de dados processados pela aplicação (e.g., quando um *subscriber* não suporta o grande massa de dados publicada pelo *publisher*). Por fim, o *middleware* fornece componentes para o monitoramento da QoC durante a execução das aplicações, possibilitando que as aplicações consumidoras das informações de contexto possam receber diversas notificações sobre a variação da QoC como, por exemplo, serem notificados quando seus requisitos de QoC não puderem ser mais cumpridos pelo provedor de serviço.

2.2.1 Arquitetura

A Figura 4 mostra a arquitetura do CDDL e seus componentes. O CDDL disponibiliza uma *interface* utilizada por aplicações, através dela é gerenciado o CDDL.

Os componentes do CDDL são descritos abaixo.

- *Publisher*: componente responsável pela publicação dos dados, de origem do Short-range Sensing Presence Actuation (S2PA) ou da camada de aplicação. O *publisher* fornece recursos de publicação de consultas como, por exemplo, utilizando o suporte Event Processing Language (EPL), a aplicação pode consultar (dentre qualquer publicador) quais os serviços de localização estão disponíveis e cuja idade das informações seja no máximo de 10 segundo. Além disso, o *publisher* entrega os dados ao *broker* de acordo os parâmetros e políticas de QoS;

- *Subscriber*: componente responsável pela subscrição em tópicos. O *subscriber* fornece recursos para receber consultas como, por exemplo, o *subscriber* caso não conheça de antemão o identificador do publicador, pode receber este identificador através de uma consulta aos diretórios de serviços, qualquer informação que o publicador

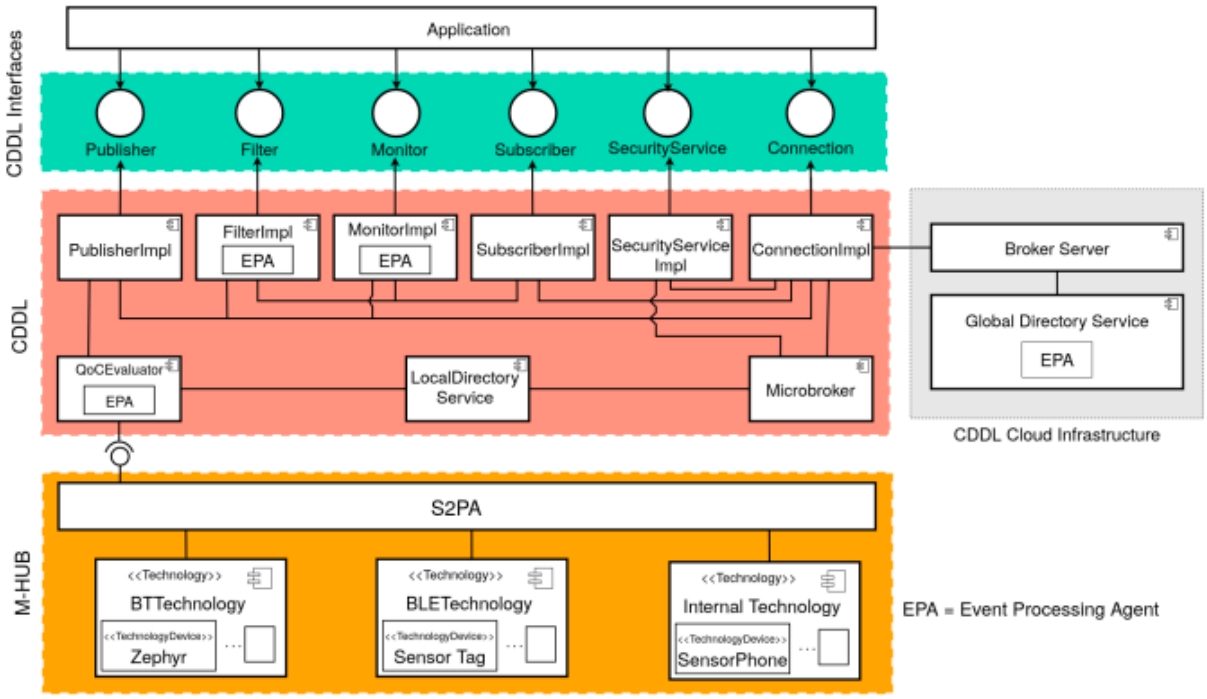

Figura 4 – CDDL integrado ao M-Hub (GOMES *et al.*, 2017).

estiver disponibilizando serão assinadas. Além disso, o *subscriber* entrega os dados à camada de aplicação é feita através de parâmetros e políticas de QoS;

- *Local Directory Service*: componente responsável pelo gerenciamento dos registros dos provedores de QoS como, por exemplo, os tipos de contexto disponíveis, nível da QoS de distribuição oferecidos pelos provedores;
- *Connection*: componente responsável por estabelecer conexões locais (i.e., *Micro broker*) ou remotos (i.e., *broker*);
- *Micro Broker*: a função de um *broker* é intermediar a comunicação entre publicadores e subscritores. O *Micro broker* é a versão mais simples (menos componentes) do *broker server* voltado para dispositivos móveis da plataforma Android. O *micro Broker* é configurado para aceitar conexões locais. Desse forma, é possível haver troca de mensagens entre aplicações que executam no mesmo dispositivo, mas também podem funcionar se estiverem conectados na mesma Wireless Local Area Network (WLAN);
- *QoC Evaluator*: componente que recebe dados de contexto enriquecidos com metadados de QoI de origem do S2PA. O *QoC evaluator* adicionar valores de alguns parâmetros de QoI, que não são preenchidos na fase de aquisição como, por exemplo, qualidade média dos dados de contexto para cada sensor;
- *Filter*: componente que filtra os dados no conteúdo de seus atributos, podem ser utilizados pelo *publisher*, *subscriber* e pelo *broker* com a filtragem por tópicos. Por

exemplo, uma aplicação pode usar o *filter* para filtrar os dados recebidos pelo *subscriber* somente com acurácia de 50%;

- *Monitor*: componente responsável por criar regras (EPL) para o monitoramento de eventos, utilizados pelos *publisher* e *subscriber*. Por exemplo, pode uma aplicação ao utilizar o *monitor* para criar uma regra em EPL no *subscriber* para monitorar os dados recebidos com acurácia 80%, caso a condição seja satisfeita, gere uma notificação na camada de aplicação;

- *Server Broker*: componente responsável por intermediar a comunicação entre publicadores e subscritores à nuvem. Diferente do *micro broker* executado em dispositivos móveis, o *server broker* executado em computador e possui mais capacidade de distribuição de dados;

- *Global Directory Service*: componente responsável pelo registro dos provedores de serviços conectados a nuvem, e atualizada a partir das listas de serviços publicada pelo diretório local;

- *Security Service*: componente responsável por habilitar o estabelecimento de canais seguros de comunicação, com autenticação e controle de acesso aos dados. Além disso, este componente também gerencia as chaves criptográficas a serem utilizadas, gera requisições de assinatura de certificado digital, armazena certificados digitais e mantém uma lista de controle de acesso.

2.2.2 Características

O objetivo ao utilizar o CDDL considera a necessidade de incorporar mecanismos de gerenciamento e QoC, de modo que o programador possa se concentrar na implementação de requisitos inerentes à lógica de negócio das aplicações. O resultado disso é a redução da complexidade.

As funcionalidades atendidas pelo CDDL são:

- *Distribuição de dados local e remota*: provê mecanismo de distribuição de dados de contexto, de modo a possibilitar o compartilhamento de informações entre produtores e consumidores de contexto. O *middleware* provê suporte tanto a distribuição local (produtores e consumidores executando no mesmo dispositivo) como a distribuição remota (produtores e consumidores executam em dispositivos distintos);

- *Registro e descoberta de serviço*: mecanismo que permite descrever as características dos serviços de contexto disponíveis, de modo que possam ser registradas em diretórios locais, que armazenam dados de serviços disponíveis no mesmo dispositivo, ou globais,

que armazenam na nuvem dados de serviços pertencentes a um ou mais produtores de um domínio;

- *Modelo de programação uniforme e independente de localização*: independentemente da localização dos produtores, consumidores, diretórios de serviços e *brokers*, sejam locais ou remotos, as *interfaces* de programação utilizadas para registro, publicação, consulta e subscrição de dados de contexto devem ser uniformes, ou seja, a maneira de programar uma aplicação que consome dados de contexto de origem local deve ser a mesma para desenvolver uma aplicação remota;

- *Entrega confiável de dados em cenários de mobilidade*: de modo a abranger também a distribuição de informações de contexto em cenários de mobilidade irrestrita, o CDDL provê mecanismo que minimizem a perda de dados cuja entrega não foi confirmada, eventualmente devido a problemas relacionados à conectividade fraca ou intermitente dos *gateways* locais ou aplicações consumidoras com a Internet;

- *Provisionamento e monitoramento de QoI*: permite inserir na informação de contexto valores (metadados de QoI) que descrevem a qualidade dos dados, podendo ser no momento da aquisição dos dados (i.e., S2PA) ou pelo *QoC evaluator*;

- *Provisionamento e monitoramento de QoS de distribuição*: permite configurar políticas que definem a QoS de distribuição dos dados de contexto feito pelos publicadores e subscritores, podendo controlar como as informações de contexto serão publicadas, recebidas e armazenadas.

2.2.3 Segurança com CDDL

A segurança computacional é um tema que deve ser considerado ao desenvolver aplicações, especialmente quando tais aplicações envolvem o consumo e a distribuição de dados sensíveis. Como exemplo, podemos citar as aplicações de fenotipagem digital, que podem lidar com dados de saúde coletados remotamente de usuários, a partir de sensores vestíveis. Para adicionar serviços de segurança em ambientes de produção, deve-se haver formas de garantir algumas propriedades básicas de segurança, tais como: confidencialidade, integridade, autenticação e controle de acesso (MARTINS; SAUKAS; ZANARDO, 2004).

Devido à necessidade de desenvolver-se aplicações de IoT/IoMT seguras, foi adicionado ao CDDL uma camada de segurança denominada de Security Service (SecS) (CARDOSO *et al.*, 2020a; CARDOSO *et al.*, 2020b). Na Figura 4, mostra o componente *SecurityServiceImpl*, que implementa o serviço de segurança. De maneira geral, o SecS é responsável por gerar chaves criptográficas, gerar requisições de assinatura certificado digital, importar certificados digitais e manter uma lista de controle de acesso. Como o CDDL utiliza internamente o MQTT como protocolo de comunicação, sua arquitetura de

segurança foi pensada para aproveitar recursos de segurança já suportados pelo MQTT. Dessa forma, o CDDL utiliza o Transport Layer Security (TLS) e listas de controle de acesso para garantir o estabelecimento de canais seguros de comunicação. Também mostra uma comunicação direta entre o *SecurityServiceImpl* e os componentes *ConnectionImpl* e *Micro broker*. Portanto, também é responsabilidade do SecS disponibilizar as chaves criptográficas e certificados digitais para os dois componentes do CDDL que estabelecem de fato as conexões (*ConnectionImpl* e *Micro broker*).

O objetivo da camada de segurança do CDDL é fornecer mecanismos para o estabelecimento de comunicação segura, autenticação e controle de acesso. Durante o desenvolvimento do SecS, consideraram-se as seguintes políticas de segurança.

- Uma instância do CDDL deve manter dois certificados digitais, um para o usuário e outro para a autoridade certificadora;
- A troca de mensagens deve utilizar criptografia;
- A autenticação deve ser mútua em uma conexão;
- O controle de acesso deve, por padrão, recusar todas as mensagens, exceto se a permissão seja garantida pelo usuário;
- Por fim, o uso do *middleware* com os recursos de segurança deve ser opcional, permitindo a não utilização do SecS.

A confidencialidade, integridade e autenticidade na troca de dados é garantida utilizando o TLS. Além disso, o controle de acesso é feito através de uma lista de controle de acesso, que contem quais usuários podem publicar ou se subscrever em quais tópicos. Combinando o TLS com o controle de acesso, torna-se possível garantir o canal seguro de comunicação, habilitando, assim, o desenvolvimento de aplicações seguras com o *middleware* M-Hub/CDDL.

A confiança entre as partes comunicadoras, durante a fase de autenticação, é estabelecida através do certificado da autoridade certificadora. Cada instância do CDDL deve utilizar o SecS para importar o certificado do usuário e também importar o certificado da autoridade certificadora considerada cofiável pelo usuário. A conexão ocorre entre cliente e *micro broker*, ou cliente e *broker*, apenas ocorre caso ambos apresentem certificados digitais assinados pela autoridade certificadora.

2.3 M-Hub

O *Mobile Hub (M-Hub)* (GOMES *et al.*, 2016; SILVA *et al.*, 2020) é um serviço de *middleware* que executa em dispositivos móveis, responsável pela descoberta, conexão

e aquisição de dados de objetos inteligentes (e.g., sensores, atuadores, relógios, robôs, veículos) próximos e acessíveis a partir de tecnologias Wireless Personal Area Network (WPAN) de curto alcance, como *Bluetooth Classic* e Bluetooh Low Energy (BLE), ou que estejam embutidos no próprio dispositivo móvel. Ele é utilizado na solução proposta para coletar dados físicos e virtuais dos dispositivos móveis. Portanto, o M-Hub é um *gateway*, que serve de ponte entre os objetos inteligentes e a *Internet*. O M-Hub é integrado ao CDDL, ela realiza a aquisição dos dados e repassa para o CDDL distribuir.

2.3.1 Arquitetura

A Figura 4 mostra o M-Hub integrado ao CDDL, a qual é composta por diversos serviços e gerenciadores locais, todos executando em segundo plano e paralelamente com as aplicações móveis do usuário. Segue abaixo os componentes do M-Hub:

- *LocationService*: responsável por coletar a localização do dispositivo móvel e adicionar a todas as mensagens publicadas pelo M-Hub;
- *S2PA*: responsável pela descoberta, monitoramento e registro de sensores e atuadores;
- *EnergyManager*: define as frequências da busca por novos sensores e da consulta ao serviço de localização a partir do monitoramento diário do nível da bateria do dispositivo móvel, com objetivo de minimizar o consumo de energia;
- *Mobile Event Processing Agent (MEPA)*: fornece um motor de inferência *Esper*/Complex Event Processing (CEP) criando regras para avaliar os fluxos de dados dos sensores. O *Esper* possui uma EPL para implementar a regra responsável por avaliar os fluxos de dados dos sensores;
- *BT Technology*: responsável pela integração com dispositivos que se comunicam por meio das tecnologias WPAN *Bluetooh Classic*;
- *BLE Technology*: responsável pela integração com dispositivos *Bluetooh Low Energy*;
- *Internal Technology*: responsável pela coleta de dados de sensores embarcados no *smartphone* (e.g., GPS, acelerômetro, magnetrômetro) com Sistema Operacional Android.

2.3.2 Características

Pode-se observar as seguintes funcionalidades abaixo:

- *Descoberta de sensores próximos*: diariamente, o M-Hub procura objetos inteligentes próximos que anunciam seus identificadores. Estas informações sobre objetos inteligentes alcançáveis são mantidas no banco de dados do M-Hub;

- *Conexão com sensores*: estabelecer um link de comunicação com o sensor conforme o protocolo WPAN utilizado, após a conexão o M-Hub passa a receber os dados dos sensores;
- *Protocolo de transcodificação*: o M-Hub transcodifica e serializa os dados de origem de sensores conforme o tipo, marca e fabricante do sensor;
- *Configuração e controle dos atuadores*: o M-Hub pode enviar comandos através da WPAN para os atuadores conectados de modo a gerenciá-los;
- *Pré-processamento de dados do sensor*: o M-Hub permite aplicar uma função de pré-processamento (e.g., agregação, formatação, filtragem) antes de os dados serem enviado;
- *Processamento na borda*: permite distribuír parte do processamento das informações de contexto para a "borda" da rede, diminui a quantidade de informações a serem transmitidas para a nuvem, e ganha em escalabilidade do sistema;
- *Processamento ciente de energia*: o M-Hub monitora o nível de bateria do dispositivo móvel de modo a liberar ações que coordena as ações dos serviços como, por exemplo, reduzir a frequência da publicação dos dados caso o nível da bateria esteja baixa.

3 Trabalhos Relacionados

Para caracterizar o estado da arte da pesquisa com os trabalhos relacionados ao desenvolvimento de novas aplicações de sensoriamento para fenotipagem digital, foi realizada uma RSL publicada em (MENDES *et al.*, 2022). Este capítulo apresenta, inicialmente, um resumo da metodologia usada para a condução da revisão. Em seguida, descreve os resultados encontrados, apresentando as aplicações que se destacaram e por possuir características mais próximas da solução proposta.

3.1 Condução de uma RSL

A RSL é uma metodologia que fornece meios para identificar, avaliar e interpretar o estado da arte em uma área de pesquisa, respondendo a questões de pesquisa de interesse (KITCHENHAM; CHARTERS, 2007). Esses métodos sistemáticos utilizados para selecionar, avaliar, coletar e analisar pesquisas relevantes precisam seguir procedimentos validados presentes na literatura para aumentar a integridade de revisão realizada e permitir que outros pesquisadores reproduzam os métodos adotados (MONITORAMENTO..., 2020). Assim, este estudo foi realizado com base nas diretrizes para RSL em Engenharia de Software propostas por Kitchenham e Charters (KITCHENHAM; CHARTERS, 2007). Essa revisão seguiu três fases principais: planejamento, condução e divulgação dos resultados. A realização dessas fases foi apoiada pelo *Parsifal*[1], uma ferramenta de suporte *online* para executar RSL no contexto da Engenharia de Software, que documenta os processos executados e fornece suporte aos pesquisadores através de um espaço de trabalho compartilhado.

O objetivo da RSL é fornecer uma caracterização técnica e resumo de aplicações de sensoriamento e *datasets* públicos para DPMH. Por "público" entende *datasets* que estão disponíveis para baixar para uso gratuito em outros empreendimentos de pesquisas. O tópico "aplicações de sensoriamento" é abordado neste trabalho resumidamente. A revisão pode ser um ponto de partida não apenas para conhecer as atuais aplicações de sensoriamento para DPMH (o que possibilita o desenvolvimento de novas soluções), mas também para encontrar soluções reutilizáveis. Portanto, os objetivos alcançados pela revisão são: (i) apresentam resultados de uma busca sistemática em bibliotecas digitais; (ii) identificaram tendências e oportunidades de pesquisas para DPMH; (iii) encontraram todas as aplicações de sensoriamento para DPMH. Os resultados desta revisão sistemática também são relevantes para engenheiros de dados e especialistas em Machine Learning (ML) que dedicam esforços no desenvolvimento de soluções DPMH.

[1] https://parsif.al/

Buscou-se identificar estudos que tenham apresentado aplicações de sensoriamento capazes de coletar dados. Dois pesquisadores realizaram uma pesquisa exaustiva em bibliotecas digitais. A busca de artigos relatando aplicações de sensoriamento foi realizada em 8 bibliotecas digitais.

Foi projetado as *strings* de pesquisa para recuperar artigos apresentando aplicações de sensoriamento para DPMH. Essas *strings* de pesquisa foram cuidadosamente projetadas para atender ao foco da pesquisa. A *string* de busca dos artigos foi desenvolvida com base no objetivo da revisão, questões de pesquisa e suas motivações. Usaram palavras-chave e seus sinônimos para maximizar os resultados. Para evitar a falta de artigos, avaliaram a adequação da *string* em uma pesquisa piloto, foi usado os estudos desenvolvidos por Liang et al. (LIANG; ZHENG; ZENG, 2019) (*ScienceDirect*) e Torous et al. (TOROUS *et al.*, 2016) (*PubMed*) como artigos de controle. Essa busca piloto conseguiu recuperar os estudos citados, demonstrando assim sua capacidade de encontrar artigos relevantes para esta revisão. Ao final da busca, artigos duplicados foram identificados e removidos usando o *Parsif.al*. Um conjunto de critérios de seleção foi definido para rastrear artigos de pesquisa. É importante ressaltar que nenhum limite de intervalo de datas foi aplicado à literatura incluída na revisão.

Na fase de seleção, dois pesquisadores (e.g., autores) realizaram o processo de seleção dos artigos com base nos critérios de inclusão e exclusão. Esse processo consistiu em três fases sequencias: (i) triagem dos estudos por meio da análise de metadados (i.e., título, resumo e palavras-chave); (ii) análise do texto completo dos artigos selecionados na fase de triagem; e (iii) condução do *snowballing* (WOHLIN, 2014).

Na etapa de extração de dados dos artigos selecionados para responder às questões de pesquisa definidas na revisão. Para tanto, um formulário de extração de dados foi elaborado por dois autores e validado pelos juízes. O formulário foi usado para extrair informações relevantes apresentadas pelos estudos revisados, permitindo assim, responder às questões de pesquisa e identificar questões em aberto e tendências de pesquisa.

Com os resultados da revisão, foi utilizado oito bibliotecas digitais para busca de artigos científicos que apresentassem aplicações de sensoriamento para DPMH. Os pesquisadores usaram a abordagem de *snowballing* e adicionaram 5 artigos, o resultado foi encontrado 31 artigos (aplicações de sensoriamento).

3.2 Aplicações de Sensoriamento

A Tabela 1 apresenta 7 dos 31 aplicações que mais se aproxima dos objetivos deste trabalho, os motivos são: aplicações que usam sensores físicos e virtuais para monitorar, avaliar e tratar a saúde mental e bem-estar das pessoas, reuso da ferramenta, ou seja, adaptando-se aos requisitos de cada projeto, fornece recursos para implementar outros

componentes, construir e modelar a ferramenta conforme as necessidades.

Na tabela, as aplicações são caracterizadas considerando o "Sistema Operacional", as "Fontes de dados de contexto" são sintetizados com base no trabalho de Palaghias (PALAGHIAS *et al.*, 2016), que classifica os sensores utilizados pelas aplicações em posicionamento, inercial, virtual e ambiente, conforme detalhado na Seção 2.1.4. A coluna "extensível" indica que a solução pode ser estendida, ou seja, não foram desenvolvidas para estados mentais específicos e podem ser aplicadas a diversos cenários. A última coluna da tabela destaca se a ferramenta computacional consegue compor fenótipos digitais a partir das informações de alto nível, ou seja, a ferramenta coleta os dados brutos dos dispositivos móveis e vestíveis dos usuários, esses dados são pré-processados e inferido informações de alto nível que representam comportamento, hábitos, humor. Portanto, pode observar que o framework *OpenDP* se destaca entre as outras ferramentas, por ser extensível, e por último, capaz de compor fenótipos digitais.

Nome	**Sistema Operacional**	**Fonte de dados de contexto**	**Extensível**	**Compõe fenótipos digitais**
Aware	Android/iOS	Posicionamento, Inercial e Virtual	Sim	Não
Funf	Android	Posicionamento, Inercial e Virtual	Sim	Não
Purple Robot	Android	Posicionamento, Inercial e Virtual	Não	Não
Beiwe	Android/iOS	Posicionamento, Inercial, Virtual e Ambiente	Não	Sim
Sensus	Android/iOS	Posicionamento, Inercial, Virtual e Ambiente	Não	Não
SituMan	Android	Posicionamento e Inercial	Não	Não
Lamp	Android/iOS	Posicionamento	Não	Sim
OpenDP	Android	Posicionamento, Inercial, Virtual e Ambiente	Sim	Sim

Tabela 1 – Lista de artigos de pesquisa.

3.2.1 *Aware*

Aware (FERREIRA; KOSTAKOS; DEY, 2015) (ano da última atualização: 2020) é uma das plataformas identificada na revisão que possui características semelhantes com a solução proposta neste trabalho. É uma plataforma aberta de pesquisa de computação móvel para mineração, visualização e análise de dados, e está disponível como aplicação Android. O *Aware* disponibiliza uma API e fornece aos três *interfaces* que os usuários podem configurar a plataforma como desejar, conforme pode ser observado na Figura 5 os nomes das *interfaces* visuais: o *Sensor Manager*, o *Plugin Manager* e o *Stream*.

A Figura 5 mostra a primeira *interface Sensor Manager*. O *Sensor Manager* é responsável por apresentar informações dos sensores utilizados no *smartphone* como: quais

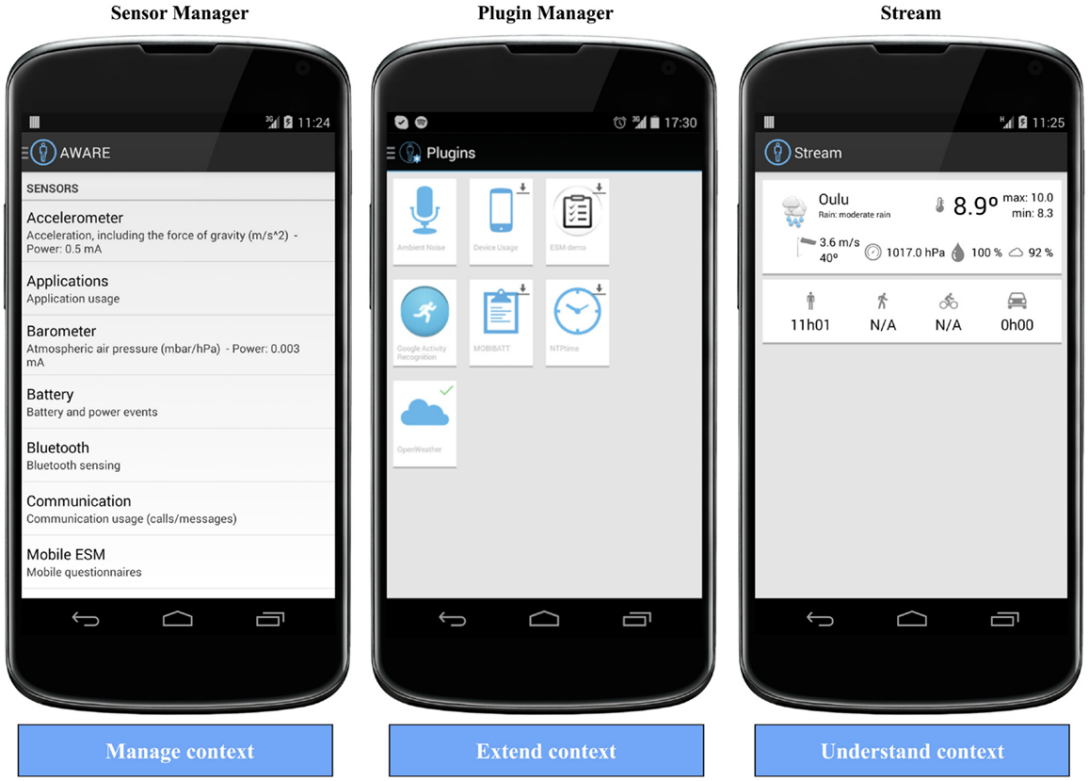

Figura 5 – Aplicação móvel *Aware* (FERREIRA; KOSTAKOS; DEY, 2015).

os sensores ativos, porcentagem do consumo de energia, identificador do usuário e o log de mensagens do *Aware*. Já a *interface Plugin Manager* é responsável apresentar quais os *plugin* estão disponíveis para serem usados. O *plugin* trata-se de uma situação de interesse (e.g., atividade física, sociabilidade) que vai inferir informações de alto nível, por exemplo, caso seja disponibilizado para o usuário o *plugin* de atividade física, deverá aparecer na tela do *Plugin Manager* para o usuário baixar e ativar. A última tela na Figura 5 apresenta a *interface Stream*. A *interface Stream* é responsável por apresentar um resumo com informações inferidas pelos *plugin*, conforme pode ser observado na Figura 5 informações de tempo, temperatura, velocidade do vento.

A Figura 6 apresenta a arquitetura do *Aware* composta pela camada *Client* e *Server*, e os principais componentes. A camada *Client* do *Aware* é composta por vários sensores físicos e virtuais (e.g., GPS, SMS, chamadas telefônicas), agendamento de questionários, *plugins* (que coleta dados de um ou mais sensores. A camada *Client* os dados dos sensores são coletados (*Sensing*) sendo processados para inferir informações de alto nível (*Analysis*), além disso, possuem um banco de dados local (SQLite), e utiliza recursos nativos do Android como: *Broadcast Receivers* (ANDROID, 2022c), *Content Observers* (ANDROID, 2022a) e *Content Providers* (ANDROID, 2022b) para compartilhar os dados de contexto produzido pelo *Aware*. Já a camada do *Aware Server* também possui um banco de dados (MySQL) para o armazenamento dos dados gerados pelo *Aware*, uma *Dashboard* para visualização dos dados de contexto.

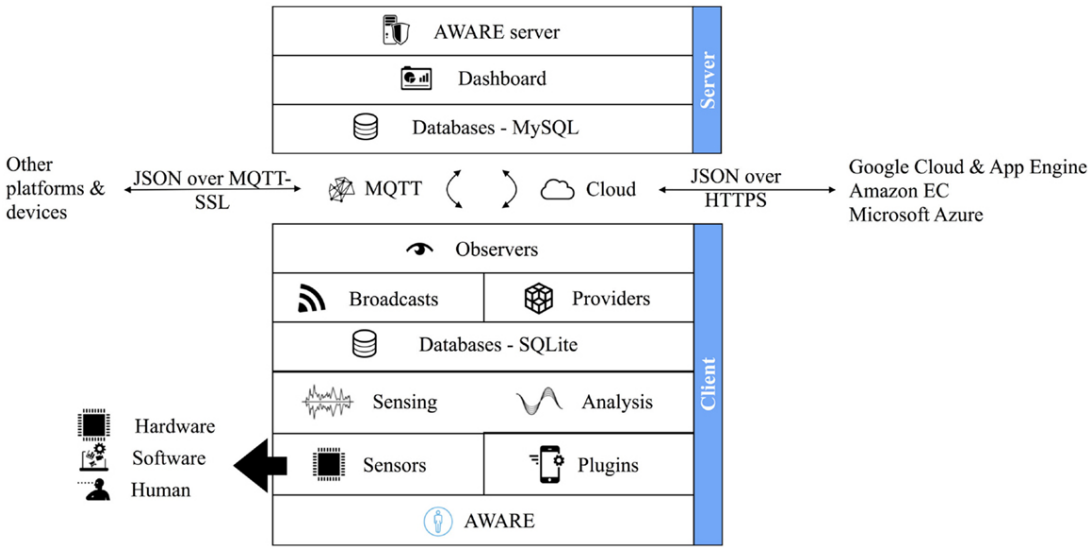

Figura 6 – Arquitetura do *Aware* (FERREIRA; KOSTAKOS; DEY, 2015).

3.2.2 *Funf*

Funf (AHARONY *et al.*, 2011a) (ano da última atualização: 2016) é um framework extensível, possui uma API para adicionar novas funcionalidades e gerenciar a ferramenta. Além de disponibilizar uma aplicação que coleta os dados dos sensores, ele aciona relatórios entregues por *e-mail* aos interessados sobre o *status* dos *smartphones* (e.g., ligado, desligado) e notificações na tela do *smatphone* do usuário, como pode ser observado na Figura 7. A Figura 7 exibe uma aplicação que utilizou o framework *Funf*, da esquerda para a direita, a *interface* mostrando a versão do framework, estado de sincronização com o servidor, a próxima *interface* apresenta questionários, e a última *interface* apresenta a configuração dos sensores.

O objetivo do *Funf* é fornecer um conjunto de funcionalidades reutilizáveis e de código aberto, permitindo a coleta, o envio dos dados e configuração dos sensores no *smartphones*. O desenvolvedor pode adaptar o *Funf* para atender os requisitos de cada pesquisa usando uma API, podendo adicionar novas funcionalidades à sua arquitetura.

A Figura 8 exibe a arquitetura do framework *Funf*. O framework *Funf* possui um componente *Funf Manager* responsável por gerenciar e configurar o framework, também é responsável por selecionar os sensores que serão utilizados para coletar os dados, possui um componente *probe*, significa que cada sensor (e.g., GPS, SMS, acelerômetro, magnetrômetro) utilizado para coletar os dados é chamado *probe* na arquitetura do *Funf*, e o componente *pepiline*, responsável pelo armazenamento, criptografia e distribuição dos dados dos sensores para o servidor.

O componente *Funf Manager* possui o serviço principal do framework *Funf*, é responsável pelo agendamento das diferentes ações de coleta de dados (e.g., definir a frequência de coleta os dados sensores). A configuração é definida de forma que as ações

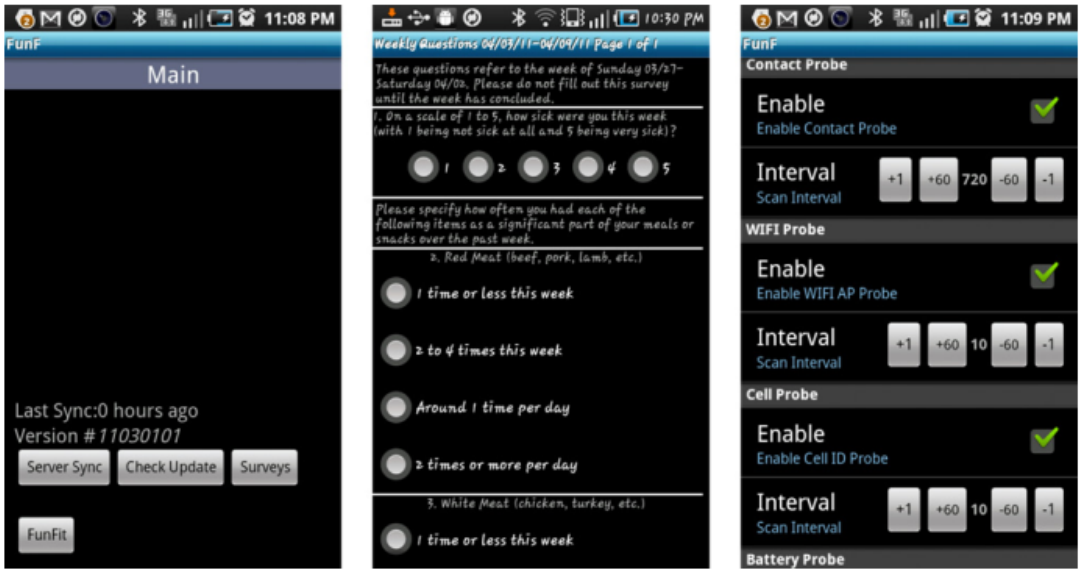

Figura 7 – *Interfaces* visuais do *Funf*: estado de sincronização e exibição de versão (à esquerda), questionário (centro) e de configuração do sensor (direita) (AHARONY *et al.*, 2011a).

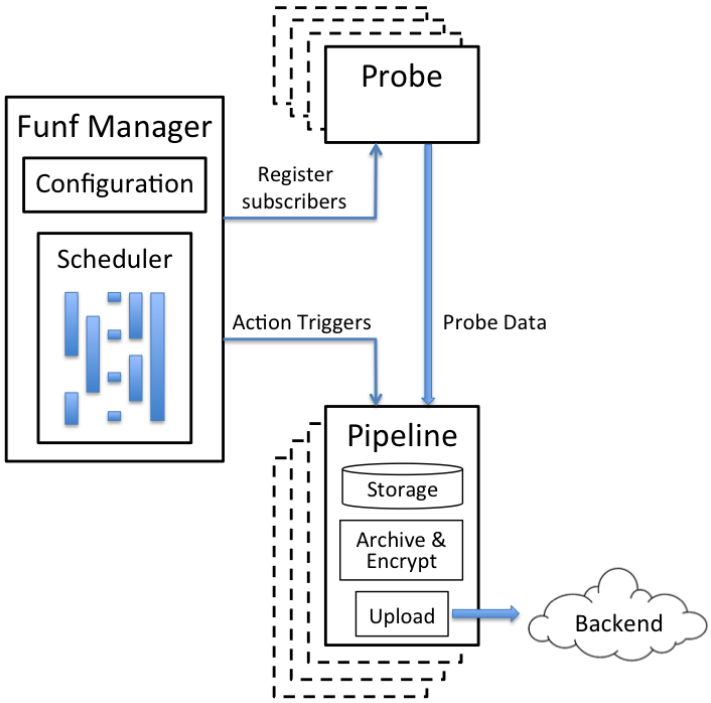

Figura 8 – Arquitetura de alto nível do lado do *smartphone Funf* (AHARONY *et al.*, 2011a).

que consomem muita bateria (e.g., varreduras de GPS) sejam executadas em intervalos mais longos, minimizando o consumo da bateria. Todos os *probe* suportam um conjunto de comportamentos e cada um define um conjunto de parâmetros de configuração que o controla e o formato de sua saída (e.g., pode habilitar o sensor GPS para coletar os dados de coordenadas a cada 5 minutos). Os *probes* podem ser configurados localmente

no dispositivo ou remotamente por meio do servidor. Os dados coletados dos sensores do *smartphone* são salvos localmente no banco de dados SQLite, a cada três horas os dados são recuperados do banco e distribuídos (*upload*) para o servidor, minimizando o consumo de energia e armazenamento. Caso a conexão com a rede caia, os dados são enviados assim que retornar a conexão. Por fim, o servidor recebe e processa todos os dados dos sensores e os insere em um banco de dados MySQL.

3.2.3 *Purple Robot*

O *Purple Robot* (SCHUELLER *et al.*, 2014) (ano da última atualização: 2016) é um framework que apoia a criação de aplicações móveis e *web* com objetivo de identificar comportamentos dos usuários. O *Purple Robot* fornece os seguintes recursos: gerenciamento de usuários (e.g., cadastrar/excluir/editar), criação de conteúdo (e..g., texto, vídeo, áudio), entrega de relatórios (e.g., por *e-mail*) e gerenciamento de dados (e.g., disponibiliza os dados brutos dos sensores).

O gerenciamento de usuários é um elemento importante, pois garante que os dados sejam acessados por pessoas autorizadas e as informações sejam criptografadas. O sistema *Purple* rastreia quem entra no sistema, quando, e consegue fornecer acesso às informações e conteúdos corretos para cada usuário individual e parte interessada. Isso também é crucial para a prática de garantir questões de privacidade e segurança. Para oferecer suporte à privacidade e segurança, um recurso do *Purple* permite que o administrador crie identificadores e senhas exclusivos e designem esses usuários a um estudo/pesquisa específico. Outro aspecto do gerenciamento de usuários é permitir a definição de variáveis relacionadas a um usuário e a especificação de como essas variáveis correspondem a outros recursos na intervenção. Por exemplo, o gerenciamento de usuários do *Purple* inclui a capacidade de definir quais partes interessadas (e.g., profissional de saúde, pesquisador) têm permissão para acessar informações demográficas ou pessoais de saúde.

O *Purple* permite que pesquisadores que não dominam o conhecimento de programação possam participar diretamente do processo de desenvolvimento da aplicação móvel. A outra funcionalidade do *Purple* é a criação de conteúdo. O *Purple* usa o termo "conteúdo" para referir a qualquer informação fornecida a um usuário (e.g., texto), notificações (e.g., *e-mail*, mensagens, notificações no *smartphone*). O *Purple* suporta a aquisição de vários sensores de smartphone (e.g., GPS, acelerômetros, altímetros, luminosidade). Ele também fornece uma API usada para conectar dispositivos vestíveis e permite que sensores adicionais sejam adicionados.

Outra funcionalidade do *Purple* é a entrega dos conteúdos. Isso inclui exibições de calendário e *Interfaces* visuais. A entrega de conteúdo permite essencialmente que a intervenção seja executada para o usuário final em seu dispositivo móvel ou computador com base em regras definidas nos elementos de autoria e gerenciamento de conteúdo no

Purple.

3.2.4 *Beiwe*

Uma das plataformas mais conhecidas e utilizadas em estudos na saúde é o *Beiwe* (TOROUS *et al.*, 2016) (ano da última atualização: 2021). Ele possui os seguintes componentes: um portal (página *web*) utilizado para gerenciar estudos, uma aplicação móvel responsável pela coleta de dados de diversos sensores de *smartphones* e, por fim, o *Beiwe* integra-se com servidores em nuvem Amazon Web Services (AWS) para processar os dados coletados. O *Beiwe* tem sido usado em diferentes estudos, por exemplo, o trabalho de Henson (HENSON *et al.*, 2021) usou o *Beiwe* para prever recaídas de esquizofrenia, com base na detecção de anomalias de sociabilidade (chamada telefônica e registros de SMS), mobilidade (coordenadas geográficas), tempo de tela ligada e monitoramento do sono dos usuários.

A Figura 9 mostra o fluxo de trabalho da plataforma *Beiwe*. O processo de pesquisa na plataforma inicia com o portal (primeira etapa na Figura 9). O portal (página *web*) pode personalizar a aplicação móvel (segunda etapa na Figura 9) para um determinado estudo, isso permite que os pesquisadores configurem pesquisas, especifiquem o conteúdo das aplicações, quais sensores são usados para a coleta de dados e a frequência de coleta. Isso resulta na geração de uma nova aplicação específica para um estudo.

Figura 9 – Fluxo de trabalho da plataforma *Beiwe* (TOROUS *et al.*, 2016).

Depois que o estudo é criado no portal, configurado para a aplicação com os sensores a serem utilizados, ela fica disponível para baixar da rede. O *Beiwe* armazena os dados coletados em *buffer* no dispositivo até que o Wi-Fi esteja disponível, ponto onde os dados são carregados para o banco de dados do servidor (terceira etapa na Figura 9) e apagados do dispositivo. O *Beiwe*, os dados coletados são analisados por técnicas de aprendizagem de máquina (quarta etapa na Figura 9). A análise de dados não é feita em tempo real no smartphone, mas em um servidor dedicado. Por fim, os resultados das análises de dados (quinta e última etapa na Figura 9) são demonstrados pelo portal, onde possui recursos gráficos que o pesquisador pode explorar. Como o *Beiwe* disponibiliza o *dataset* com os dados coletados e o resultado da análise, o pesquisador tem liberdade para usar outra ferramenta gráfica para a visualização dos dados.

3.2.5 *Sensus*

Outra solução relacionada ao nosso trabalho é o *Sensus* (XIONG *et al.*, 2016) (ano da última atualização: 2019). O *Sensus* é uma ferramenta exclusiva para uso em pesquisas científicas (Figura 10). Ele consiste em duas aplicações móveis, uma instalada nos *smartphones* dos usuários monitorados para coletar dados dos sensores e outro deixado com os profissionais da saúde para gerenciar os estudos, e um servidor em nuvem para processar os dados coletados. O *Sensus* pode ser utilizado em outras pesquisas, pois código aberto e disponibiliza uma API para configurar a plataforma.

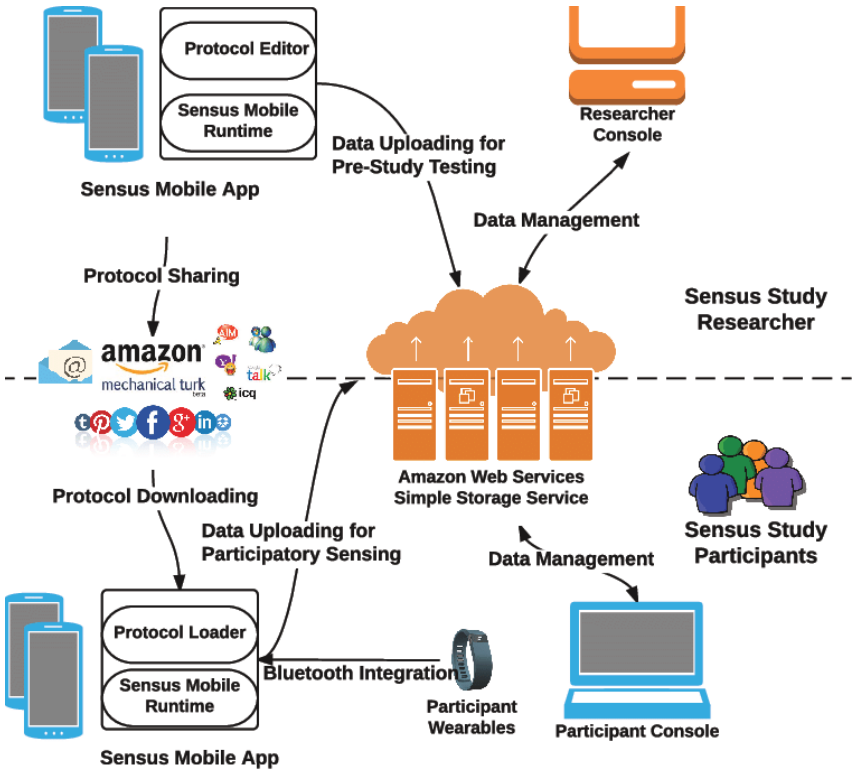

Figura 10 – Arquitetura de alto nível do *Sensus* (XIONG *et al.*, 2016).

A Figura 10 apresenta uma arquitetura de alto nível do *Sensus* a partir das perspectivas do pesquisador (parte superior da Figura 10) e dos usuários (parte inferior da Figura 10). Às duas perspectivas compartilham o arquivo de protocolo e o armazenamento de dados de estudo dentro do AWS. A pesquisa inicia com os pesquisadores usando a aplicação *Sensus* para configurar um protocolo (arquivo que contém a seleção dos sensores que vão usar, *id* do usuário, informações da pesquisa de consentimento do usuário), disseminado para os usuários do estudo como um arquivo JSON criptografado. Cada usuário do estudo recebe o arquivo de protocolo e carrega-o na aplicação móvel *Sensus* em seu dispositivo para dar início a execução. A diferença da aplicação *Sensus* do pesquisador para a aplicação *Sensus* do usuário, é o componente *Protocol Editor* e *Protocol Loader*, o primeiro é responsável por criar a pesquisa e editar, selecionar os sensores que serão utilizados, editar as informações de consentimento/contrato da pesquisa para o usuário, e

o segundo é responsável por carregar o arquivo de protocolo e dar início a pesquisa com base na configuração.

A Figura 11 mostra a aplicação *Sensus* em tempo de execução, o editor de protocolo e o carregamento. Conforme a Figura 11, o *Sensus* pode emitir um evento indicando que uma chamada acabou de ser encerrada (passo 1). O evento de chamadas telefônicas é identificado pelo *Sensus*, em seguida, o *Sensus* dispara uma ação após a chamada ser encerrada para coletar dados de contexto (e.g., momento da ligação, duração) (passo 2). Além dos dados de contexto coletados, o *Sensus* obtém as coordenadas de localização para adicionar ao relatório (passo 3); o *Sensus* exibe o relatório na *interface* do usuário (passo 4). Finalmente, as informações coletadas do dispositivo móvel são periodicamente carregadas para o servidor (passo 5).

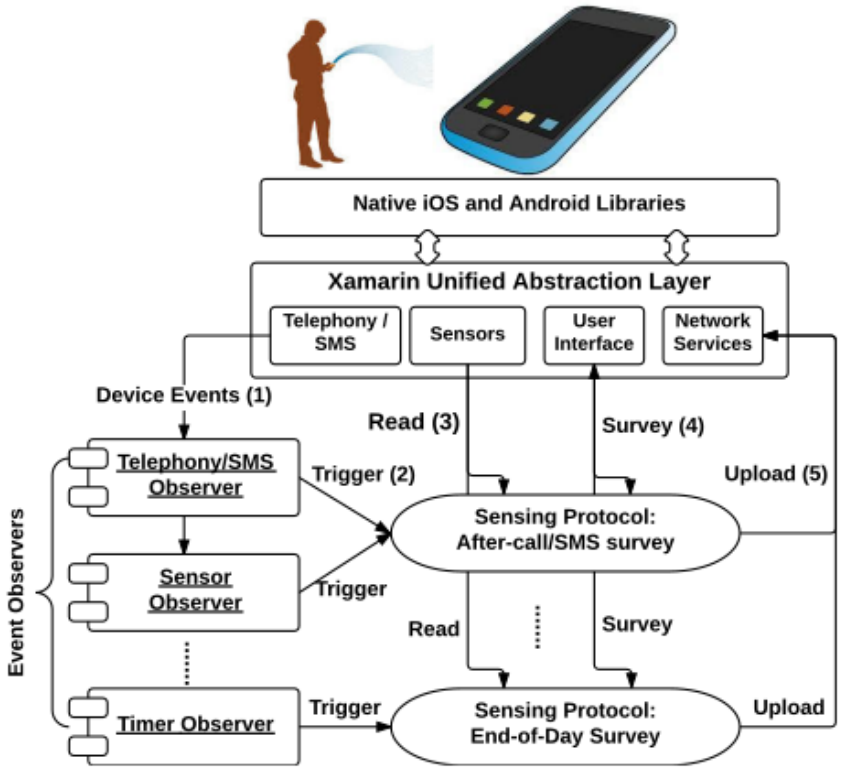

Figura 11 – Aplicação móvel *Sensus* em tempo de execução (XIONG *et al.*, 2016).

3.2.6 *SituMan*

O *SituMan* (TELES *et al.*, 2017) (ano da última atualização: 2016) é um dos trabalhos semelhantes à solução proposta neste trabalho. O *SituMan* é uma solução reutilizável (i.e., um serviço Android) que identifica situações da rotina diária de usuários de dispositivos móveis. Ele possui um mecanismo de inferência que faz uso de lógica *fuzzy* que reconhece situações usando dados de contexto coletados de sensores. Os recursos do motor de inferência são disponibilizados através de uma API para que as aplicações possam

permitir o reconhecimento de situações de interesse. As situações de interesse podem ser úteis tanto para a solicitação de autor-relatos dos usuários em momentos mais adequados, quanto para que os profissionais de saúde mental conheçam seu cotidiano.

A Figura 12 apresenta a arquitetura do *SituMan* (TELES *et al.*, 2017). Os serviços *web* são usados para receber informações sobre o estado mental e experiências dos usuários fornecidos pela aplicação móvel *MoodBustter*. O *MoodBustter* é uma aplicação móvel *Ecological Momentary Assessment* (EMA)/*Ecological Momentary Intervention* (EMI) para usuário em tratamento de depressão. Uma pessoa pode usá-lo para obter uma visão precisa de como seu humor muda ao longo do tempo. Ele faz perguntas sobre o estado do usuário em vários momentos do dia, ou seja, os usuários são avaliados relatando repetidamente suas experiências e estado mental (e.g., humor, motivação, ansiedade) ao longo do dia. Ele também apresenta as respostas acumuladas em gráficos, que posteriormente podem ser consultadas pelo profissional de saúde mental. Esses gráficos também podem dar aos usuários uma visão melhor sobre seu próprio comportamento e sintomas.

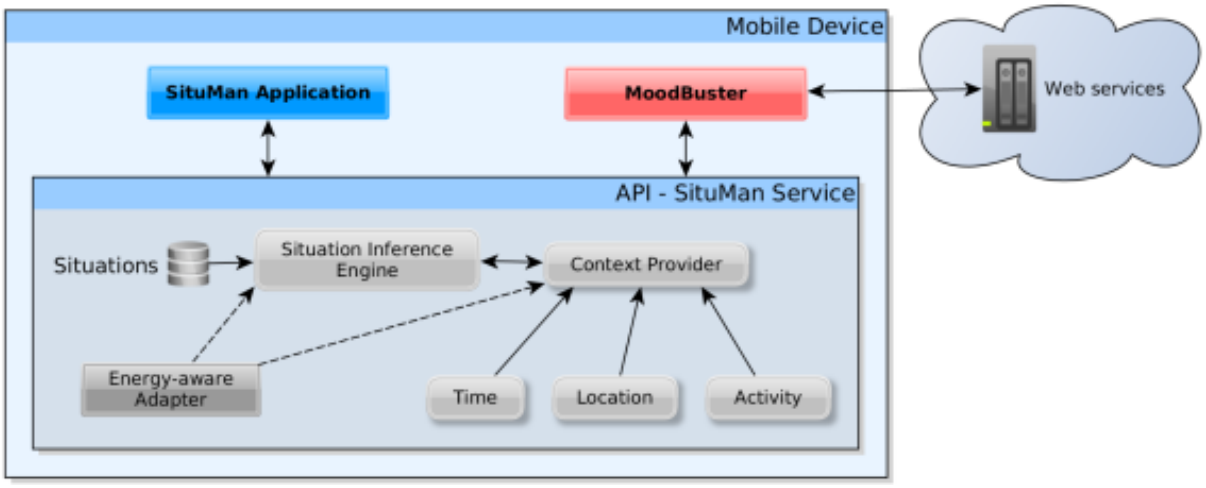

Figura 12 – Arquitetura do *SituMan* (TELES *et al.*, 2017).

O componente *Situation Inference Engine* contém o motor de inferência *fuzzy*, que realiza periodicamente o processo de inferência e salva o resultado no banco de dados de situações. A base de dados é responsável por armazenar as situações definidas pelos usuários e também todas as situações inferidas. A *Situation Inference Engine* obtém dados de contexto do *Context Provider* para realizar o processo de inferência da situação. O *Context Provider* é responsável por obter, formatar e fornecer dados de contexto. O *timestamp* é obtido do relógio local do dispositivo móvel. A atividade do usuário é detectada usando o *Google Play Services* por meio da API de reconhecimento de atividade. A localização dos usuários é obtida de GPS ou rede móveis por meio da API de serviços de localização do Google. Por último, o componente *Energy-aware Adapter* é usado no *SituMan* para ajustar o consumo de energia dependendo do nível de uso da bateria do dispositivo móvel.

3.2.7 *Lamp*

O *Lamp* (WISNIEWSKI; HENSON; TOROUS, 2019) (ano da última atualização: 2022) é uma aplicação composto por diversos questionários e jogos. Os jogos possuem testes de cognição neuropsicológico que fornece uma avaliação básica das habilidades cognitivas dos usuários. Ele é destinado à coleta ativa de dados (e.g., por questionários) e passiva (e.g., toque de tela).

A Figura 13 apresenta a arquitetura da plataforma *Lamp* (WISNIEWSKI; HENSON; TOROUS, 2019). A plataforma *Lamp* funciona com 4 componentes (PSYCHIATRY, 2020; PSYCHIATRY, 2021): o *App*, a *Dashboard*, o Banco de Dados e o *Cortex*. O usuário usa o *App* para responder pesquisas (e.g., questionários), interagir com jogos cognitivos conforme pode ser observado na Figura 14, acessando dicas e recursos úteis ou fazendo exercícios de meditação e respiração; quando ativado e configurado, a aplicação coleta dados do sensor do acelerômetro, localização, pedômetro do dispositivo móvel sem interromper o usuário. Ele também coleta metadados sobre o uso da aplicação pelo usuário, tais como quanto tempo certas perguntas levaram para serem respondidas em uma pesquisa ou quais dicas úteis os usuários inseriram na aplicação, e carrega os dados com segurança para um servidor indicado pelo pesquisador.

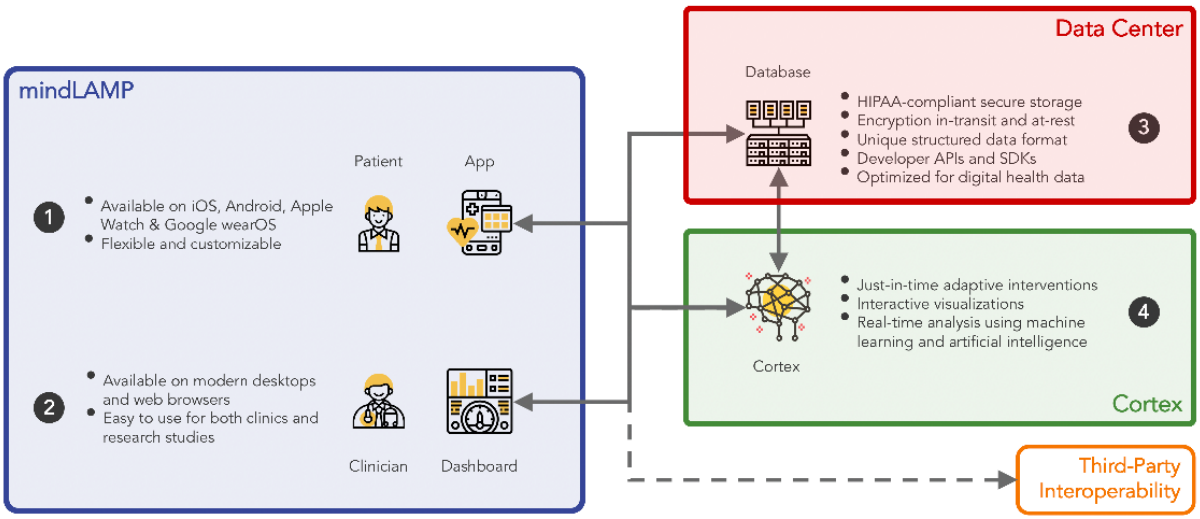

Figura 13 – Arquitetura do *Lamp* (WISNIEWSKI; HENSON; TOROUS, 2019; PSYCHIATRY, 2020; PSYCHIATRY, 2021).

O segundo componente é a *Dashboard*. Os profissionais de saúde podem criar, personalizar e agendar questionários para que usuários interajam, além de acessar informações quase em tempo real sobre usuários. Além disso, a *dashboard* possui um portal de dados para uma visão abrangente dos dados de usuários. A plataforma *Lamp* gerência um banco de dados, trata-se do terceiro componente que integra a plataforma, os dados são indexados de forma cronológica para que o *Lamp* consiga inferir as informações de alto nível. Os pesquisadores podem construir ferramentas reutilizáveis, *pipelines* de análise e conduzir

outras pesquisas reproduzíveis. O quarto e último componente, o *Cortex*, é a *pipeline* de análise de dados que foi projetada para se conectar facilmente ao banco de dados e extrair informações clinicamente úteis, como comportamento sedentário, tempo que permanece em casa, uso de dispositivo/tela.

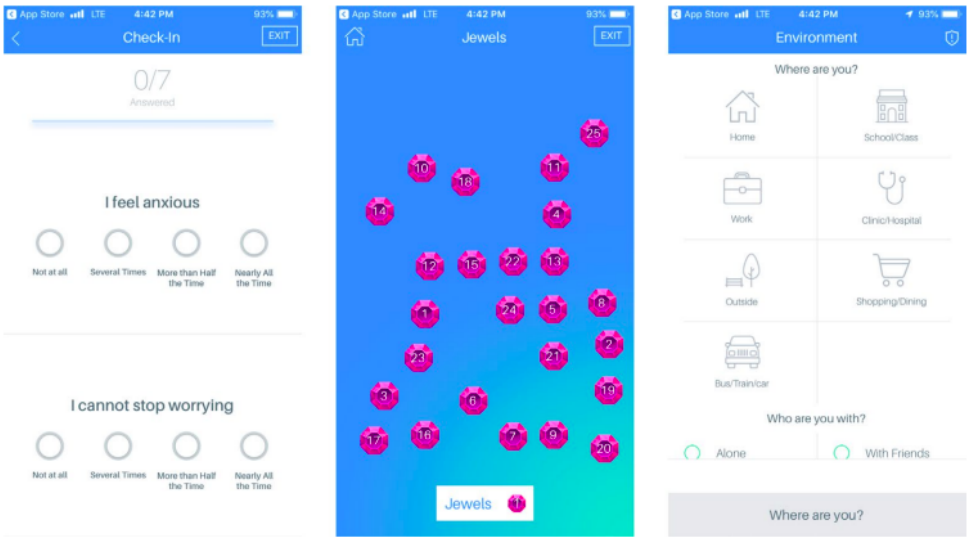

Figura 14 – Telas do *Lamp* (WISNIEWSKI; HENSON; TOROUS, 2019).

3.3 Discussão

Este capítulo apresentou uma sumarização dos resultados de uma RSL sobre aplicações de sensoriamento para fenotipagem digital de saúde mental. Foram apresentadas sete aplicações móveis que mais se destacaram por conter funcionalidades similares com a solução proposta nesta pesquisa. A solução proposta nesta pesquisa de mestrado difere das anteriores em vários aspectos. O primeiro ponto a destacar é que a solução proposta é reusável e extensível, então contribuindo no desenvolvimento de outras pesquisas de fenotipagem digital. Ela permite que desenvolvedores implementem aplicações para atender requisitos específicos do cenário. O desenvolvedor, usuário da solução proposta, não precisa implementar do zero recursos como, por exemplo, a coleta de dados de sensores físico e virtuais, e a distribuição desses dados para a nuvem. Ao mesmo tempo, o arcabouço de software permite a inclusão de novos componentes à arquitetura, estendendo as funcionalidades já providas.

O segundo ponto é que, ao invés de focar em um estado mental específico, a solução proposta é genérica, e pode ser reutilizada em diferentes estudos/pesquisas que podem envolver diferentes comportamentos, hábitos e situações de interesse. Portanto, o framework não é limitado a um transtorno mental específico. Por fim, outro diferencial da solução proposta é que ela coleta dados não somente oriundos de sensores de *smartphones*, mas também permite que o *smartphone* se conecte via *Bluetooth* com diferentes dispositivos

vestíveis (e.g., relógios inteligentes) para realizar a coleta de dados. Isso aumenta a variedade de dados, então possibilitando a geração de mais tipos de informações de alto nível a serem identificadas.

4 Solução Proposta

O framework *OpenDP* (do inglês, ***Open Digital Phenotyping of Mental Health of Framework***) foi desenvolvido no Laboratório de Sistemas Distribuídos Inteligentes da Universidade Federal do Maranhão (LSDi-UFMA), como objetivo de facilitar o desenvolvimento de aplicações móveis de fenotipagem digital. Diante disso, este trabalho de mestrado se concentrou na implementação de um framework para dispositivos móveis.

4.1 Requisitos do *OpenDP*

De modo a fornecer um amplo suporte ao desenvolvimento de aplicações para dispositivos móveis de fenotipagem digital, sem deixar de abranger também o desenvolvimento de aplicações para a área de saúde mental com foco em transtornos mentais, definimos que o *OpenDP* deveria atender os seguintes requisitos:

1. *Extensibilidade*: a solução deve prover um mecanismo para adicionar novos recursos, ou seja, ela deve ser uma solução genérica, capaz de trabalhar com diferentes transtornos mentais, permitindo a inclusão do processo de identificação de variadas situações de interesse (e.g., mobilidade, atividade física, sociabilidade);
2. *Reutilizável*: esse requisito expressa que a solução deva ser aproveitada para outros estudos/pesquisas, até mesmo para o monitoramento do bem-estar das pessoas;
3. *Geração de fenótipos digitais*: a solução deve conseguir compor fenótipos digitais por meio de informações de alto nível e de dados brutos dos sensores dos dispositivos móveis e vestíveis;
4. *Coletar dados de sensores físicos e virtuais*: esse requisito expressa que a solução deva conseguir coletar dados de sensores físicos e virtuais de dispositivos móveis e vestíveis;
5. *Comunicação segura*: esse requisito expressa que a solução deva garantir propriedades básicas de segurança, especialmente, quando tais aplicações envolvem o consumo e a distribuição de dados sensíveis;
6. *Código aberto*: esse requisito expressa que a solução deva ser de código aberto, o seu código-fonte fica disponível para o público acessar.

A seção seguinte apresenta uma visão geral do framework. Nela é possível identificar os componentes que implementam as funcionalidades necessárias para atender os requisitos levantados.

4.2 Visão Geral

O framework *OpenDP* está implementado na linguagem Java para o sistema operacional Android. Ele é *open source*, o código-fonte está disponível para baixar no servidor do *GitHub* (JEANCOMP, 2022), acompanhado da documentação, sob a licença Lesser General Public License (LGPL) versão 3 (GNU, 2022).

A solução proposta é dividida em duas partes, a primeira contém o *core*, e a segunda o *plugin*. O *core* é o núcleo do framework, onde está centralizado o controle dos recursos como: gerenciamento dos sensores, gerenciamento dos módulos de processamento de dados brutos no *core*, gerenciamento de um ou mais *plugins*, e composição dos fenótipos digitais. O *core* possui duas instâncias do CDDL, em que a primeira fica no *DPManagerService*, onde roda o *micro broker* que serve de comunicação entre os componentes do framework, e a segunda instância do CDDL fica no *PhenotypeComposer*, a qual é usada para estabelecer um canal de comunicação com o *broker* externo para enviar os fenótipos digitais. A segunda parte do framework é o *plugin*, sendo então uma arquitetura que é possível adicionar novos módulos de processamento de dados, ou seja, é possível estender o framework. Para exemplificar melhor o conceito de *plugin* no framework, podemos citar um exemplo de uso do framework no monitoramento do bem-estar humano. Instalando no *smartphone* do usuário o framework com dois módulos de processamento de dados: a atividade física para fazer reconhecimento do estado físico (andando, correndo, dirigindo) e a sociabilidade para identificar se o usuário está socializando. Através do *plugin*, o desenvolvedor pode adicionar um novo módulo para monitorar o usuário sem precisar recompilar o projeto, bastando implementar o novo módulo na arquitetura do *plugin*, instalar e executar.

A arquitetura geral do framework é ilustrada na Figura 15. Os seguintes componentes compõem o *core*:

- *DPManager (Digital Phenotyping Manager)*: componente responsável por gerenciar o framework, a qual utiliza o padrão *builder* (GUPTA, 2014) para fornecer as configurações necessárias para o funcionamento da ferramenta, tais como informar o endereço do *host*, porta e modo de composição dos fenótipos digitais. Para consultar quais os módulos de processamento de dados estão disponíveis, ao iniciar o framework, o *DPManager* mantém uma lista dos *DataProcessor* disponíveis para uso, uma lista de *DataProcessor* ativos que estão sendo executados no *core* e no *plugin*. Portanto, é através do *DPManager* que iniciar/parar o framework, ativar/desativar os módulos de processamento de dados (i.e., DataProcessor), habilitar/desabilitar o serviço de segurança e configurar o modo de composição de fenótipos digitais;
- *DPManagerService*: serviço responsável por iniciar os componentes do framework (e.g., *ProcessorManager*, *PhenotypeComposer*, *RawDataCollector*). Visando garantir que o framework mantenha-se em execução e não seja derrubado pelo sistema

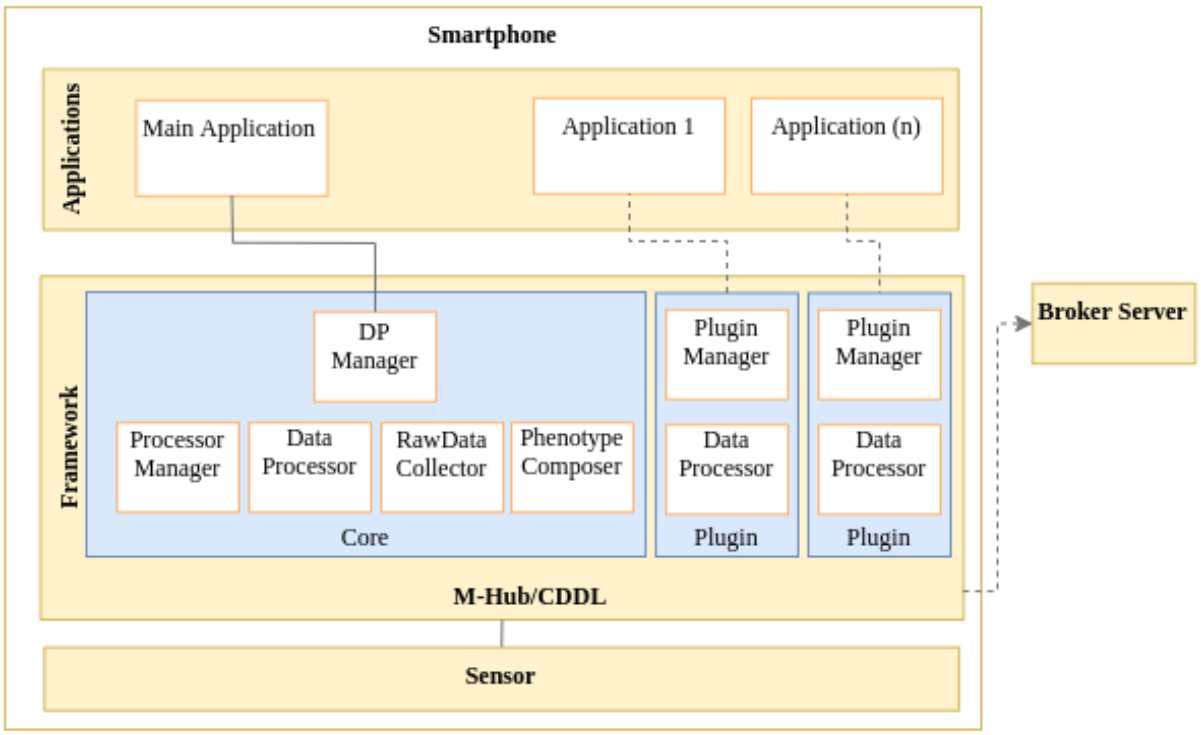

Figura 15 – Arquitetura do Framework *OpenDP*.

operacional Android, foi colocado o serviço *DPManagerService* em primeiro plano, de forma permanente. Dessa forma, um alerta fica na barra de notificação do *smartphone* indicando que o framework está em execução. O *DPManagerService* contém uma instância do CDDL para rodar o *micro broker*, servindo de canal de comunicação entre os componentes internos do *core*, e também através desse *micro broker* que o *plugin* se conecta para trocar mensagens com o *core*;

- *ProcessorManager*: serviço responsável pelo gerenciamento dos *DataProcessor*. Ele recebe do *DPManager* a lista dos *DataProcessor* que devem ser ativados ou desativados. O *ProcessorManager* também recebe de cada *DataProcessor* uma lista com os nomes dos sensores que serão inciados, fazendo o gerenciamento de cada sensor e suas respectivas permissões em tempo de execução. Para isso, ele utiliza uma tabela de controle interna para gerenciar todos os sensores ativos em uso no momento. Por exemplo, o *ProcessorManager* impede que um sensor de localização seja desligado por um *DataProcessor* mesmo que outro *DataProcessor* esteja usando este sensor, ou um sensor seja iniciado duas vezes por mais de um *DataProcessor*. Surgindo o conflito de sensores como citado acima, o *ProcessorManager* gera uma exceção informando ao desenvolvedor que o sensor já foi iniciado.

- *DataProcessor*: serviço responsável por inferir e gerar informações de alto nível através da coleta de dados brutos dos sensores. Ao criar um *DataProcessor*, o desenvolvedor informa o nome do *DataProcessor* para controle interno e uma lista de sensores

que queira receber os dados de contexto Após a sua inicialização, ele estará pronto para receber os dados de contexto e implementar a sua regra de negócio para inferir as informações de alto nível, identificando os eventos de fenotipagem digital. O *DataProcessor* é onde o desenvolvedor implementará a sua regra de negócio, usar o algoritmo (e.g., ML, CEP, logical *fuzzy*) que convém para gerar as informações de alto nível. As informações de alto nível inferidas são enviadas para o *PhenotypeComposer*, onde irão compor os fenótipos digitais, conforme o modo de composição previamente selecionado. O desenvolvedor, ao criar um módulo de processamento de dados brutos estendendo da classe *DataProcessor*, deve atentar-se para a taxa de geração de dados, pois existem sensores que oferecem geração de dados contínuos (maior consumo de energia) e periódicos (menor consumo de energia). Surge então o termo Frequência de Geração de Dados (FGD). Por exemplo, o sensor acelerômetro possui uma frequência de geração de dados contínua, e para não comprometer o consumo de energia, pode-se configurar o sensor acelerômetro para a cada intervalo de tempo coletar o dado, e em seguida desativá-lo. Já o sensor virtual de chamadas telefônicas e mensagens de texto (SMS) possui uma frequência de geração de dados periódica, não chegando a comprometer o consumo de energia do dispositivo móvel. Portanto, a FGD define o intervalo máximo em que dados de contexto são gerados, ou seja, a taxa de amostragem;

- *PhenotypeComposer*: serviço responsável pelo gerenciamento e composição dos fenótipos digitais. O *PhenotypeComposer* possui três modos de configuração. O primeiro *send_when_it_arrives* é o mais simples dos modos de composição de fenótipo, pois a medida que as informações de alto nível chegam no *PhenotypeComposer*, ele imediatamente distribui ao *broker* externo. Nesse modo, a informação não é mantida para haver uma composição com outros dados. As aplicações de saúde mental que requeiram uma certa urgência (e.g., aplicações que inferem sintomas ou traços de ideação suicida) em ter informações dos usuários podem se interessar por esse modo de composição. O segundo modo de composição é o *group_all*, o qual só distribui as informações de alto nível para o *broker* quando chegarem as informações de alto nível de todos os módulos de processamento de dados que estejam ativos. Enquanto essa condição não for satisfeita, as informações são armazenadas no banco de dados. Por exemplo, chega no *PhenotypeComposer* uma informação de alto nível gerada pelo *DataProcessor* denominado Physical_Activity, a informação é armazenada em banco de dados até chegar no *PhenotypeComposer* informações de alto nível dos outros *DataProcessor* ativos. O terceiro modo de composição é o *frequency*, o qual possui um *schedule* que controla a distribuição das informações de alto nível para o *broker* externo, com base na frequência (e.g., segundos, minutos, horas) previamente selecionada ao iniciar o framework. Ao dispará o *schedule*, todas as informações de alto nível armazenadas no banco de dados são recuperados e distribuídas para o

broker externo. A partir do modo de composição de fenótipos digitais *frequency*, surge o termo Frequência de Distribuição de Dados (FDD), o qual refere-se a atribuição de uma frequência em que o fenótipo digital gerado é distribuído para o servidor. Portanto, a FDD define o intervalo em que o fenótipo é enviado para o *broker*;

- *RawDataCollector*: serviço responsável pela coleta de dados brutos de contexto para compor também fenótipos digitais. O *RawDataCollector* precisa dos nomes dos sensores para iniciar a coleta dos dados e escolher o modo de composição dos fenótipos digitais, os quais são similares ao do *PhenotypeComposer*. O modo de composição *send_when_it_arrives* envia os dados brutos para *broker* (i.e., a medida que os dados vão sendo recebidos pelo *RawDataCollector*, é distribuído) e o modo de composição *frequency* funciona através de um *schedule*, quando disparado, recupera do banco de dados todos os dados brutos de contexto e distribui ao *broker*. O *RawDataCollector* também fornece recursos de EPL/CEP (JUNIOR *et al.*, 2019) para um melhor pré-processamento dos dados. O desenvolvedor pode implementar uma regra CEP para filtrar ou monitorar o fluxo de dados gerados pelos sensores.

O M-Hub (GOMES *et al.*, 2017) não contemplava os sensores virtuais (PALAGHIAS *et al.*, 2016), e atualmente, muitas ferramentas de fenotipagem digital estão utilizando esses tipos de sensores virtuais para inferir informações de alto nível, como a socialização dos usuários (TOROUS *et al.*, 2016) e a identificação dos padrões do sono (AHARONY *et al.*, 2011b; TOROUS *et al.*, 2016). Diante disso, neste trabalho foi implementado os *drivers* de chamadas telefônicas, mensagem de texto (SMS), toque de tela, e tela ligada/desligada, e adicionados ao M-Hub como tecnologia interna (mostrado na Figura 4), onde pode ser visualizado como um *SensorPhone*. Dessa forma, o framework *OpenDP* consegue coletar dados dos usuário através dos sensores virtuais e gerar informações de alto nível a partir deles.

4.3 Aspectos de Implementação dos Componentes do *Core*

Como descrito na Seção 1.5, o escopo desse trabalho de mestrado é o desenvolvimento de um framework que facilite a criação de aplicações móveis de fenotipagem digital. Por isso, descrevemos os componentes do *core* como:

4.3.1 *DPManager*

O *DPManager* é o componente principal do framework. Através dela é feito o gerenciamento do framework. Por exemplo, o *start/stop* do framework, a configuração do modo de composição do fenótipo digital, o *start/stop* dos módulos de processamento de dados brutos. O componente *DPManager* é representado no diagrama de classe na Figura 16.

Optou-se por mostrar a interface *DPInterface* e o componente *DPManagerService* com o qual o *DPManager* interage para iniciar os serviços internos.

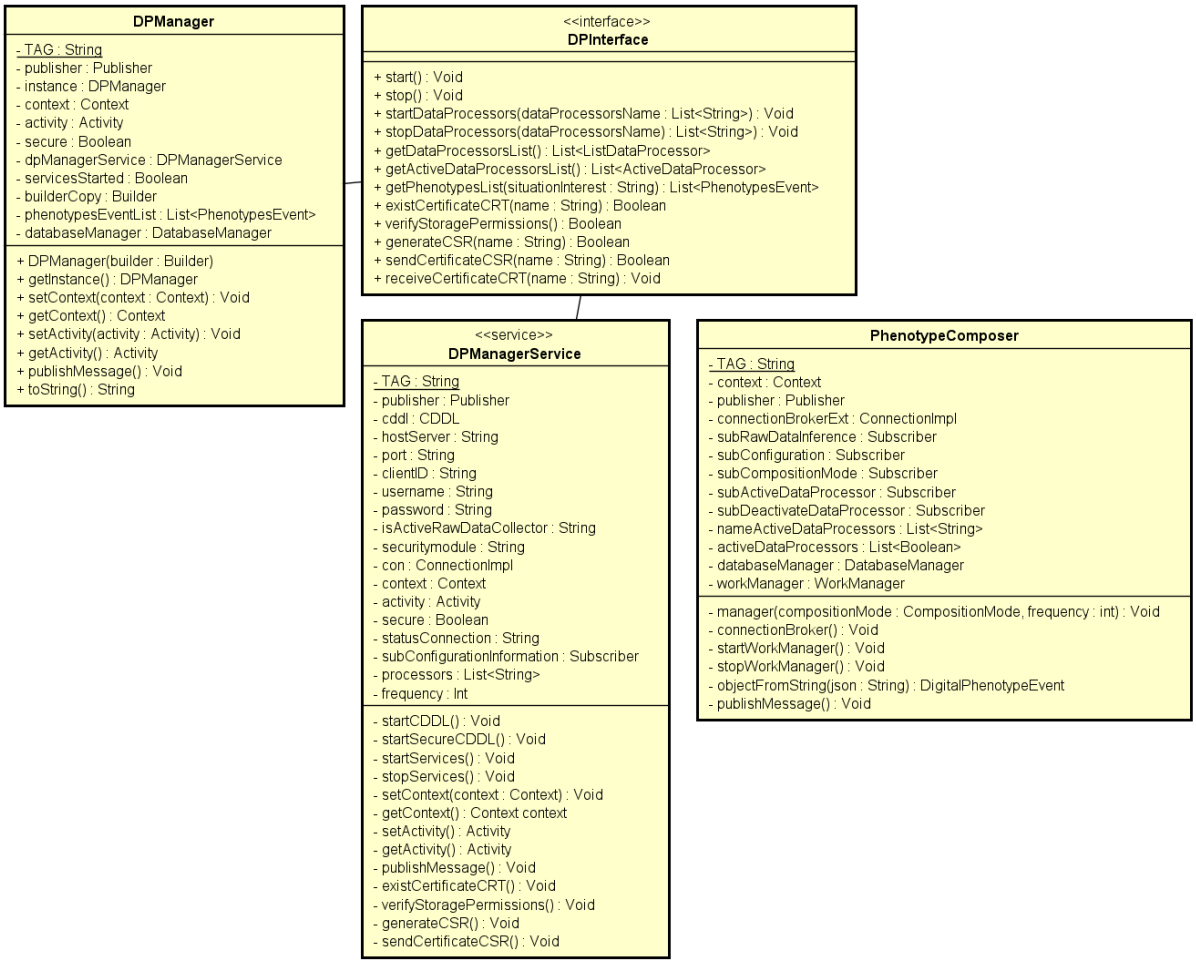

Figura 16 – Diagrama de classe do *DPManager* e outros componentes.

A Figura 16 apresenta a *DPInterface* que fornece todas as assinaturas dos métodos que a *DPManager* precisa para gerenciar o framework. Os métodos *start()* e *stop()* são responsáveis por iniciar ou parar a execução do framework, os métodos *startDataProcessors()* e *stopDataProcessors()* são responsáveis por iniciar os módulos de processamento de dados brutos, os quais necessitam como parâmetro uma lista com os nomes dos processadores que se deseja iniciar/parar a execução. Já com o método *getDataProcessorsList()* obtém-se a lista dos módulos de processadores disponíveis para execução, e o método *getPhenotypesList()* retorna uma lista com as informações de alto nível resultantes das inferências (e.g., eventos de ligação telefônica) gerada por um determinado módulo de processamento de dados, por exemplo, a linha de código "*dpManager.getInstance().getPhenotypesList(Online_Sociability)*;", retorna todas as informações de alto nível geradas pelo módulo *Online_Sociability*.

Outro importante componente representado no diagrama de classe na Figura 16 que o *DPManager* interage é o *DPManagerService*. Trata-se do serviço principal do

framework que fica responsável por guardar a conexão do *Micro Broker* (primeira instância do CDDL) utilizada para comunicação interna entre os componentes do framework. O *DPManagerService* é iniciado pelo *DPManager* e colocada em primeiro plano para reduzir a possibilidade do sistema operacional Android derrubar a execução do serviço. Dessa forma, o framework pode ser fechado e o serviço irá continuar rodando em segundo plano até que o usuário solicite seu fechamento. O *DPManagerService* fica responsável por iniciar o restante dos serviços internos do framework (e.g., *ProcessorManager*, *PhenotypeComposer*).

O diagrama de sequência apresentado na Figura 17 mostra o processo do *DPManager* e seu relacionamento com os demais componentes para iniciar o framework. O componente *DPManager* é inicializado pelo construtor com o padrão *builder*. Através deste padrão, o componente recebe os valores de entrada (e.g., *hostServer*) para configurar a conexão com o *broker* externo, que receberá os fenótipos digitais do framework, além do modo de composição do fenótipo (*frequency*, *group_all* e *send_when_it_arrives*). Em seguida, é inicializado o componente *DPManagerService* que, por sua vez, vai iniciar o CDDL com o *micro broker* e os serviços do framework (*ProcessorManager, PhenotypeComposer*). Assim que o componente *PhenotypeComposer* é iniciado, ele dispara uma mensagem para o *DPManagerService* informando que está ativo e, em seguida, o *DPManagerService* publica duas mensagens: a primeira com os dados do *broker* e a segunda com o modo de composição do fenótipo digital. O *PhenotypeComposer* recebe os dados para estabelecer conexão com o *broker* externo e configurar o modo de composição do fenótipo digital.

4.3.2 *DataProcessor*

O *DataProcessor* é o componente utilizado para criar os módulos de processamento de dados (e.g., sociabilidade, atividade física) no framework. O objetivo principal de um *DataProcessor* é usar dados dos sensores para gerar informações de alto nível. O diagrama de classe mostrado na Figura 18 apresenta o componente *DataProcessor*. O *DataProcessor* é uma classe abstrata e estendida da classe *Service* do Android. Devido a essa característica, foram criados três métodos: *init()*, *do()* e *end()*, representando, respectivamente, o início, o processamento e o fim do processamento de dados. O método *init()* é executado quando o método *onCreate()* é chamado na classe *Service* do Android. Ele deve conter o necessário para inicializar os módulos de processamento de dados, como: definir o nome do módulo, selecionar os sensores para receber os dados brutos. O método *do()* é executado toda vez que o método *onStartCommand()* é chamado na classe *Service* do Android. É nele onde é realizado o processamento dos dados de contexto. O desenvolvedor que usa *OpenDP* pode aplicar diferentes técnicas para inferir informações de alto nível, tais como: de regras fuzzy, processamento de eventos complexos e aprendizado de máquina. O método *end()* é chamado pelo método *onDestroy()*.

Todos os dados coletados dos sensores são recebidos pelo método *onSensorData-*

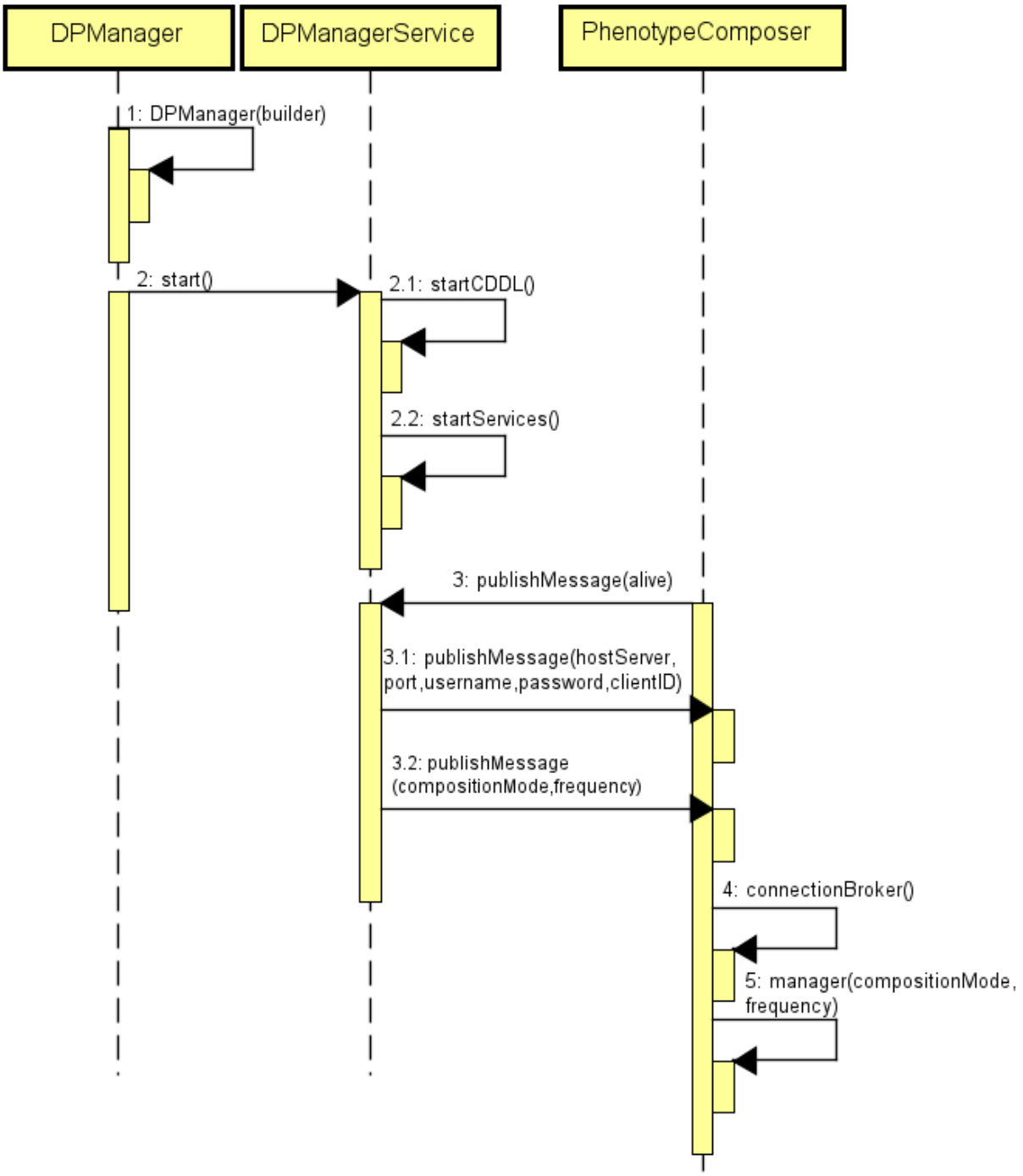

Figura 17 – Diagrama de sequência para iniciar o framework.

Arrived(message) através do parâmetro *message* (i.e., classe de metadados herdada do CDDL (GOMES *et al.*, 2017)). De posse dos dados, o desenvolvedor pode usar o método *inferencePhenotypingEvent(message)* ou *do()* para implementar a lógica de inferência, a diferença que o método *do()* é executado ao término do método *onStartCommand()* da classe *Service* do Android e o *inferencePhenotypingEvent(message)* não.

O método *saveDigitalPhenotypeEvent(digitalPhenotypeEvent)* salva em banco de dados no *smartphone* as informações de alto nível geradas pelos módulos de processamento de dados. Uma vantagem em usar esse método, é que podem ser recuperadas as informações

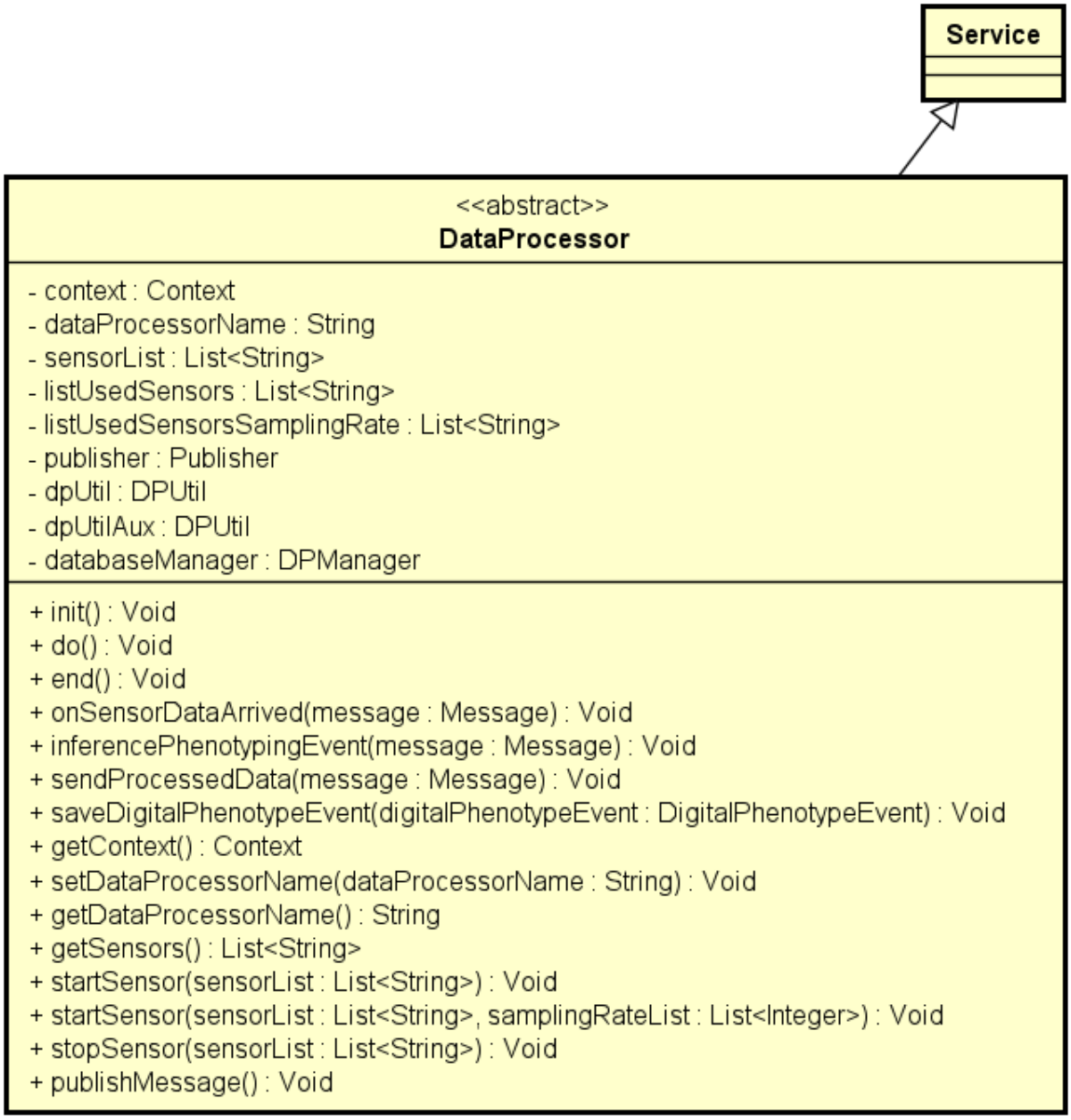

Figura 18 – Diagrama de classe do *DataProcessor*.

de alto nível e apresentadas na *interface* do usuário usando o método *getPhenotypeList()* do *DPManager*. Para compor fenótipos digitais, o componente *PhenotypeComposer* recebe os dados no formato *DigitalPhenotypeEvent* (mostrado na Seção 4.3.3). Para isso, o desenvolvedor modela as informações de alto nível no formato *DigitalPhenotypeEvent*, definindo os valores. Para iniciar um sensor no framework, o desenvolvedor identifica os sensores (físicos e virtuais) do *smartphone* disponíveis através do método *getSensors()* que retorna uma lista dos nomes dos sensores. Depois o desenvolvedor sabendo dos nomes dos sensores, utiliza o método *startSensor()* para *started* os sensores.

O diagrama de sequência apresentado na Figura 19 mostra o processo de inicialização de um *DataProcessor*. O processo de inicialização de um *DataProcessor* começa pelo

componente *DPManager* recebendo os nomes dos módulos de processamento de dados para serem inicializados. Depois o desenvolvedor envia uma lista com os nomes dos *DataProcessor* que devem ser inicializados para o *ProcessorManager*. Logo após receber a lista, o *ProcessorManager* verifica se o nome do módulo é válido e está disponível para execução no framework, e caso esteja disponível para execução, coloca-o em execução. Em seguida, o *ProcessorManager* atualiza a lista de módulos de processamento ativos em um banco de dados. Após salvar a lista dos módulos que estão em execução no framework, o desenvolvedor publica uma mensagem para o *PhenotypeComposer* com o nome do módulo para atualizar a lista dos módulos para controle interno. O *DataProcessor* é executado, o método *init()* é usado para configurar os sensores (e.g., selecionar os sensores que serão ativados), informando para o *ProcessorManager* a lista de sensores que deseja utilizar. Internamente, o *ProcessorManager* é responsável por gerenciar (inicia, parar) tanto os componentes *DataProcessor* quanto os sensores utilizados por eles. Caso o framework identifica um determinado sensor já está em uso por outro *DataProcessor*, ele gera uma exceção para o *DataProcessor* informando que aquele está sendo utilizado.

Após o *DataProcessor* executar o método *init()*, ativando os sensores que deseja coletar os dados, o *PhenotypeComposer* está apto a compor os fenótipos digitais. O *DataProcessor* recebe os dados dos sensores quando chegam através do método *onSensorDataArrives(message)*. Em seguida, o desenvolvedor implementa a lógica de negócio nos métodos *do()* ou *inferencePhenotypeEvent(message)*.

Usando o método *sendProcessedData(message)* para enviar o resultado do processamento para o componente *PhenotypeComposer* compor os fenótipos digitais conforme o modo de composição (*frequency, group_all* ou *send_when_it_arrives*) previamente definido pelo desenvolvedor. Essas informações de alto nível já processadas recebidas do *PhenotypeComposer* têm a finalidade de compor o fenótipo digital, e serão distribuídas para o *broker* externo de acordo com o modo de composição. O método *saveDigitalPhenotypeEvent(digitalPhenotypeEvent)* é usado para salvar no banco de dados local as informações de alto nível inferidas pelos módulos.

Os modos de composição de fenótipos digitais utilizados no *PhenotypeComposer* foram implementados da seguinte forma:

- *send_when_it_arrives*: o componente *PhenotypeComposer* recebe as informações de alto nível dos *DataProcessor*, são imediatamente distribuídos para o *broker* externo. Nesse modo, o *PhenotypeComposer* funciona como um gateway para a rede externa;
- *group_all*: ao receber as informações de alto nível dos *DataProcessor*, apenas serão distribuídos para o *broker* externo caso receba informações de alto nível de todos os *DataProcessor* ativos. Entre os tempos de distribuição, o *PhenotypeComposer* armazena as informações de alto nível no banco de dados local;

Figura 19 – Diagrama de sequência para iniciar um *DataProcessor*.

- *frequency*: o modo de composição *frequency* contém um *schedule* (segundo, minutos, horas) que ao disparar, recupera do banco de dados todas as informações de alto nível e distribui para o *broker* externo. Entre os tempos de distribuição, o *PhenotypeComposer* armazena as informações de alto nível no banco de dados local.

4.3.3 *DigitalPhenotypeEvent*

A Figura 20 exibe a representação de informações de alto nível inferidas pelos *DataProcessor*, podendo ser chamado evento de fenotipagem digital (e.g., ligação telefônica,

troca de mensagens de texto, estado físico do usuário). Antes de detalhar como representar um fenótipo digital, é importante definir o que são atributos e situações no framework. Os atributos são propriedades que caracterizam um evento de fenotipagem digital. *Timestamp* e coordenadas de localização são exemplos de atributos de eventos de reconhecimento de estado físico do usuário. Os transtornos mentais causam mudanças comportamentais. Essas mudanças comportamentais são situações de interesse para profissionais de saúde mental e pesquisadores utilizarem para o monitoramento do bem-estar ou contribuir no tratamento do transtorno mental. Correndo, caminhando, dirigindo são possíveis estados físicos dos usuários que representam situações de interesse para profissionais de saúde mental e pesquisadores. Através do modelo apresentado na Figura 20 é possível representar os eventos de fenotipagem digital, e o conjunto desses objetos vão compor o fenótipo digital do usuário (*DigitalPhenotype*).

Portanto, o objeto *DigitalPhenotypeEvent* podem representar os eventos de fenotipagem digital, como: identificador do usuário (*uid*), nome do módulo de processamento de dados *dataProcessorName*, momento da ocorrência do evento (*startDateTime, endDateTime*), e o conjunto de objetos *DigitalPhenotypeEvent* compõe um *DigitalPhenotype* que representa o fenótipo digital. O objeto *Attribute* podem representar as propriedades dos eventos de fenotipagem digital, como: nome do atributo, valor, tipo. O objeto *Situation* pode representar a situação de interesse, como: nome da situação de interesse (*label*), descrição.

4.3.4 *RawDataCollector*

O componente *RawDataCollector* também pode ser utilizado para distribuir dados de contexto de origem dos sensores dos dispositivos móveis para o *broker* externo, conforme pode ser observado na Figura 21. O componente *RawDataCollector* é estendido da classe *Service* do Android. Ele é iniciado através do componente *DPManagerService*. Para executar o *RawDataCollector*, é necessário fornecer os nomes dos sensores para iniciar a coleta de dados e o modo de composição de dados de contexto. O *RawDataCollector* disponibiliza dois modos de composição: o *send_when_it_arrives* e o *frequency*. A seguir são apresentados:

- *send_when_it_arrives*: o componente *RawDataComposer* recebe os dados de contexto dos sensores, os quais são imediatamente distribuídos para o *broker* externo;
- *frequency*: o modo de composição *frequency* contém um *schedule* (segundo, minutos, horas) que, ao disparar, recupera do banco de dados todos os dados de contexto e distribui para o *broker* externo. Entre os tempos de distribuição, o *RawDataComposer* armazena os dados de contexto no banco de dados local.

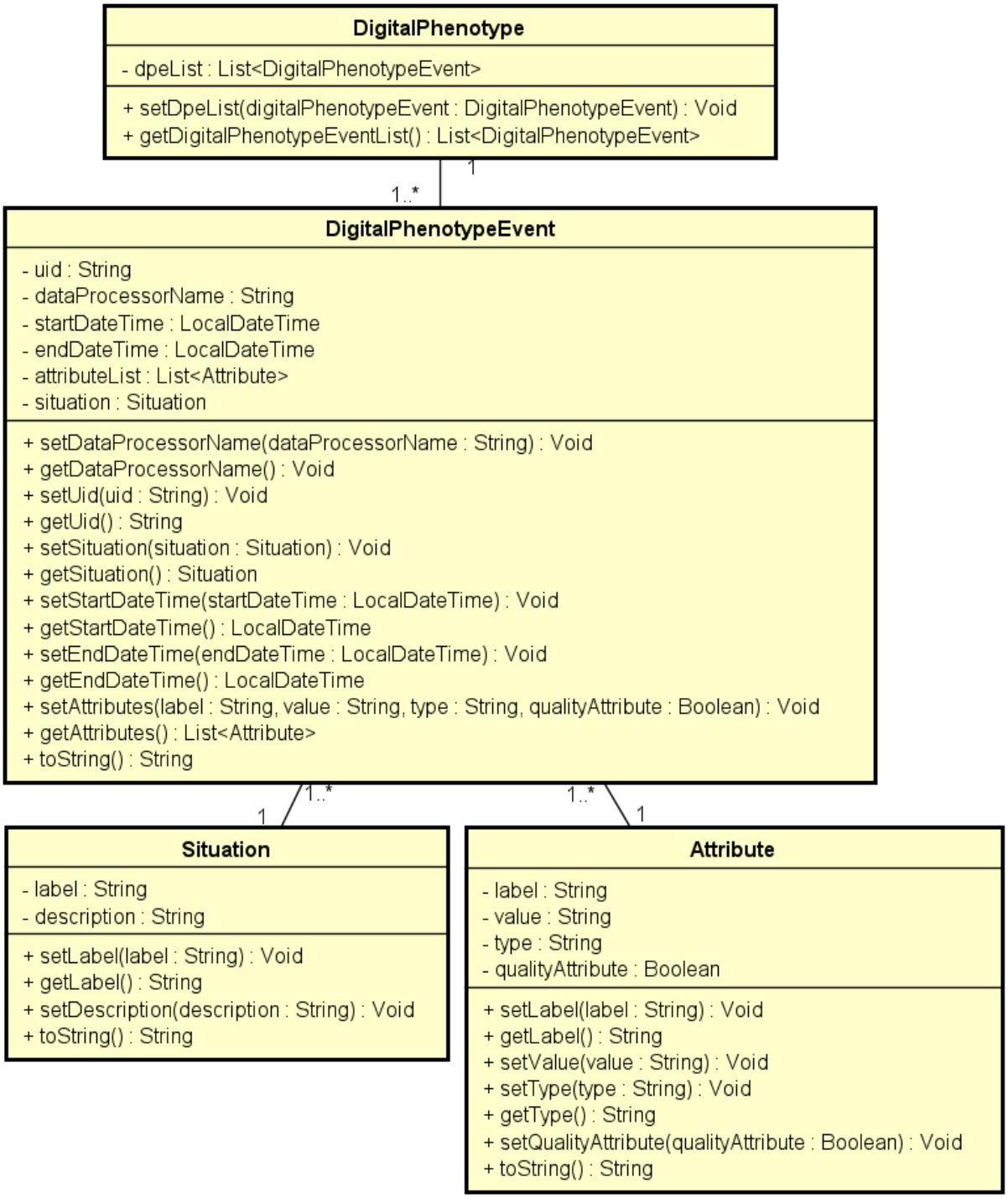

Figura 20 – Representação de fenótipos digitais no framework.

Para distribuir, modelar e detectar eventos complexos sobre o fluxo de dados gerados pelos sensores, o componente *RawDataCollector* utiliza o EPL/CEP (GOMES *et al.*, 2017). O EPL fornece uma linguagem semelhante a SQL com cláusulas *SELECT*, *FROM*, *WHERE*, *GROUP BY*, *HAVING* e *ORDER BY*, podendo implementar conceitos de associação, filtragem e agregação.

Para exemplificar o uso da linguagem EPL, suponha que um determinada aplicação ciente de contexto monitore a temperatura corporal de um usuário através de um sensor.

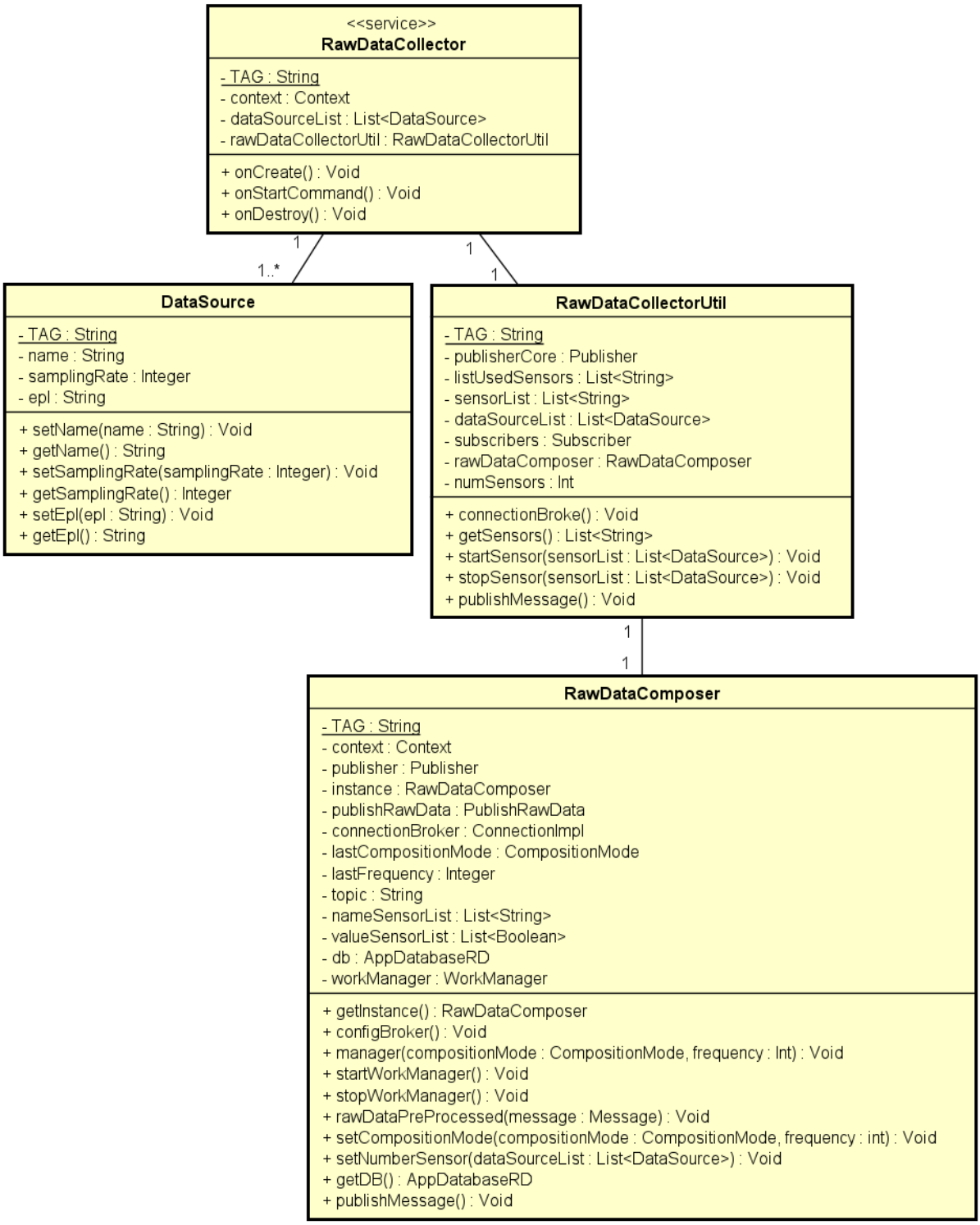

Figura 21 – Diagrama de classes do *RawDataCollector* e seus componentes.

Usando a EPL é possível criar a seguinte regra: avisar quando três leituras seguidas estiverem acima de 38 graus. A seguir está a regra criada:

Quadro 4.1 – Exemplo de regra utilizando EPL

```
1 select * from TemperatureEvent
2 match_recognize (
```

```
3       measures A as temp1, B as temp2, C as temp3
4       pattern (A B C)
5       define
6       A as A.temperature > 38,
7       B as B.temperature > 38,
8       C as C.temperature > 38)
```

O diagrama de classe do *RawDataCollector* na Figura 21 mostra sua relação com os demais componentes para compor os dados de contexto e distribuir para o *broker* externo. O componente *DataSource* é responsável por representar os sensores e as EPL. O componente *RawDataCollectorUtil* é responsável por abrir uma conexão com o *broker* externo, ativar os sensores dos dispositivos a partir da lista de *DataSource*, e o componente *RawDataComposer* é responsável por receber os dados de sensores (são pré-processados pela EPL) de origem do *RawDataCollector* para compor dados de contexto.

A Figura 22 mostra como inicia o componente *RawDataCollector* através do diagrama de sequência. Antes de executar o serviço *RawDataCollector*, o desenvolvedor precisa configurar os sensores informando os nomes, as EPL e o modo de composição. O componente *DPManagerService* inicia o serviço *RawDataCollector*, o método *onCreate()* no *RawDataCollector* é executado para que o framework identifique quais os sensores, as EPL e o modo de composição será usado. Em seguida, o método *onStartCommand()* é executado criando uma instância do componente *RawDataCollectorUtil* para estabelecer um canal de comunicação com o *broker* externo, enviar a lista de sensores para o componente *ProcessorManager* iniciar os sensores e criar *subscriber* para o *RawDataCollectorUtil* possa receber os dados dos sensores. O componente *RawDataCollectorUtil* acessa o padrão *builder* do *DPManager* para obter *host*, *port* e outras informações para conectar ao *broker* externo. O método *onMessageArrived(message)* (momento que a *epl* é aplicada sobre o fluxo de dados) recebe os dados dos sensores e envia para o componente *RawDataComposer* compor os dados de contexto. O método *rawDataPreProcessed(message)* no componente *RawDataComposer* recebe os dados dos sensores, e conforme o modo de composição distribui para o *broker* externo.

4.4 Aspectos de Implementação dos Componentes do *Plugin*

Nesta seção mostraremos um recurso de extensibilidade do framework, em que o desenvolvedor pode adicionar novos módulos de processamento de dados, sem a necessidade de recompilar o projeto ou mesmo interromper a coleta dos dados dos usuários. Pelo fato do framework utilizar uma comunicação *publisher/subscriber*, isso permite a inserção de *plugin*.

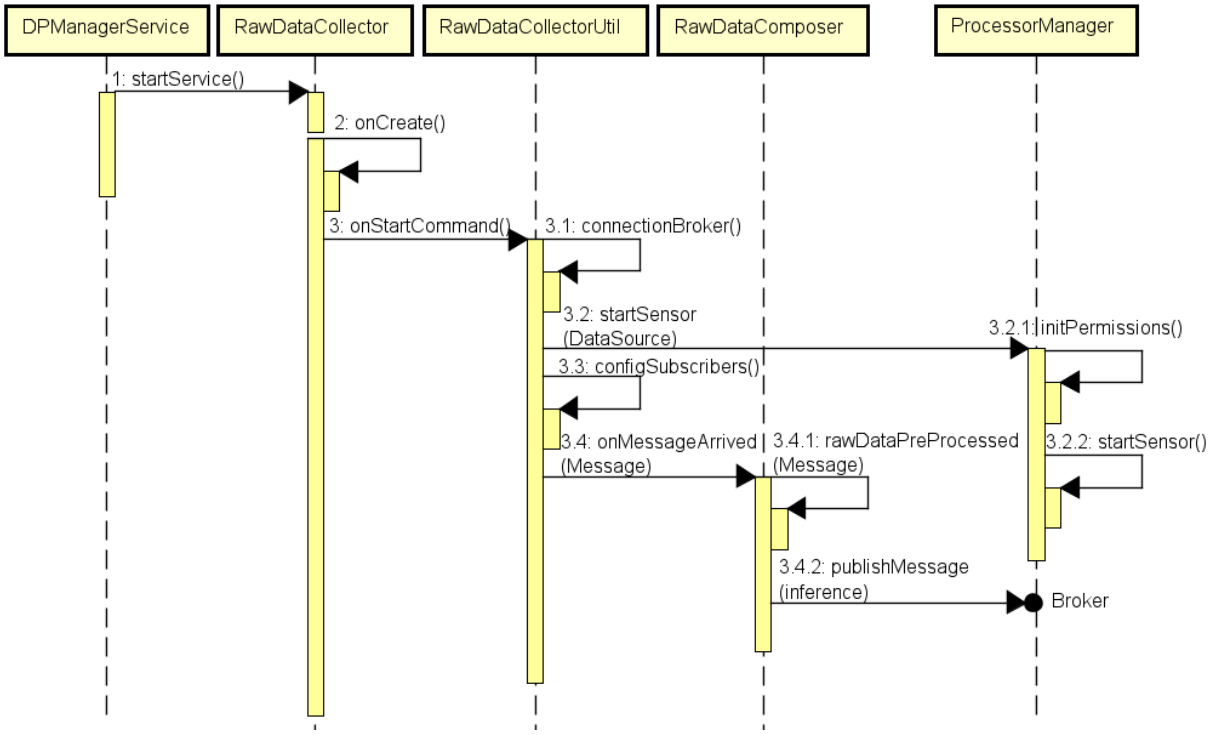

Figura 22 – Diagrama de sequência para iniciar um RawDataCollector.

4.4.1 *PluginManager*

O diagrama de classe do *PluginManager* mostrado na Figura 23 apresenta os componentes com os quais ele tem relacionamento. A arquitetura do *plugin* possui poucos componentes. Logo, o *plugin* é responsável por informar ao *core* a lista dos módulos de processamento implementados, informar os sensores que deseja utilizar para receber os dados de contexto e devolver para *core* o resultado da inferência, os quais irão compor os fenótipos digitais no *core*, conforme mostra a Figura 15. O componente *DPManagerService* é um serviço Android. Assim como no *core*, ela possui uma conexão interna do CDDL. Portanto, quando o *plugin* é executado, ele tenta estabelecer uma conexão com o *micro broker* previamente criado pelo *core*. Caso não consiga ou a conexão caia, é gerada uma exceção informando o ocorrido para o desenvolvedor. O *PluginManager* também é responsável por enviar uma lista interna dos módulos de processamento de dados (*startProcessor(List<String>processorName)*) para o *core*, ele envia a lista para deixar disponíveis para uso, caso o usuário ou profissional de saúde deseje, ele pode iniciar.

Ao iniciar a aplicação do *plugin*, é executada o componente *PluginManager*, a qual se encarrega de iniciar o serviço *DPManagerService*. O componente *DPManagerService* conecta-se ao *micro broker* previamente já criado pelo *core*, ou seja, o *core* precisa estar funcionando para que o *plugin* possa se conectar ao *micro broker*. Logo os dados para configuração do *micro broker* devem estar previamente configurados no serviço *DPManagerService* do *plugin* (e.g., *hostServer*, *clientID*, *port*, lista com os nomes dos módulos de processamento de dados no *plugin*). Assim como o serviço *DPManagerService*

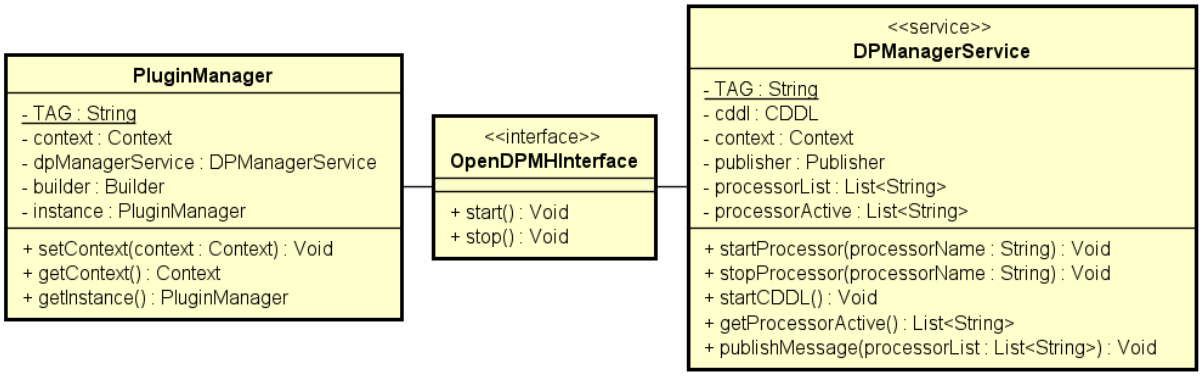

Figura 23 – Diagrama de classe do *PluginManager*.

do *core* foi colocado em primeiro plano, houve a necessidade de colocar também o serviço *DPManagerService* do *plugin*, com o mesmo objetivo, evitar que o Android derrube o serviço do *plugin*. Dessa forma, o usuário pode fechar a aplicação do *plugin*, e o serviço continuará sua execução em segundo plano. Só irá finalizar o *plugin* caso o desenvolvedor desejar chamar o método *stop()*.

4.4.2 *DataProcessor*

O *DataProcessor* da arquitetura do *plugin* mostrado na Figura 24 tem a mesma funcionalidade do *DataProcessor* do *core* apresentado na Seção 4.3.2. O *DataProcessor* do *plugin* recebe os dados de contexto vindos dos sensores e gera informações de alto nível, porém, o resultado da inferência, é enviado para o *core* compor os fenótipos digitais. Portanto, o desenvolvedor pode implementar vários módulos de processamento de dados em um *plugin*, apenas garantindo que os nomes dos módulos do *plugin* não sejam iguais.

A Figura 25 mostra o diagrama de sequência para iniciar um *DataProcessor* no *plugin*. Depois que o *PluginManager* é executado, ele se encarrega de iniciar o serviço *DPManagerService* do *plugin*, previamente já está configurado com os dados do *hostServer*, o *clientID*, número da porta, e a lista com os nomes dos módulos de processamento de dados brutos. Dessa forma, o *DPManagerService* se encarrega de iniciar a conexão com o *micro broker*. Em seguida, o *DPManagerService* envia para o *core* a lista com os nomes dos módulos de processamento de dados brutos. De posse da lista com os nomes, o *core* pode ou não selecionar os módulos (que estão implementados no *plugin*) deseja ativar e devolve para o *plugin* colocar em execução.

O *plugin*, ao finalizar a ativação dos módulos, dispara uma mensagem carregando os nomes dos módulos ativados para o *ProcessorManager* do *core* gerenciar. No *DataProcessor* é chamado o método *init()*, que fica responsável por passar a lista de sensores que deseja ativar para o *ProcessorManager* do *core* poder ativá-los. Os sensores então são ativados e o framework começa a receber os dados dos sensores. No *plugin*, o método responsável por receber os dados é o *onSensorDataArrived(message)*. Fica a critério

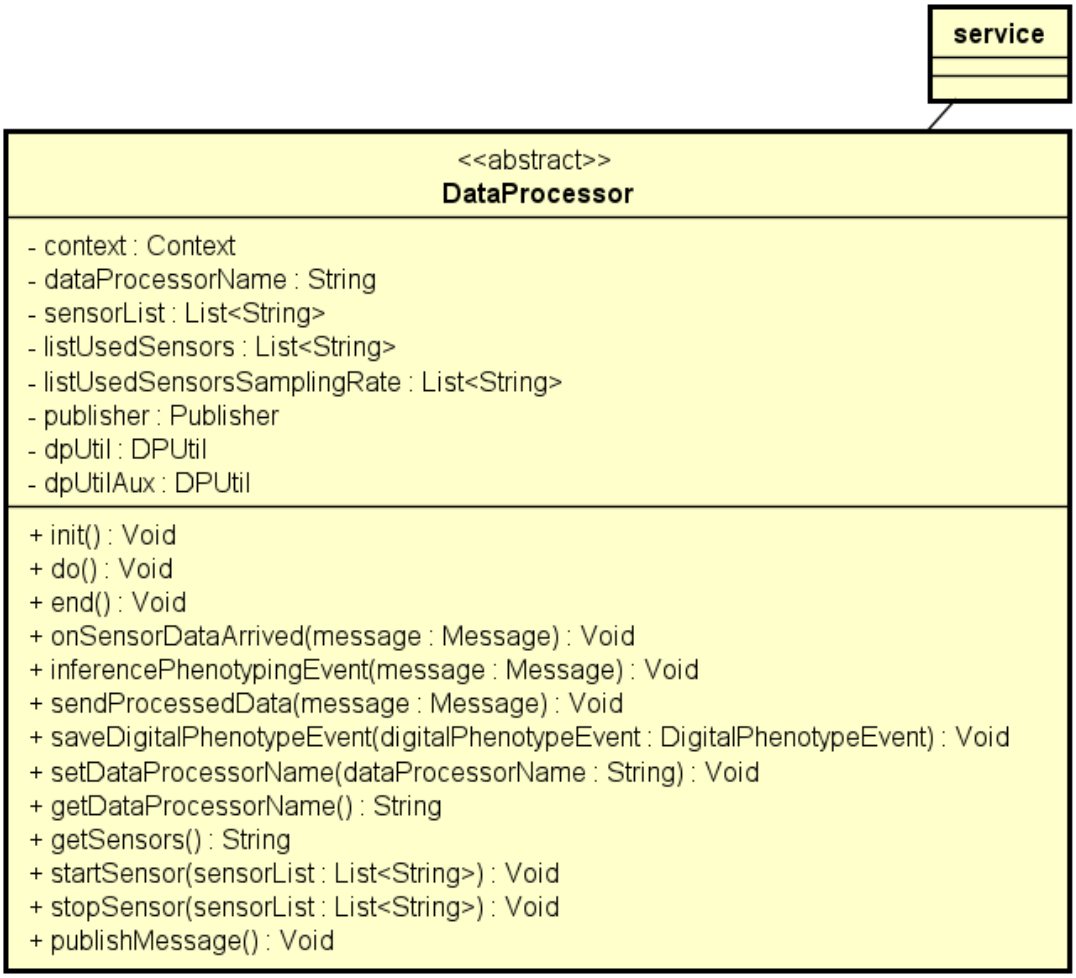

Figura 24 – Diagrama de classe do *DataProcessor* no *plugin*.

do desenvolvedor usar os métodos *do(message)* ou *inferencePhenotypeEvent(message)* para implementar a regra de negócio e inferir informações de alto nível. Em seguida, o método *sendProcessedData(message)* é usado para enviar o resultado das inferências para o *PhenotypeComposer* no *core*. Finalmente, o *core* compõe os fenótipos digitais conforme o modo de composição dos fenótipos configurado. No *plugin*, o desenvolvedor pode usar o método *saveDigitalPhenotypeEvent(digitalPhenotypeEvent)* para salvar as informações de alto nível geradas para poderem ser visualizadas pelo usuário na *interface* gráfica.

4.5 Segurança no *OpenDP*

Ao desenvolver novas aplicações de fenotipagem digital através do framework *OpenDP*, é necessário que elas consigam garantir a segurança da informação, especialmente quando dados sensíveis estão envolvidos. É crucial garantir mecanismos de segurança na distribuição dos dados em aplicações de fenotipagem digital. Para isso, encapsulamos no framework o serviço de segurança do *middleware* CDDL.

A Figura 16 apresenta o diagrama de classe e os métodos para ativar o serviço

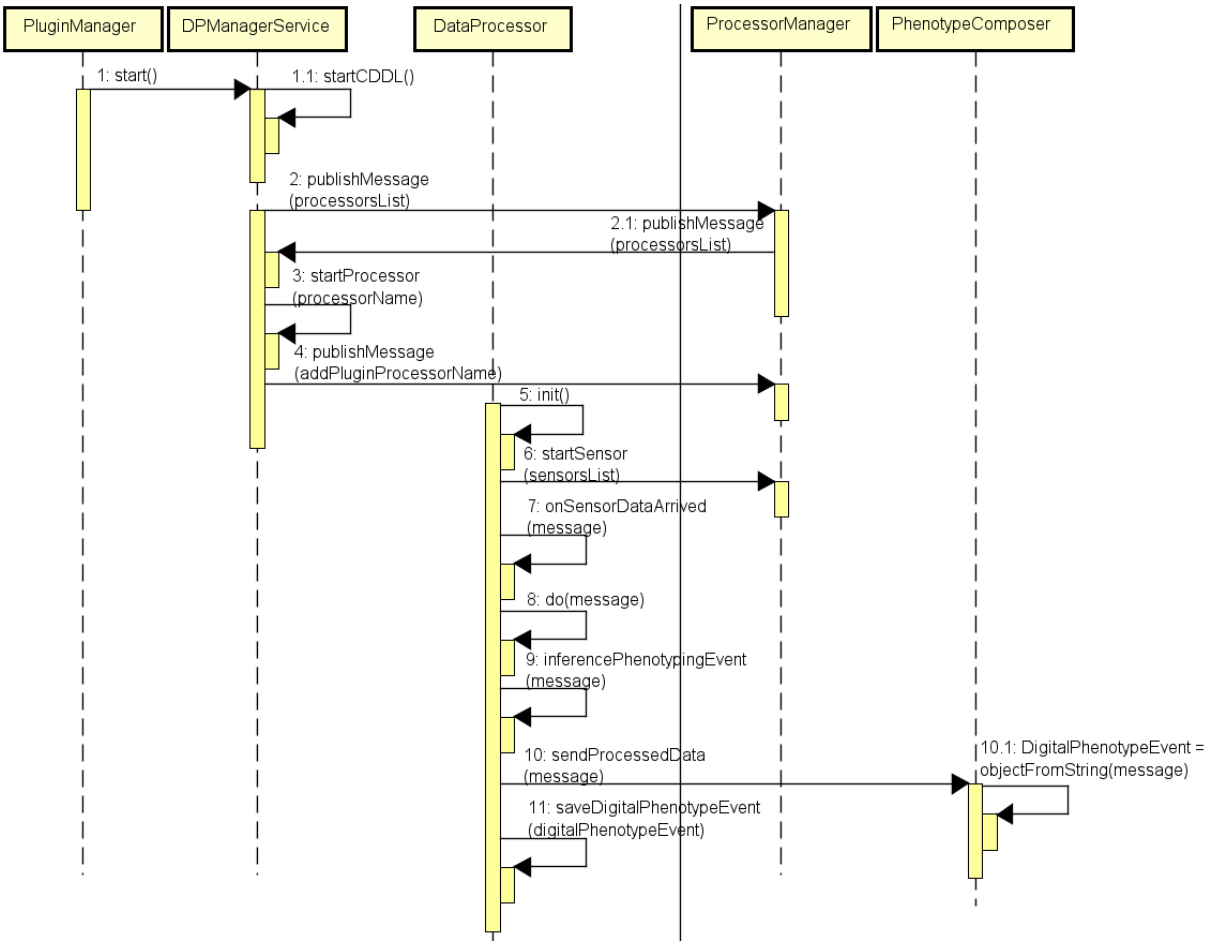

Figura 25 – Diagrama de sequência para iniciar um *DataProcessor* no *plugin*.

de segurança no framework. Todo o processo para ativar o serviço de segurança do framework inicia pelo *DPManager*. Para isso, fornecemos a interface *DPInterface* que contém as assinaturas dos métodos necessários para o desenvolvedor. O componente *DPManagerService* é responsável por iniciar os serviços do framework depois que finalizar os procedimentos de inicialização do serviço de segurança. O componente *PhenotypeComposer* é responsável por abrir a conexão segura com o *broker* externo. Previamente, o certificado digital já deve ter sido gerado, autorizado e assinado pela Autoridade Certificado (CA).

A Figura 26 mostra como é ativado o serviço de segurança no framework. Para que o serviço de segurança seja utilizado, o desenvolvedor precisa gerar os certificados digitais devidamente autenticados e assinados pela CA antes de iniciar o framework.

O componente *DPManager*, ao ser iniciado, usa no construtor o padrão *builder* para receber as informações (e.g., *Country, State, Locality*) para configurar o certificado digital, como pode ser visto na Figura 26. Em seguida, é checado se o certificado digital já foi criado. Se está criado, segue para a inicialização do framework. Caso contrário, inicia o processo de configuração do certificado digital, em que é verificado se o usuário deu permissão para acessar os arquivos no *smartphone* para buscar o certificado digital assinado pela autoridade certificadora. A seguir, é gerado o certificado digital (não assinado) com a

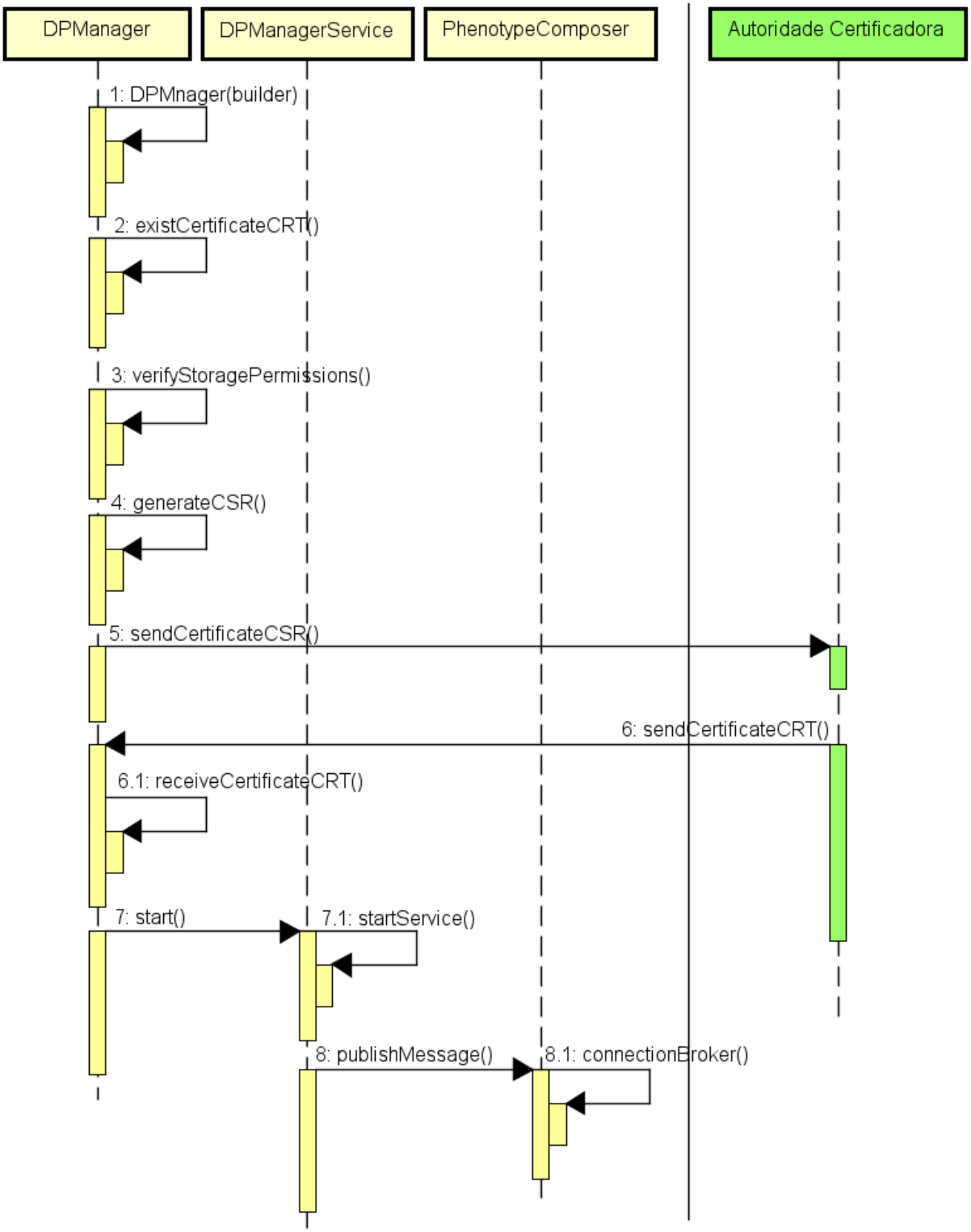

Figura 26 – Diagrama de sequência ativando Segurança no *OpenDP*.

requisição para a autoridade certificado autenticar e assinar.

O desenvolvedor pode usar os métodos *sendCertificateCSR()* e *receiveCertificate-CRT()* para implementar a conexão com a autoridade certificado de forma automática, enviando e recebendo o certificado digital. O desenvolvedor seleciona o certificado digital com a requisição no *smartphone* e envia para CA assinar. Após autenticação do certificado

digital pela CA e devidamente assinado, o desenvolvedor insere o certificado digital no armazenamento externo do dispositivo móvel. Só depois que o certificado digital devidamente assinado pela autoridade certificadora está no armazenamento externo do dispositivo móvel, o framework é inicializado. O componente *DPManagerService* então inicializa os serviços do framework e envia uma mensagem com as informações (e.g., *hostServer, port*) para o *PhenotypeComposer* estabelecer uma conexão segura com o *broker* externo, só então que o componente *PhenotypeComposer* conecta com o *broker* externo.

Para evitar que uma aplicação maliciosa intercepte a comunicação entre o *core* e o *plugin*, também é necessário abrir um canal de comunicação segura entre eles, logo, o *plugin* terá que iniciar o serviço de segurança. Para o *plugin* iniciar o serviço de segurança, são usados os mesmos métodos utilizados pelo *core*, conforme pode ser observado na Figura 26. O *plugin* usa também o padrão *builder* para receber as informações para configurar o certificado digital, em seguida verifica se já foi criado o certificado digital. Caso não tenha sido criado, o framework envia uma solicitação de permissão para o usuário liberar o acesso ao armazenamento externo do *smartphone*. Em seguida, o certificado digital é gerado com a requisição, a CA assina e autentica o certificado digital, o desenvolvedor salva o certificado assinado no armazenamento externo do *smartphone* para que o framework possa encontrar. Dessa forma, está configurado o serviço de segurança no *plugin* e pronto para estabelecer um canal de comunicação segura com o *core*.

5 Estudos de Caso e Avaliação Experimental

Neste capítulo, iremos demonstrar que a solução proposta pode ser aplicada ao desenvolvimento de aplicações móveis de fenotipagem digital. Para isso, implementamos estudos de caso baseados em cenários de aplicação usando o framework no dia-a-dia para coletar dados e inferir informações de alto nível para compor fenótipos digitais. O primeiro estudo de caso usando somente o *core*, e no segundo estudo de caso adicionando o *plugin*. Em seguida, realizamos avaliações experimentais de consumo de energia do dispositivo móvel, analisando o impacto do uso do framework ao longo do dia. As avaliações foram baseadas em cenários de teste: a primeira avaliação foi feita usando apenas o *core* e variando os modos de composição de fenótipos digitais; na segunda avaliação, adicionamos o *plugin*; por fim, na terceira, habilitamos o uso de segurança no framework.

5.1 Estudos de Caso

Esta seção descreve a implementação de duas aplicações com o uso do framework, usadas como estudos de caso. Um estudo de caso com o uso somente do *core*, e outro com a adição do uso do *plugin*. Nos dois estudos de caso, o objetivo é demonstrar os mecanismos providos pelo framework para o desenvolvimento de aplicações móveis de fenotipagem digital. A ideia é analisar o comportamento da solução proposta por sua aplicação em um domínio específico, então verificando a sua adequação para a fenotipagem digital.

5.1.1 Estudo de Caso 1: Usando somente o *Core*

Os transtornos mentais causam mudanças comportamentais que representam um indicador relevante de início, presença ou desenvolvimento de transtornos mentais (MOURA *et al.*, 2021). Essas mudanças comportamentais são situações de interesse para profissionais de saúde mental, visto que são usados com base para a realização de avaliações e intervenções (MONITORAMENTO. . . , 2020). Em particular, as mudanças de comportamento social podem representar indicadores relevantes de transtornos mentais, pois a sociabilidade dos indivíduos tem implicações significativas para seu estado de bem-estar (UMBERSON; MONTEZ, 2010). As características das relações sociais podem representar aspectos capazes de proteger ou contribuir para o desenvolvimento de transtornos mentais. Por exemplo, há evidências de que o suporte social é um fator relevante para a saúde mental (GRAV *et al.*, 2012; FAUTH *et al.*, 2012), dado que existe maior probabilidade de depressão entre pessoas que não tem apoio social. Há também evidências de que o isolamento social está associado a problemas de saúde mental, como depressão, ansiedade e ideação suicida (BEUTEL *et al.*, 2017). Além disso, o isolamento social imposto para reduzir a taxa de contágio pelo

coronavírus (COVID-19) pode impactar ainda mais a saúde mental global (TORALES *et al.*, 2020). Portanto, os sintomas de transtornos mentais podem ser externalizados por mudanças nos comportamentos sociais, caracterizando assim uma situação de interesse para o acompanhamento da saúde mental. Para implementar o primeiro estudo de caso, consideramos a sociabilidade como situação de interesse em um cenário ilustrativo descrito a seguir.

Uma psicóloga chamada Andressa de Moura utilizou uma aplicação desenvolvida com o framework para entender os comportamentos e hábitos de uma usuária chamada Maria Azevedo. Na consulta (primeira sessão), a psicóloga coletou informações sobre a usuária através de auto-relatos. Maria informou morar em São Luís-MA, tinha formação superior em Administração, casada, mãe de um filho, e trabalhava em uma empresa privada como secretária há um ano. Ela ainda informou que a razão pela qual estava a procura de terapia era devido a estresses pessoais, conflitos no relacionamento, e também problemas profissionais. Maria descreveu os sintomas físicos, mentais e emocionais, os quais indicavam ansiedade, depressão, e também mudanças de hábitos alimentares. A partir das informações obtidas, a psicóloga decidiu utilizar a aplicação de fenotipagem digital para, inicialmente, entender melhor o comportamento de sociabilidade de Maria, pois julgou ser um aspecto importante a ser analisado. Andressa explicou a Maria que utilizaria uma aplicação móvel para dar apoio à terapia, e que nenhuma informação privada coletada (e.g., chamadas telefônicas e mensagens de texto) seria exposta. Maria então concordou e autorizou o monitoramento, permitindo a instalação da aplicação em seu smartphone. A partir da segunda sessão semanal, a psicóloga passou a analisar os resultados alcançados com o emprego da aplicação de fenotipagem digital.

Uma aplicação foi desenvolvida para o primeiro estudo de caso, denominada *Sociability Monitoring Mobile System (SMMS)*, conforme pode ser observado na Figura 27, uma arquitetura instanciada do framework. Foram implementados dois módulos de processamento de dados: o *Physical_Sociability* e o *Online_Sociability* usando o framework *OpenDP*, os quais compuseram os fenótipos digitais. A aplicação consistiu de dois módulos (TELES *et al.*, 2020). O primeiro chamado de *Online_Sociability* tinha a finalidade de identificar a socialização *online* do usuário através da coleta de dados dos sensores de chamadas telefônicas e de troca de mensagens de texto (SMS). O framework *OpenDP* já disponibiliza esses sensores para coletar os dados. Qualquer evento desses dois sensores virtuais indica que o usuário do aparelho móvel está socializando ou tem a intensão de socializar, logo, basta monitorar os eventos detectados pelos sensores, tais como: recebimento e início de chamadas telefônicas, recebimento e envio de SMS. O segundo módulo de processamento de dados implementado foi o *Physical_Sociability*, o qual teve a finalidade de identificar se o usuário está socializando presencial. Através do microfone, é feita a captura do áudio e inferido se o áudio apresenta voz humana. Caso encontre voz humana, dispara-se uma mensagem alertando que foi encontrada voz humana nas proximidades do dispositivo

móvel utilizado para coletar os dados. Para implementar o módulo *Physical_Sociability*, utilizamos a biblioteca *VAD* (KONOVALOV, 2022) que recebe amostras de áudio e analisa se tem voz humana.

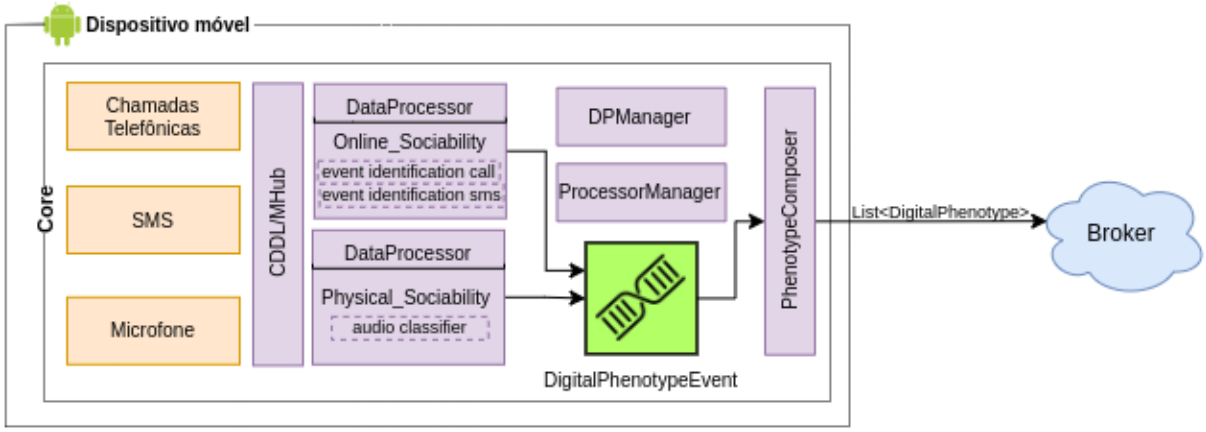

Figura 27 – Estudo de caso 1: arquitetura da aplicação *SMMS* implementada com o *OpenDP*.

A Figura 27 mostra o primeiro estudo de caso com a arquitetura da aplicação *SMMS* implementada com o uso do *OpenDP*. No *core* foram implementados os dois módulos de processamento: o *Online_Sociability* e o *Physical_Sociability*. Cada um deles recebe dados dos sensores aos quais selecionaram para serem ativados pelo *ProcessorManager*. A medida que os módulos de processamento de dados inferem as informações de alto nível, são gerados eventos de fenotipagem digital (i.e., *DigitalPhenotypeEvent*). A classe *PhenotypeComposer* recebe esses eventos com informações de alto nível para compor os fenótipos digitais, o modo de composição do fenótipo escolhido pelo desenvolvedor foi *frequency*, com intervalo de distribuição de dados para o *broker* externo de 15 minutos. O modo *frequency* com intervalo de 15 minutos foi escolhido com a intenção distribuir os dados com maior frequência, diferente dos intervalos 30, 45, e 60 que o intervalo de distribuição é menor. Por fim, o *PhenotypeComposer* tem uma lista de fenótipos guardada em *List<DigitalPhenotype>* a ser distribuída para o *broker*.

A Figura 28 mostra as primeiras telas da aplicação *SMMS* implementadas com os recursos do framework *OpenDP*. A Figura 28a mostra a tela inicial da aplicação *SMMS* com a porcentagem da bateria (dado usado nas avaliações experimentais de consumo de energia), o identificador do usuário no CDDL, o *status* do framework (ligado/desligado), um botão de liga/desliga do framework, algumas informações dos módulos de processamento de dados (i.e., situações de interesse) e, na parte inferior da tela, o botão de adicionar módulos de processamento de dados. A Figura 28b apresenta a tela de configuração do framework com algumas definições: endereço IP do *broker* externo, o número da porta, o identificador do usuário no CDDL, e o modo de composição do fenótipo digital. A Figura 28c descreve as situações de interesse disponíveis para uso, as quais representam módulos de processamento em execução.

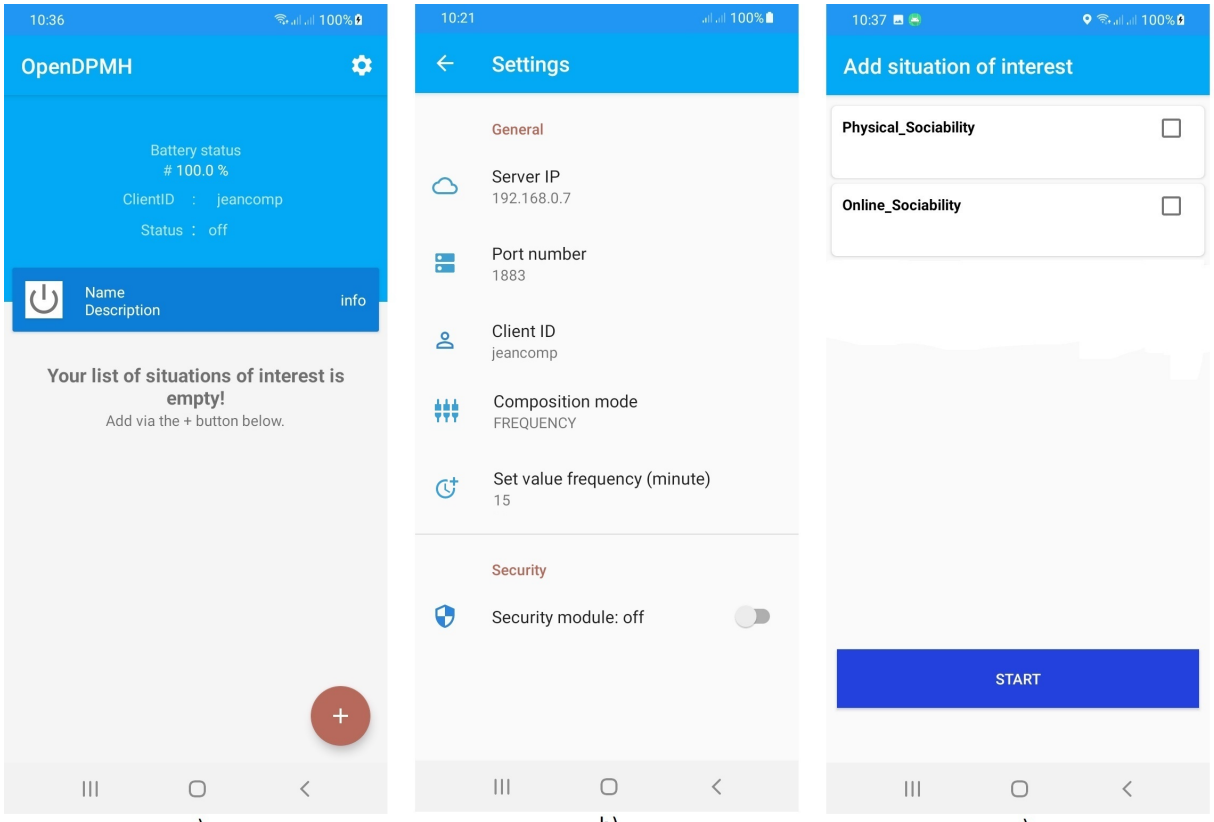

Figura 28 – Primeiras telas da aplicação *SMMS*.

A Figura 29 mostra as segundas telas da aplicação *SMMS* usando o *OpenDP*. A Figura 29 é a continuação da Figura 28 executando a aplicação *SMMS*. Após selecionar os módulos *Online_Sociability* e o *Physical_Sociability* para iniciar sua execução, a Figura 29a mostra os módulos que estão ativos. As telas da Figura 29b e c descrevem um resumo para acompanhar o que está sendo coletado. A Figura 29b mostra um resumo dos registros dos eventos do módulo *Physical_Sociability*: quando foi a última ocorrência do evento com data e hora, um gráfico mostrando as ocorrências por dia (eixo x dia da ocorrência e eixo y total da ocorrência), e o botão (cor laranja) para finalizar o módulo de processamento de dados. Já a Figura 29c apresenta um resumo do módulo *Online_Sociability* separando em ligações telefônicas e mensagens de texto, contento: data e hora do último evento, se é uma ligação de entrada, saída ou perdida e SMS de entra ou saída, as setas indicam se o evento é de entrada, saída ou perdida, por exemplo, para ligação telefônica, a seta para baixo indica o evento de ligação telefônica de entrada, ou seja, o usuário recebeu uma ligação telefônica.

O Quadro 5.1 mostra exemplos de composições de fenótipos digitais. As linhas 1 até 15 pertencem ao módulo *Physical_Sociability*, as quais descrevem o nome do processador de dados que gerou a informação *Physical_Sociability* (linha 2), o identificador do usuário (linha 3), a lista de atributos (linha 4 a 12) e a situação de interesse inferida (linha 13 a 15). Interpretando essas informações, o framework *OpenDP* detectou a socialização do

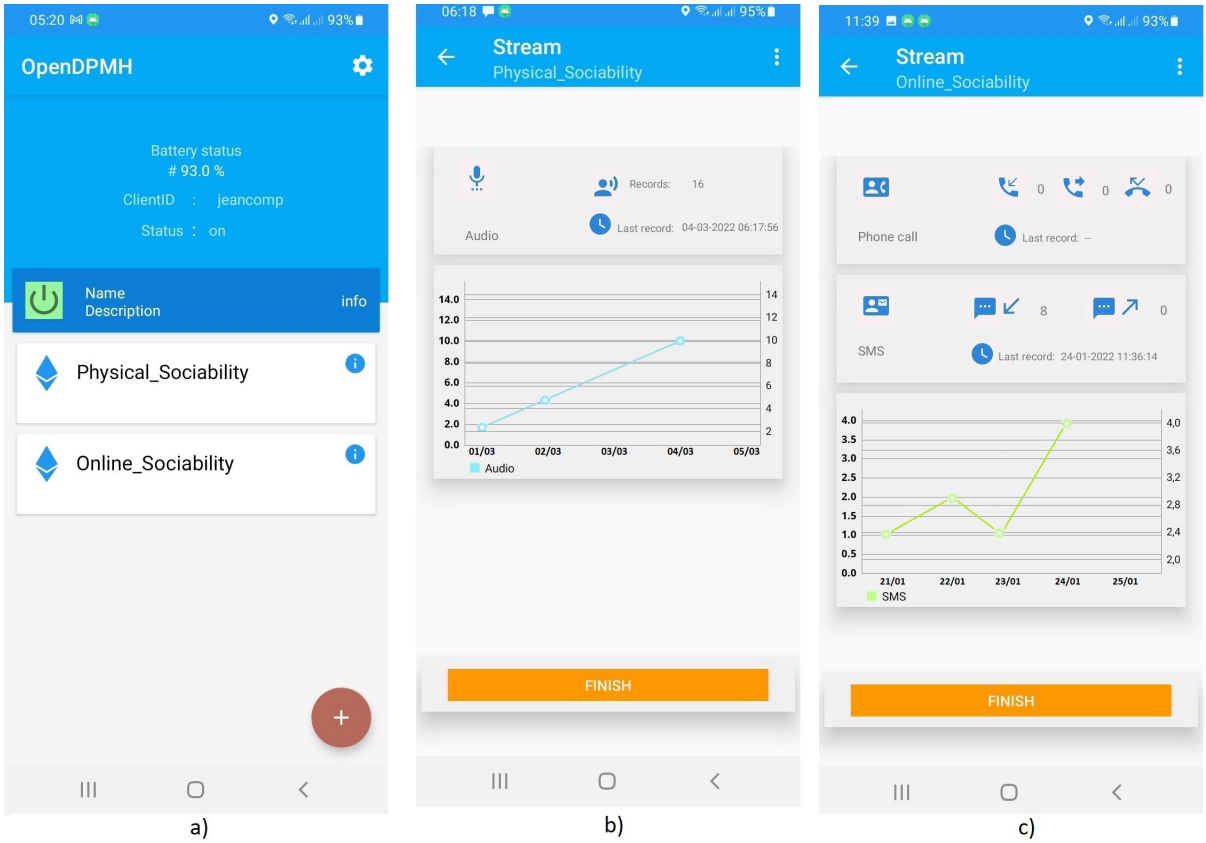

Figura 29 – Segundas telas da aplicação *SMMS*.

usuário usando o sensor microfone. Foi identificado voz humana no áudio capturado cerca de 14:24:06 (horário) do dia 7 de março de 2022 (linha 10). A medida que o framework no smartphone compõe os fenótipos digitais e distribui para os interessados, a psicóloga tem a possibilidade de criar um perfil comportamental da usuária Maria.

Quadro 5.1 – Exemplos de eventos de fenótipos digitais.

```
1  DigitalPhenotype
2  DataProcessorName: Physical_Sociability
3  Uid: jeancomp
4  Attributes
5  Label: message
6  Value: Human_voice_detected
7  Type: String
8  QualityAttribute: false
9  Label: timestamp
10 Value: 1646673846382
11 Type: Date
12 QualityAttribute: false
13 Situation
14 Label: Sociability
15 Description: We identify Physical_Sociability by the user through the audio.
16 ═══════════════════════════════════════════════════════════════
17 DigitalPhenotype
18 DataProcessorName: Online_Sociability
19 Uid: jeancomp
20 Attributes
21 Label: TipoMensagem
```

```
22 Value: Entrada
23 Type: String
24 QualityAttribute: false
25 Label: CorpoMensagem
26 Value: 121fbfe502a826f6388ab14c92ef8590
27 Type: String
28 QualityAttribute: false
29 Label: Timestamp
30 Value: 1647543836385
31 Type: Date
32 QualityAttribute: false
33 Label: NumeroDestinatario
34 Value: f3bcb1d8875a18bc6bd40478604dbcc1
35 Type: String
36 QualityAttribute: false
37 Situation
38 Label: SMS_Online_Sociability
39 Description: We identify Online_Sociability by the user through the SMS.
40 ================================================================
41 DigitalPhenotype
42 DataProcessorName: Online_Sociability
43 Uid: jeancomp
44 Attributes
45 Label: NumeroTelefone
46 Value: f3bcb1d8875a18bc6bd40478604dbcc1
47 Type: String
48 QualityAttribute: false
49 Label: TipoChamada
50 Value: Saida
51 Type: String
52 QualityAttribute: false
53 Label: Timestamp
54 Value: 1646916835807
55 Type: Date
56 QualityAttribute: false
57 Label: Duracao(seg)
58 Value: 10
59 Type: String
60 QualityAttribute: false
61 Situation
62 Label: PhoneCall_Online_Sociability
63 Description: We identify OnlineSociability by the user through the phoneCall.
```

O Quadro 5.1 (linha 17 a 39) exibe outro exemplo de informações de alto nível que compõe o fenótipo digital. Agora nesse exemplo, trata-se de uma informação de alto nível do processador *Online_Sociability*, possui atributos como: tipo da mensagem, corpo da mensagem (mensagem criptografada), horário do evento, número do destinatário (mensagem criptografada) e a situação de interesse inferida. Interpretando o evento do fenótipo digital, temos que o framework detectou que o usuário socializou através de uma mensagem de texto (SMS) por volta das 16:03:56 (horário) do dia 17 de março de 2022, numa terça-feira (linha 30). Lembrando que o número do destinatário e o corpo da mensagem são criptografada por uma função md5 (WIKIPEDIA, 2022).

O Quadro 5.1 (linha 41 a 63) exibe outro exemplo de informações de alto nível que

compõe o fenótipo digital. Diferente do exemplo anterior (linha 17 a 39) citado acima, esse evento foi detectado por uma chamada telefônica. Possui atributos como: o tipo de chamada telefônica, a data e horário que o evento ocorreu, a duração da chamada telefônica e a situação de interesse. Interpretando esse evento, temos que o framework detectou a socialização através de uma chamada telefônica do usuário por volta das 09:53:55 (horário) do dia 10 de março de 2022, numa quinta-feira (linha 54), as informações de número de contato são criptografadas.

Nesse estudo de caso, os resultados alcançados mostram que o framework proposto integrado com o *middleware* M-Hub/CDDL conseguiu coletar dados de sensores virtuais (chamadas telefônica e SMS), recurso esse que não estava implementado no M-Hub/CDDL. Também foi demonstrado que a ferramenta consegue através dos módulos de processamento de dados inferir informações de alto nível, ou seja, delega-se ao desenvolvedor o papel de implementar o motor de inferência de dados brutos (*DataProcessor*). Foi mostrado através do Quadro 5.1 exemplos de composições de fenótipos digitais a partir das inferências dos dados de contexto gerados pelos *DataProcessor*.

Outro ponto positivo alcançado nesse estudo de caso é que a solução proposta facilitou o desenvolvimento de uma aplicação de fenotipagem digital. Se a aplicação fosse desenvolvida sem o uso do framework, o desenvolvedor teria de implementar, dentre outros recursos, todo o processo de coleta de dados de sensores, a lógica do modo de composição de fenótipos digitais, além da lógica dos módulos de processamento de dados.

Portanto, a aplicação *SMMS* de fenotipagem digital desenvolvida para entender o comportamento social e possíveis hábitos da usuária alcançou resultados satisfatórios. A aplicação *SMMS* conseguiu monitorar o dia-a-dia e identificar quando a Maria estava socializando, a partir das informações obtidas da situação de interesse (sociabilidade), ou seja, a composição dos fenótipos digitais, a psicóloga pode aplicar medidas (e.g., tratamento, terapia) que vão garantir o bem-estar da usuária.

5.1.2 Estudo de Caso 2: adicionando um *Plugin*

Para o segundo estudo de caso, foi usado o cenário de aplicação descrito na seção anterior, em que a paciente Maria é monitorada remotamente pela psicóloga Andressa. Ao passar das sessões, a psicóloga percebeu haver a necessidade de adicionar uma nova situação de interesse para monitorar Maria: a atividade física humana (e.g., correndo, andando, dirigindo). Então, após algumas semanas com o uso da aplicação *SMMS* usando o framework *OpenDP* e instalada no *smartphone* de Maria para o monitoramento da sociabilidade, foi implementado e instalado o módulo *Physical_Activity*, visando identificar o estado físico da usuária no decorrer do dia-a-dia.

Com base a necessidade de estender o framework, adicionando uma nova situação de

interesse (i.e., *Physical_Activity*) sem precisar recompilar o projeto que estar executando a aplicação do framework e ter que interromper a coleta de dados, usamos um *plugin*. Foi implementado uma nova aplicação rodando a situação de interesse *Physical_Activity*. A nova aplicação utiliza a Activity Recognition (API) (GOOGLE, 2022) para inferir os estados físicos do usuário, por exemplo, correndo, dirigindo, inclinado, andando e parado. Importantemente, a documentação da biblioteca disponibilizada pela Google não deixa claro quais sensores são usados para inferir as informações de alto nível. A Figura 30 mostra a arquitetura da aplicação *SMMS* com o uso do *plugin* instalado no dispositivo móvel da usuária.

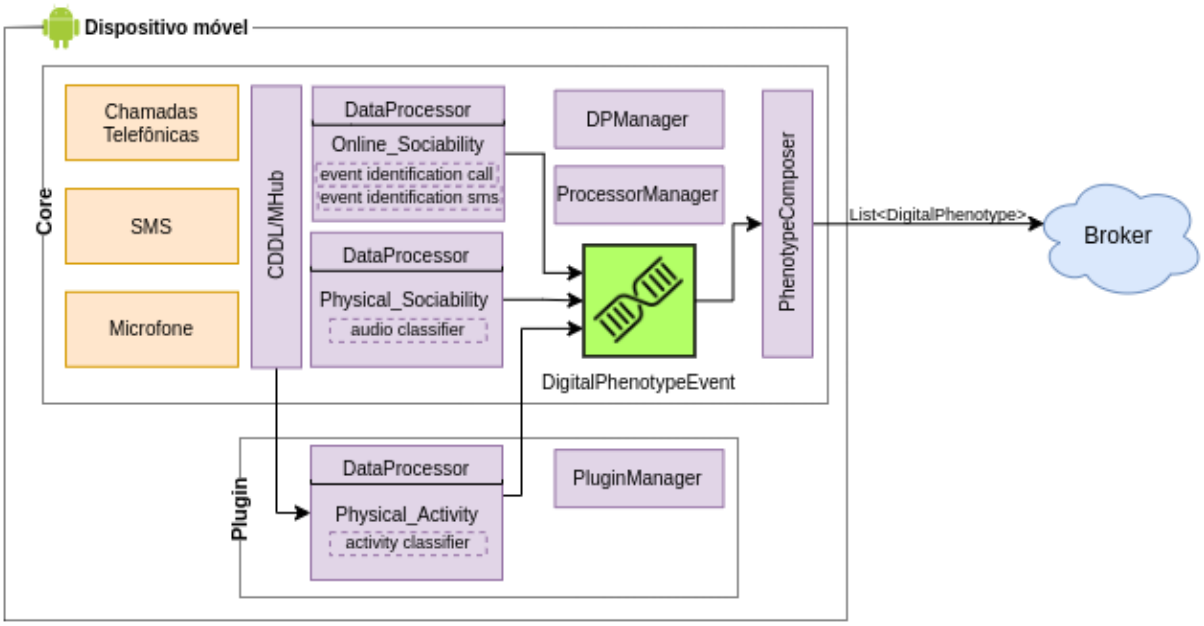

Figura 30 – Estudo de caso 2: arquitetura do *SMMS* com *plugin*

A Figura 31 exibi as telas da aplicação *SMMS* usando o *plugin* para adicionar outro módulo de processamento de dados do estudo de caso 2. A Figura 31a indica que apenas os módulos *Physical_Sociability* e *Online_Sociability* estão sendo executados. Após executar a aplicação *plugin*, o módulo *Physical_Activity* pode ser ativado, como visto na Figura 31b, pois está disponível para uso. A tela da Figura 31c mostra que o módulo foi executado com êxito e, finalmente, a tela da Figura 31d mostra um resumo do que está sendo inferido pelo módulo *Physical_Activity*: o tipo de estado físico detectado (i.e., se o usuário está correndo, dirigindo, etc).

O Quadro 5.2 descreve um exemplo de evento de fenótipo digital do módulo *Physical_Activity*. O exemplo apresenta uma lista de atributos como a *confidence* (linha 5). A propriedade *confidence* é sempre um número inteiro, variando de 0 a 100. Se uma atividade for acompanhada por uma propriedade de confiança de 75% ou mais, geralmente é seguro presumir que o usuário está realizando essa atividade. Essa é uma informação provida pela API (GOOGLE, 2022). Outro atributo é a data e hora que ocorreu o evento

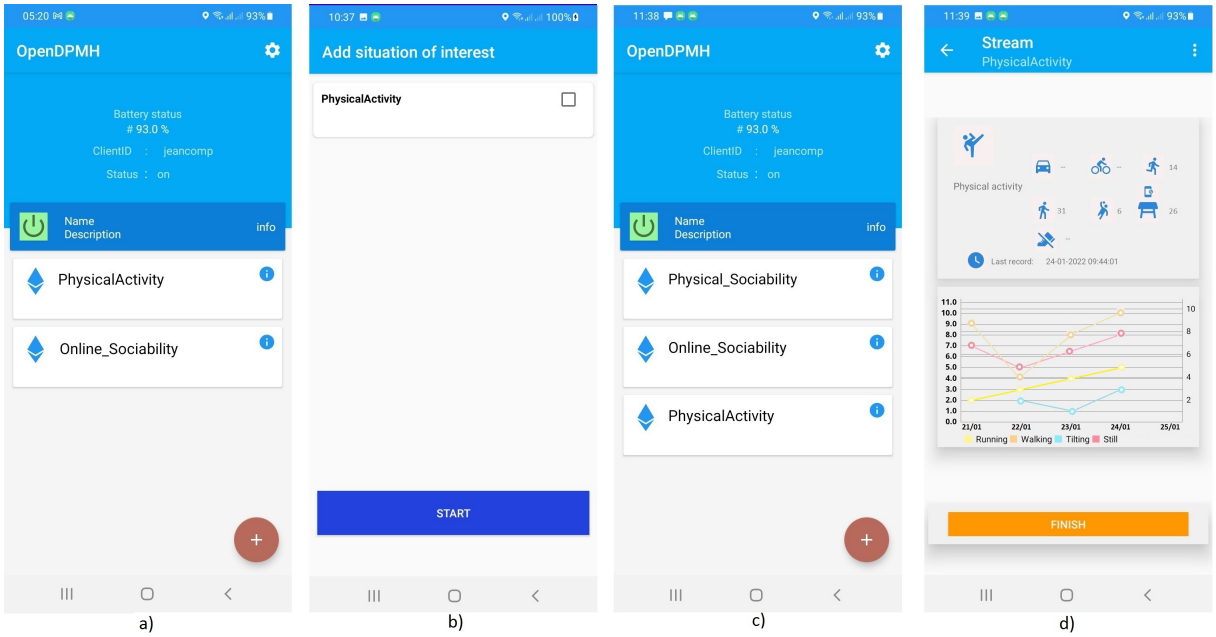

Figura 31 – Telas da aplicação *SMMS* usando o *OpenDP* com o *plugin*.

(linha 9) e, por último, a situação (linha 13) mostra o resultado da inferência do estado físico do usuário. Interpretando essas informações, podemos dizer que o framework *OpenDP* fez o reconhecimento do estado físico do usuário, identificou que o usuário estava andando por volta das 10:41:59 (horário) do dia 9 de março de 2022 numa quarta-feira (linha 10). Portanto, a medida que o framework compõe os fenótipos digitais, o profissional de saúde consegue obter informações sobre o comportamento do usuário.

Quadro 5.2 – Exemplo de um evento de fenótipo digital para atividade física do usuário.

```
1  DigitalPhenotype
2  DataProcessorName: PhysicalActivity
3  Uid: jeancomp
4  Attributes
5  Label: Confidence
6  Value: 1
7  Type: Integer
8  QualityAttribute: false
9  Label: timestamp
10 Value: 1646833319778
11 Type: Date
12 QualityAttribute: false
13 Situation
14 Label: Walking
15 Description: Type of activity
```

O segundo estudo de caso demonstra que a arquitetura do *plugin* funciona e conseguiu trocar informações com o *core*, e o módulo de processamento de dados implementado no *plugin* conseguiu receber os dados dos sensores, processar e devolver para o *core* compor os fenótipos. Além disso, o desenvolvedor pode implementar diferentes módulos no *plugin*, então tendo liberdade para estender o framework. Portanto, o framework *OpenDP* disponibiliza recursos de extensibilidade e poder ser reutilizado em outras pesquisas.

5.2 Avaliação Experimental do Consumo de Energia

Realizamos uma avaliação experimental utilizando a métrica de consumo de energia para analisar o impacto causado no dispositivo móvel com o framework *OpenDP* em execução. Realizamos a primeira avaliação experimental para analisar o impacto do consumo de energia em relação aos três modos de composição do fenótipo digital (*send_when_it_arrives*, *group_all* e *frequency*). A segunda avaliação experimental objetivou analisar o impacto do consumo de energia usando o *plugin* para estender o framework. Já a terceira avaliação visou analisar o consumo ao ativar a segurança do framework.

5.2.1 Configurações dos Experimentos

As aplicações móveis de fenotipagem digital não devem consumir energia em excesso. Caso contrário, ao acelerar o esgotamento da bateria, a aplicação será utilizada por pouco tempo, diminuindo o tempo de monitoramento e coleta de dados do usuário. Além disso, o esgotamento acelerado de bateria irá possivelmente criar insatisfação do usuário, causando a desinstalação da aplicação. Ao considerar o consumo de energia um ponto-chave para a adoção de aplicações de fenotipagem digital, o objetivo dos experimentos foi avaliar o impacto que o framework *OpenDP* trouxe para o consumo de energia de um *smartphone*.

Nos experimentos, utilizamos as aplicações desenvolvidas nos estudos de caso anteriores com diferentes configurações para monitorar o consumo de energia. As aplicações foram executadas por um período de 10 horas. Após a inicialização, as aplicações começavam a coletar os dados dos sensores e compor os fenótipos digitais. O nível da bateria foi medido antes e depois do tempo de execução. Para isso, utilizamos a biblioteca do Android *BatteryManager* (DEVELOPERS, 2022) criada para esta finalidade. Além disso, com a exceção das aplicações que integram o sistema operacional Android (e.g., calendário, app de troca de SMS, app de ligação telefônica), não haviam outras aplicações sendo executados no dispositivo móvel. Os experimentos foram realizados em um *smartphone* Samsung Galaxy A01, de 32 GB de armazenamento em disco, com 2 GB de memória RAM, rede 4G, Wi-Fi, bateria de 3000 mAh, processador Octa-Core 2.0 GHz, e sistema operacional Android.

5.2.2 Primeira Avaliação

O objetivo da primeira avaliação foi analisar o impacto do consumo de energia causado pelo framework funcionando com dois módulos de processamento (*Physical_Activity* e *Physical_Sociability*) de dados nos três modos de composição de fenótipos digitais (*frequency*, *group_all* e *send_when_it_arrives*). O modo *frequency* utilizou duas configurações de frequência para envio de dados ao *broker*: alta (a cada 15 minutos) e baixa (a cada 45 minutos). O módulo *Physical_Activity* foi configurado para, caso não fosse inferido

qualquer atividade em um período de um minuto, ele disparava uma mensagem sem dado para o *PhenotypeComposer*. O objetivo dessa condição foi não deixar o *PhenotypeComposer* sem receber dados por muito tempo. O módulo *Physical_Sociability* foi configurado para processar por um minuto e pausar por 3 minutos. Essas configurações dos módulos permanecesses fixas em todas as avaliações. Dessa forma, criamos cinco cenários (*setups*) para a primeira avaliação, como visto abaixo.

1. ***Setup 1***: *smartphone* com configuração de fábrica;
2. ***Setup 2***: Somente aplicação do *core* executando; Módulos de processamento: *Physical_Activity* e *Physical_Sociability*; Modo de composição: *frequency* alta (15min);
3. ***Setup 3***: Somente aplicação do *core* executando; Módulos de processamento: *Physical_Activity* e *Physical_Sociability*; Modo de composição: *frequency* baixa (45min);
4. ***Setup 4***: Somente aplicação do *core* executando; Módulos de processamento: *Physical_Activity* e *Physical_Sociability*; Modo de composição: *group_all*;
5. ***Setup 5***: Somente aplicação do *core* executando; Módulos de processamento: *Physical_Activity* e *Physical_Sociability*; Modo de composição: *send_when_it_arrives*.

Os resultados alcançados mostram que a variação do consumo de energia foi pequena entre os modos de composição de fenótipos digitais por um período de 10 horas, conforme pode ser observado na Figura 32. Nos *setups* 2 e 3, houve o consumo de 8% da bateria (3000 mAh) para o intervalo que os fenótipos são distribuídos a cada 15 minutos e 7% para 45 minutos. Já no modo de composição *group_all*, houve 10% de consumo de energia, e 13% no modo *send_when_it_arrives*. Comparando os três modos de composição de fenótipo entre si, observamos que houve um consumo maior para o modo *send_when_it_arrives*, pois ocorre um aumento na frequência de distribuição de dados para o *broker* externo. Para o modo de composição de fenótipos digitais *send_when_it_arrives*, toda informação que chega no *PhenotypeComposer* é distribuída para o *broker* imediatamente. Por isso, entre os três modos de composição, ele é o que mais consume energia. No modo de composição de fenótipo *group_all*, a frequência é baseada na frequência de geração de dados dos processadores de dados: a regra para distribuição dos dados no *PhenotypeComposer* é satisfeita quando chegarem todos os dados dos módulos ativos. Já para o modo *frequency*, quanto mais alta a frequência de distribuição dos dados para o *broker*, maior será o consumo de energia.

5.2.3 Segunda Avaliação

O objetivo da segunda avaliação foi analisar o impacto do consumo de energia causado com a adição do *plugin*. As aplicações de cada cenário também foram executadas por um período de 10 horas. Definimos o intervalo de distribuição dos fenótipos

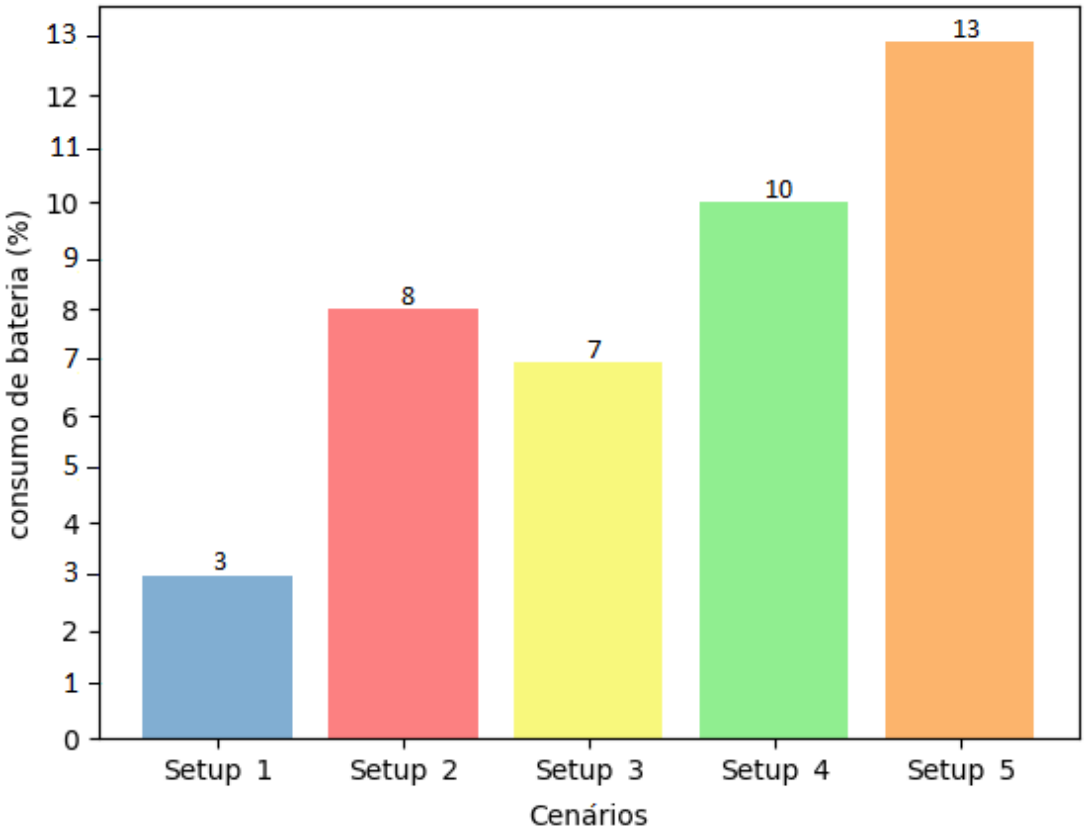

Figura 32 – Resultados da primeira avaliação.

no modo *frequency* para alta (a cada 15 minutos). Utilizamos apenas um módulo de processamento de dados (*Physical_Activity*) no *plugin*, com nenhum módulo no *core*. Dessa forma, temos quatro *setups* na segunda avaliação, como visto abaixo. Ressaltamos que realizamos experimentos com o modo *group_all*, e os resultados foram parecidos com o modo *send_when_it_arrives*. Como utilizamos apenas um módulo de processamento, o modo de composição *group_all* funciona do mesmo modo do *send_when_it_arrives*.

1. ***Setup 1***: *Core* executando *Physical_Activity*, sem o *plugin*; Modo de composição: *frequency* alta (15min);
2. ***Setup 2***: *Core* sem módulo de processamento e o *plugin* executando *Physical_Activity*; Modo de composição: *frequency* alta (15min);
3. ***Setup 3***: *Core* executando *Physical_Activity*, sem o *plugin*; Modo de composição: *send_when_it_arrives*;
4. ***Setup 4***: *Core* sem módulo de processamento e o *plugin* executando *Physical_Activity*; Modo de composição: *send_when_it_arrives*.

Os resultados da segunda avaliação mostram que houve uma diferença no consumo de energia com a adição do *plugin*. Na Figura 33 podemos observar os *setups* 1 e 2 com o modo *frequency*, em que o consumo de energia foi maior usando o *plugin* do que o *core* com

9% e 5%, respectivamente. Acreditamos que o consumo de energia com o uso do *plugin* foi maior devido haverem duas aplicações em execução no *smartphone* com serviços rodando em primeiro plano. O mesmo acontece nos *setups* 3 e 4 com 8% de consumo de energia usando somente o *core* e 10% com a inclusão do *plugin*. Para esses dois cenários, o modo de composição foi *send_when_it_arrives*.

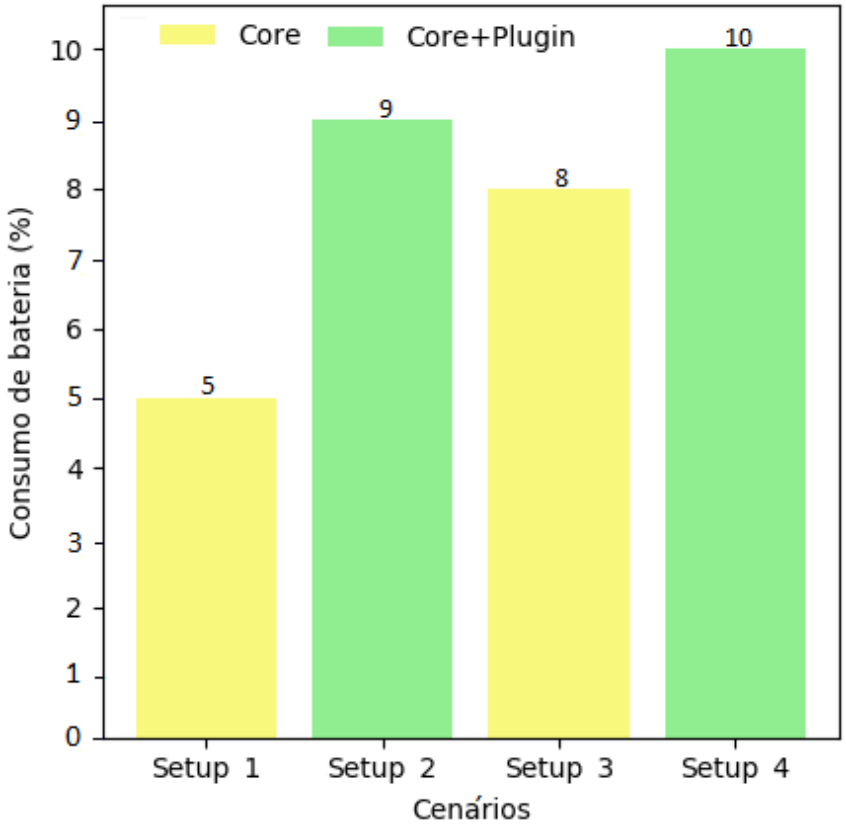

Figura 33 – Resultados da segunda avaliação.

5.2.4 Terceira Avaliação

O objetivo da terceira avaliação foi mostrar o impacto no consumo de energia ao ativar o recurso de segurança provido no framework. Para isso, usamos o *core* utilizado dois módulos de processamento de dados (*Physical_Activity* e *Physical_Sociability*), com as mesmas configurações. Selecionamos o modo de composição de fenótipos digitais *frequency* alta (distribuindo dados a cada 15 minutos). Esse modo foi escolhido devido garantir que a mesma quantidade de dados seria distribuída em ambos *setups*, os quais são apresentados abaixo.

1. ***Setup 1***: Somente aplicação do *core* executando; Módulos de processamento: *Physical_Activity* e *Physical_Sociability*; Modo de composição: *frequency* alta (15min); **Sem segurança**;

2. ***Setup 1***: Somente aplicação do *core* executando; Módulos de processamento: *Physical_Activity* e *Physical_Sociability*; Modo de composição: *frequency* alta (15min); **Com segurança**.

Os resultados mostram não haver uma grande diferença ao usar o recurso de segurança fornecido pelo framework. O consumo foi de 8% no *setup* 1 (sem segurança), e 10% no segundo experimento com o uso de segurança: uma diferença de 2% no impacto do consumo de energia. Acreditamos que essa diferença no consumo de 2% foi devido ao processo de criptografia das mensagens, pois toda mensagem enviada do *core* para o *broker* passa é criptografada. Para criptografar, há um processamento adicional e o tamanho da mensagem transmitida é aumentado.

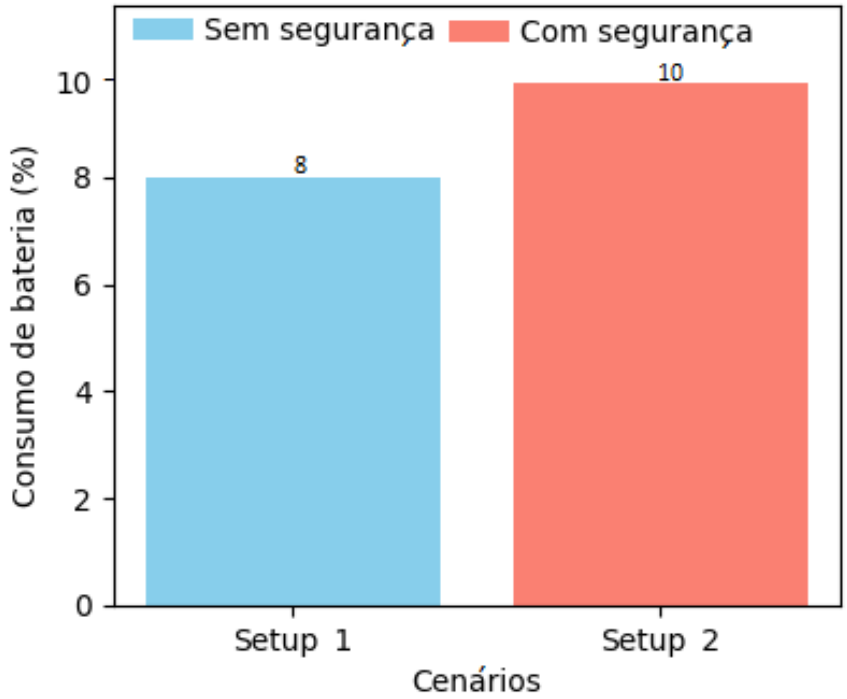

Figura 34 – Resultados da terceira avaliação.

Esses resultados mostram o custo relacionado ao consumo de bateria ao se ativar o recurso de segurança, o qual não foi elevado. Dessa forma, acreditamos que o uso da segurança no framework não é um fato que pode comprometer a utilização de uma aplicação de fenotipagem digital desenvolvida com o *OpenDP*. Logo, o desenvolvedor pode usar esse recurso para garantir proteção na distribuição dos dados.

6 Considerações Finais

Neste trabalho, foi apresentado um framework para facilitar o desenvolvimento de aplicações móveis de fenotipagem digital. Foram realizados dois estudos de caso demonstrando que os objetivos foram alcançados, como: a solução proposta consegue coletar dados de sensores virtuais, também foi demonstrado que o desenvolvedor consegue implementar e adicionar módulos de processamento de dados brutos (plugins), permitindo que o framework seja extensível e reutilizável. Os estudos de caso realizados também constatou que através dos módulos de processamento de dados foi possível compor fenótipos digitais a partir de informações de alto nível.

A partir da avaliação experimental, foi demonstrado em três cenários de monitoramento do usuário, por meio da implementação de uma aplicação de fenotipagem digital a partir do framework, com a finalidade de coletar, processar e compor fenótipos digitais, por um período de 10 horas, não comprometendo o consumo de energia do *smartphone*. Dessa forma, o desenvolvedor pode explorar os recursos disponíveis do framework para criar aplicações de fenotipagem digital.

Portanto, foi projetado e desenvolvido o *OpenDP*, um framework aberto de computação móvel sensível ao contexto para fenotipagem digital. O framework utilizando os recursos oferecidos pelo *middleware* M-Hub/CDDL, tem o potencial de permitir o acesso a vários sensores, e através dos módulos de processamento de dados gerar informações de alto nível, permitindo que as aplicações consigam caracterizar o comportamento, humor e hábitos dos usuários.

6.1 Limitações

A solução proposta neste trabalho de mestrado apresenta limitações, a começar pela versão inicial focada apenas para o sistema operacional Android. Caso haja interesse em construir aplicações de fenotipagem digital para o sistema operacional iOS, a implementação do framework não pode ser utilizada, embora sua modelagem possa ser replicada. A ferramenta também não foi avaliada com seres humanos. Portanto, nenhuma aplicação desenvolvida a partir do framework foi testada com profissionais e pacientes. Também não foi feita uma avaliação com desenvolvedores de aplicações móveis, o qual é o principal usuário, por ter contato direto com a ferramenta para a criação de aplicações. Concluindo que mais estudos experimentais devem ser realizados com a ferramenta para avaliar e validar ela.

6.2 Trabalhos Futuros

A partir dessa pesquisa, outros trabalhos podem ser desenvolvidos para dar continuidade, tais como:

- Avaliação junto aos desenvolvedores de aplicações de fenotipagem digital com uso do framework, aplicando questionários ao final do desenvolvimento das aplicações;
- Adicionar mecanismo de coleta de dados EMA/EMI (e.g., questionários) ao framework por meio da programação de formulários em eXtensible Markup Language (XML), com agendamento desses formulários;
- Adicionar o monitoramento e coleta de dados das redes sociais através de recursos que as API fornecem, identificando postagem dos usuários nas redes sociais, pois as redes sociais apresentam grande oportunidades de informações para análise de sentimento dos usuários;
- Gerar uma versão do framework para o sistema operacional iOS;
- Criar uma ferramenta de autoria para facilitar o desenvolvimento de aplicações com o uso do framework, ou seja, para profissionais de saúde que não detém conhecimento de programação, ele vai definir algumas especificações/configurações, baseado nessas especificações, é gerado um *apk* a partir do framework;
- Ferramenta de visualização de dados produzidos.

6.3 Publicações

Até o estágio atual desta pesquisa, realizamos a publicação do artigo (1) abaixo, e contribuímos para a publicação do capítulo de livro (2). Além destes trabalhos, estamos em processo de submissão do artigo (3) e contribuímos para o trabalho submetido (4).

1. **MENDES, J. P. M.** et al. Sensing apps and public data sets for digital phenotyping of mental health: Systematic review. J Med Internet Res (JMIR), v. 24, n. 2, p. e28735, Feb 2022. ISSN 1438-8871. Disponível em: <https://www.jmir.org/2022/2/e28735>;
2. Rodrigues, Ivan and Silva, Francisco and Coutinho, Luciano and **Mendes, Jean** and Teles, Ariel. Detectando padrões de sociabilidade de seres humanos através do processamento de eventos complexos. In: Minicursos da ERCEMAPI 2020. SBC, 2020. p. 72–96. Disponível em: <https://doi.org/10.5753/sbc.11.5.4>;
3. **MENDES, J. P. M.** et al. OpenDP: A Framework to Facilitate Development of Digital Phenotyping Mobile Applications. *Mobile Networks and Applications, Springer Science and Business Media LLC*;

4. Rodrigues, Ivan and Teles, Ariel and Viana, Davi and **Mendes, Jean** and Coutinho, Luciano and Silva, Francisco. Digital Phenotyping of Mental Health Using Multimodal Sensing of Multiple Situations of Interest: A Systematic Literature Review. ACM Comput. Surv., Association for Computing Machinery.

Referências

AHARONY, N. *et al.* Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, v. 7, n. 6, p. 643–659, 2011. ISSN 1574-1192. The Ninth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2011). Disponível em: <https://www.sciencedirect.com/science/article/pii/S1574119211001246>. Citado 3 vezes nas páginas x, 38 e 39.

AHARONY, N. *et al.* Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, v. 7, n. 6, p. 643–659, 2011. ISSN 1574-1192. The Ninth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2011). Disponível em: <https://www.sciencedirect.com/science/article/pii/S1574119211001246>. Citado na página 52.

ANDROID, G. D. *Content Observer*. 2022. Disponível em: <https://developer.android.com/reference/android/database/ContentObserver>. Citado na página 37.

ANDROID, G. D. *Fundamentos do provedor de conteúdo*. 2022. Disponível em: <https://developer.android.com/guide/topics/providers/content-provider-basics?hl=pt-br>. Citado na página 37.

ANDROID, G. D. *Visão geral de transmissões do Sistema Operacional Android*. 2022. Disponível em: <https://developer.android.com/guide/components/broadcasts?hl=pt-br>. Citado na página 37.

ASSOCIATION, G. *The Mobile Economy 2020*. 2021. Online; Accessed: Oct 14, 2021. Disponível em: <https://www.gsma.com/mobileeconomy/wp-content/uploads/2020/03/GSMA_MobileEconomy2020_Global.pdf>. Citado na página 15.

BERROUIGUET, S. *et al.* Combining continuous smartphone native sensors data capture and unsupervised data mining techniques for behavioral changes detection: A case series of the evidence-based behavior (eb2) study. *JMIR Mhealth Uhealth*, v. 6, n. 12, p. e197, Dec 2018. ISSN 2291-5222. Disponível em: <https://mhealth.jmir.org/2018/12/e197/>. Citado 2 vezes nas páginas 24 e 25.

BEUTEL, M. E. *et al.* Loneliness in the general population: prevalence, determinants and relations to mental health. *BMC psychiatry*, BioMed Central, v. 17, n. 1, p. 97–97, Mar 2017. ISSN 1471-244X. 28320380[pmid]. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/28320380>. Citado na página 69.

BEZERRA BENILTON, J. Tecnologias digitais, subjetividade e psicopatologia: possíveis impactos da pandemia. *Revista Latinoamericana de Psicopatologia Fundamental*, v. 23, n. 3, p. 495–508, 2020. ISSN 1415-4714. Disponível em: <http://dx.doi.org/10.1590/1415-4714.2020v23n3p495.4>. Citado na página 20.

CARDOSO, A. *et al.* Facilitando a distribuição segura de dados de contexto em aplicações de internet das coisas móveis. In: *Anais da VIII Escola Regional de Computação do Ceará,*

Maranhão e Piauí. Porto Alegre, RS, Brasil: SBC, 2020. p. 252–259. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/ercemapi/article/view/11492>. Citado na página 30.

CARDOSO, A. *et al.* Uma arquitetura para a distribuição segura de dados de contexto em aplicações de internet das coisas móveis. *Revista de Sistemas e Computação-RSC*, v. 10, n. 3, 2020. Citado na página 30.

CLARK, A.; CHALMERS, D. The extended mind. *Analysis*, v. 58, n. 1, p. 7–19, 1998. ISSN 0003-2638. Disponível em: <http://dx.doi.org/10.1093/analys/58.1.7>. Citado na página 19.

COGHLAN, S.; D'ALFONSO, S. Digital phenotyping: an epistemic and methodological analysis. *Philosophy & Technology*, Springer Science and Business Media LLC, v. 34, n. 4, p. 1905–1928, nov. 2021. Disponível em: <https://doi.org/10.1007/s13347-021-00492-1>. Citado 3 vezes nas páginas x, 20 e 21.

CONTRIBUTORS, W. *MQTT*. 2022. Online; Accessed: Mar 14, 2022. Disponível em: <https://pt.wikipedia.org/w/index.php?title=MQTT&oldid=61984339>. Citado na página 26.

CONTROL, C. C. for D.; PREVENTION. *How to protect yourself and others*. 2021. Online; Accessed: Oct 14, 2021. Disponível em: <https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/social-distancing.html>. Citado na página 15.

DATTANI, S.; RITCHIE, H.; ROSER, M. *Our World in Data*. 2021. Online; Accessed: Oct 02, 2021. Disponível em: <https://ourworldindata.org/mental-health>. Citado na página 15.

DEVELOPERS, G. *Monitor the battery level and charging state*. 2022. Online; Accessed: Mar 14, 2022. Disponível em: <https://developer.android.com/training/monitoring-device-state/battery-monitoring>. Citado na página 78.

FAUTH, E. B. *et al.* Changes in depressive symptoms in the context of disablement processes: role of demographic characteristics, cognitive function, health, and social support. *The journals of gerontology. Series B, Psychological sciences and social sciences*, Oxford University Press, v. 67, n. 2, p. 167–177, Mar 2012. ISSN 1758-5368. 21821838[pmid]. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/21821838>. Citado na página 69.

FERREIRA, D.; KOSTAKOS, V.; DEY, A. K. Aware: Mobile context instrumentation framework. *Frontiers in ICT*, v. 2, 2015. ISSN 2297-198X. Disponível em: <https://www.frontiersin.org/article/10.3389/fict.2015.00006>. Citado 4 vezes nas páginas x, 36, 37 e 38.

GARCIA-CEJA, E. *et al.* Mental health monitoring with multimodal sensing and machine learning: A survey. *Pervasive and Mobile Computing*, v. 51, p. 1–26, 2018. ISSN 1574-1192. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1574119217305692>. Citado na página 15.

GNU, F. S. F. *Licença Pública Geral Menor GNU v3.0 - Projeto GNU - Free Software Foundation*. 2022. Online; Accessed: Mar 14, 2022. Disponível em: <https://www.gnu.org/licenses/lgpl-3.0.pt-br.html>. Citado na página 49.

GOMES, B. D. T. P. *et al.* A middleware with comprehensive quality of context support for the internet of things applications. *Sensors*, v. 17, n. 12, 2017. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/17/12/2853>. Citado 8 vezes nas páginas x, 19, 25, 26, 28, 52, 55 e 60.

GOMES, B. de T. P. *et al.* A comprehensive and scalable middleware for ambient assisted living based on cloud computing and internet of things. *Concurrency and Computation: Practice and Experience*, Wiley, v. 29, n. 11, p. e4043, dez. 2016. Disponível em: <https://doi.org/10.1002/cpe.4043>. Citado 2 vezes nas páginas 19 e 31.

GOOGLE. *Activity recognition API*. 2022. Online; Accessed: Mar 14, 2022. Disponível em: <https://developers.google.com/location-context/activity-recognition>. Citado na página 76.

GRAV, S. *et al.* Association between social support and depression in the general population: the hunt study, a cross-sectional survey. *Journal of Clinical Nursing*, v. 21, n. 1-2, p. 111–120, 2012. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2702.2011.03868.x>. Citado na página 69.

GUPTA, L. *Builder Design Pattern*. 2014. Disponível em: <https://howtodoinjava.com/design-patterns/creational/builder-pattern-in-java/>. Citado na página 49.

HENSON, P. *et al.* Anomaly detection to predict relapse risk in schizophrenia. *Translational Psychiatry*, Springer Science and Business Media LLC, v. 11, n. 1, jan 2021. Disponível em: <https://doi.org/10.1038/s41398-020-01123-7>. Citado 2 vezes nas páginas 15 e 41.

HUCKVALE, K.; VENKATESH, S.; CHRISTENSEN, H. Toward clinical digital phenotyping: a timely opportunity to consider purpose, quality, and safety. *npj Digital Medicine*, Springer Science and Business Media LLC, v. 2, n. 1, set. 2019. Disponível em: <https://doi.org/10.1038/s41746-019-0166-1>. Citado na página 16.

INSEL, T. R. Digital phenotyping: Technology for a new science of behavior. *JAMA*, v. 318, n. 13, p. 1215–1216, 10 2017. ISSN 0098-7484. Disponível em: <https://doi.org/10.1001/jama.2017.11295>. Citado na página 15.

JAIN, S. *et al.* The digital phenotype. *Nature biotechnology*, v. 33, p. 462–463, 05 2015. Citado 2 vezes nas páginas 19 e 22.

JEANCOMP. *Fenotipagem_digital_saude_vs_0_1: Framework to Facilitate the Development of Digital Phenotyping Applications*. 2022. Online; Accessed: Mar 14, 2022. Disponível em: <https://github.com/jeancomp/fenotipagem_digital_saude_vs_0_1>. Citado na página 49.

JUNIOR, M. R. *et al.* Introdução ao processamento de fluxo de dados: uma abordagem orientada a eventos complexos. In: _____. *Minicursos do XXV Simpósio Brasileiro de Sistemas Multimídia e Web*. SBC, 2019. p. 45–76. ISBN 9788576694816. Disponível em: <https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/32/125/305-1?inline=1>. Citado na página 52.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. [S.l.], 2007. Citado na página 34.

KONOVALOV, G. *android-vad: This VAD library can process audio in real-time utilizing GMM which helps identify presence of human speech in an audio sample that contains a mixture of speech and noise*. 2022. Online; Accessed: Mar 14, 2022. Disponível em: <https://github.com/gkonovalov/android-vad>. Citado na página 71.

LIANG, Y.; ZHENG, X.; ZENG, D. D. A survey on big data-driven digital phenotyping of mental health. *Information Fusion*, v. 52, p. 290–307, 2019. ISSN 1566-2535. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1566253518305244>. Citado 2 vezes nas páginas 15 e 35.

MARTINS, A.; SAUKAS, E.; ZANARDO, J. Scai: Sistema de controle de acesso para os requisitos da saúde. In: *Anais do IX Congresso Brasileiro de Informática em Saúde*. [S.l.: s.n.], 2004. Citado na página 30.

MENDES, J. P. M. *et al.* Sensing apps and public data sets for digital phenotyping of mental health: Systematic review. *J Med Internet Res*, v. 24, n. 2, p. e28735, Feb 2022. ISSN 1438-8871. Disponível em: <https://www.jmir.org/2022/2/e28735>. Citado 5 vezes nas páginas x, 20, 23, 24 e 34.

MONITORAMENTO ubíquo da saúde mental: detectando padrões de sociabilidade enriquecidos por contexto através do processamento de eventos complexos. Dissertação (Mestrado), 2020. DEPARTAMENTO DE INFORMÁTICA/CCET. Disponível em: <https://tedebc.ufma.br/jspui/handle/tede/tede/3125>. Citado 2 vezes nas páginas 34 e 69.

MOURA, I. *et al.* Mental health ubiquitous monitoring supported by social situation awareness: A systematic review. *Journal of Biomedical Informatics*, v. 107, p. 103454, 2020. ISSN 1532-0464. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1532046420300824>. Citado na página 20.

MOURA, I. R. de *et al.* Recognizing context-aware human sociability patterns using pervasive monitoring for supporting mental health professionals. *Sensors*, v. 21, n. 1, 2021. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/21/1/86>. Citado na página 69.

ORGANIZATION, W. H. *Prevention of mental disorders: effective interventions and policy options: summary report / a report of the World Health Organization Dept. of Mental Health and Substance Abuse; in collaboration with the Prevention Research Centre of the Universities of Nijmegen and Maastricht*. World Health Organization, 2004. ISBN 9786161133535. Disponível em: <https://apps.who.int/iris/handle/10665/43027>. Citado na página 16.

PALAGHIAS, N. *et al.* A survey on mobile social signal processing. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 48, n. 4, mar. 2016. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/2893487>. Citado 3 vezes nas páginas 24, 36 e 52.

PASZTOR, E. Dawkins, r. 1989. (original edition 1982; reprinted 1983, 1987, 1988, 1989): The extended phenotype. oxford paperbacks, oxford university press, oxford, uk. 307 pp. f6.95. *Journal of evolutionary biology*, v. 4, n. 1, p. 161–161, 1991. ISSN 1010-061X. Disponível em: <http://dx.doi.org/10.1046/j.1420-9101.1991.4010161.x>. Citado na página 19.

PSYCHIATRY, D. of D. *MindLamp*. 2020. Online; Accessed: Mar 14, 2022. Disponível em: <https://www.digitalpsych.org/lamp.html>. Citado 2 vezes nas páginas x e 45.

PSYCHIATRY, L. C. . D. of D. *LAMP platform*. 2021. Online; Accessed: Mar 14, 2022. Disponível em: <https://docs.lamp.digital/>. Citado 2 vezes nas páginas x e 45.

SCHUELLER, S. M. *et al.* Purple: A modular system for developing and deploying behavioral intervention technologies. *J Med Internet Res*, v. 16, n. 7, p. e181, Jul 2014. ISSN 1438-8871. Disponível em: <http://www.jmir.org/2014/7/e181/>. Citado na página 40.

Sear, R. F. *et al.* Quantifying covid-19 content in the online health opinion war using machine learning. *IEEE Access*, v. 8, p. 91886–91893, 2020. Citado na página 25.

SILVA, M. *et al.* Neighborhood-aware mobile hub: An edge gateway with leader election mechanism for internet of mobile things. *Mobile Networks and Applications*, Sep 2020. ISSN 1572-8153. Disponível em: <https://doi.org/10.1007/s11036-020-01630-3>. Citado 2 vezes nas páginas 19 e 31.

STRACZKIEWICZ, M.; ONNELA, J. A systematic review of human activity recognition using smartphones. *CoRR*, abs/1910.03970, 2019. Disponível em: <http://arxiv.org/abs/1910.03970>. Citado na página 15.

TELES, A. *et al.* Internet of things applied to mental health: Concepts, applications, and perspectives. In: *IoT and ICT for Healthcare Applications*. Springer International Publishing, 2020. p. 33–58. Disponível em: <https://doi.org/10.1007/978-3-030-42934-8_4>. Citado na página 70.

TELES, A. S. *et al.* Enriching mental health mobile assessment and intervention with situation awareness. *Sensors*, v. 17, n. 1, 2017. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/17/1/127>. Citado 4 vezes nas páginas x, 24, 43 e 44.

TORALES, J. *et al.* The outbreak of covid-19 coronavirus and its impact on global mental health. *International Journal of Social Psychiatry*, 2020. Citado 2 vezes nas páginas 25 e 70.

TOROUS, J. *et al.* New tools for new research in psychiatry: A scalable and customizable platform to empower data driven smartphone research. *JMIR Mental Health*, v. 3, n. 2, 2016. Citado 7 vezes nas páginas x, 15, 20, 25, 35, 41 e 52.

TRULL, T. J.; EBNER-PRIEMER, U. Ambulatory assessment. *Annual Review of Clinical Psychology*, Annual Reviews, v. 9, n. 1, p. 151–176, mar. 2013. Disponível em: <https://doi.org/10.1146/annurev-clinpsy-050212-185510>. Citado na página 15.

UMBERSON, D.; MONTEZ, J. K. Social relationships and health: a flashpoint for health policy. *Journal of health and social behavior*, v. 51 Suppl, n. Suppl, p. S54–S66, 2010. ISSN 0022-1465. 20943583[pmid]. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/20943583>. Citado na página 69.

WANG, R. *et al.* Studentlife: Assessing mental health, academic performance and behavioral trends of college students using smartphones. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. New York, NY, USA: ACM, 2014. (UbiComp '14), p. 3–14. Citado na página 20.

WIKIPEDIA. *MD5*. 2022. Online; Accessed: Mar 14, 2022. Disponível em: <https://pt.wikipedia.org/w/index.php?title=MD5&oldid=59076806>. Citado na página 74.

WISNIEWSKI, H.; HENSON, P.; TOROUS, J. Using a smartphone app to identify clinically relevant behavior trends via symptom report, cognition scores, and exercise levels: A case series. *Frontiers in Psychiatry*, v. 10, 2019. ISSN 1664-0640. Disponível em: <https://www.frontiersin.org/article/10.3389/fpsyt.2019.00652>. Citado 5 vezes nas páginas x, 22, 25, 45 e 46.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: . New York, NY, USA: Association for Computing Machinery, 2014. (EASE '14), p. 1–10. Citado na página 35.

XIONG, H. *et al.* Sensus: A cross-platform, general-purpose system for mobile crowdsensing in human-subject studies. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. New York, NY, USA: Association for Computing Machinery, 2016. (UbiComp '16), p. 415–426. ISBN 9781450344616. Disponível em: <https://doi.org/10.1145/2971648.2971711>. Citado 3 vezes nas páginas x, 42 e 43.