UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

COLEGIADO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Mikaella Ferreira da Silva

Um sistema de rastreamento de contato com foco em anonimidade

Vitória, ES

2023

Mikaella Ferreira da Silva

Um sistema de rastreamento de contato com foco em anonimidade

Monografia apresentada ao Curso de Ciência da Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Colegiado do Curso de Ciência da Computação

Orientador: Prof. Dr. Vinícius Fernandes Soares Mota

Vitória, ES

2023

Mikaella Ferreira da Silva
Um sistema de rastreamento de contato com foco em anonimidade/ Mikaella Ferreira da Silva. – Vitória, ES, 2023-
62 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vinícius Fernandes Soares Mota

Monografia (PG) – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Colegiado do Curso de Ciência da Computação, 2023.

1. Anonimidade. 2. Rastreamento de contatos. I. Silva, Mikaella Ferreira. II. Universidade Federal do Espírito Santo. IV. Um sistema de rastreamento de contato com foco em anonimidade

CDU 02:141:005.7

Mikaella Ferreira da Silva

Um sistema de rastreamento de contato com foco em anonimidade

Monografia apresentada ao Curso de Ciência da Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Vitória, ES, (dia) de (mês) de (ano):

Prof. Dr. Vinícius Fernandes Soares Mota
Orientador

Profª. Ph.D. Patrícia Dockhorn Costa
Universidade Federal do Espírito Santo

Profª. Drª. Roberta Lima Gomes
Universidade Federal do Espírito Santo

Vitória, ES
2023

Agradecimentos

Em primeiro lugar, agradeço aos meus pais, que sempre me incentivaram a estudar e me deram suporte emocional e financeiro para concluir essa graduação.

Aos meus irmãos, por me influenciarem na escolha desse curso.

Ao meu namorado Gabriel, por todo apoio dado em diversos aspectos, principalmente por me escutar e tentar discutir uma solução comigo sobre problemas de matérias/-trabalhos em que não fazia parte.

Ao meu orientador Vinícius, pela oportunidade e assistência dada durante a realização deste trabalho.

Aos professores da banca, por aceitarem avaliar este trabalho.

Por fim, a todos os professores com quem tive aula durante a graduação, por todo conhecimento passado e dedicação.

Resumo

A pandemia do novo coronavírus mostrou o potencial de espalhamento de um vírus entre as pessoas, destacando a importância da utilização de mecanismos para contê-lo. Além do isolamento social e uso de máscaras, o uso de aplicativos de rastreamento de contatos se mostrou bastante promissor. No entanto, revela problemas de privacidade que desmotivam usuários a utilizá-lo, reduzindo a efetividade do sistema. Este trabalho propõe um sistema de rastreamento de contatos totalmente anônimo, seja para os usuários, seja para o servidor.

O sistema foi desenvolvido em duas partes: um servidor *web* e uma aplicação *mobile*, utilizando também um servidor MQTT. Os dispositivos móveis dos usuários atuaram como *beacons*, através do uso da tecnologia *Bluetooth Low Energy*, enviando um identificador único gerado pelo servidor a cada encontro. Essas informações de encontro eram posteriormente enviadas ao servidor por meio do protocolo MQTT. Ao reportar um diagnóstico positivo, o servidor busca essas informações no banco de dados e identifica os contatos de risco, para posteriormente notificar os usuários por meio do servidor MQTT. O registro do usuário não requer nenhum dado pessoal do usuário, utilizando-se apenas do SSAID, um identificador único que combina a chave de assinatura do aplicativo, usuário e dispositivo, também conhecido como Android ID, para associar ao identificador gerado.

O servidor *web* e o servidor MQTT, além do banco de dados relacional Postgre, foram implantados na nuvem *Amazon Web Services* e o aplicativo móvel disponibilizado no Google Drive para *download*. Algumas pessoas foram convidadas a utilizarem o aplicativo para testar o sistema. Os voluntários utilizaram a aplicação por 2 semanas e a quantidade de dados coletada foi satisfatória. Duas pessoas reportaram um diagnóstico positivo e os usuários que tiveram contato com eles por mais de 15 minutos a uma distância menor ou igual a 2 metros foram notificados, conforme o que a Organização Mundial da Saúde estabelece como contato de risco. Também foi possível analisar contatos com potencial de espalhamento do vírus e observar o comportamento social dos usuários, sem que eles pudessem ser identificados com alguma informação pessoal, por meio de grafos de contatos gerados a partir dos dados coletados e gráficos de função de distribuição acumulada da distância média, da duração e do grau dos nós do grafo de contato. Foi possível notar que a maioria dos contatos foram de curta duração e com uma distância média de até 2 metros, além de se observar que grande parte dos usuários teve contato com até 8 usuários ao longo das duas semanas.

Palavras-chaves: Anonimidade. Rastreamento de contatos. Privacidade. Covid. Contágio. Bluetooth.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ID	Identificador
MQTT	Message Queuing Telemetry Transport
BLE	Bluetooth Low Energy
OMS	Organização Mundial da Saúde
JSON	JavaScript Object Notation
SSAID	Service Set Android Identifier
RPC	Remote Procedure Call
API	Application Programming Interface
UUID	Universally Unique Identifier
RSSI	Received Signal Strength Indicator
AWS	Amazon Web Services
RDS	Relational Database Service
EC2	Elastic Compute Cloud
ECDSA	Elliptic Curve Digital Signature Algorithm
QoS	Quality of Service
APK	Android Package Kit
FDA	Função de Distribuição Acumulada
PIN	Personal Identification Number
AES	Advanced Encryption Standard
INRIA	Institut National de Recherche en Informatique et en Automatique
ROBERT	ROBust and privacy-presERving proximity Tracing protocol
GPS	Global Positioning System
IoT	Internet of Things
SHA	Secure Hash Algorithm

Sumário

1 Introdução

A recente pandemia causada pelo novo coronavírus revelou a necessidade de utilizar mecanismos eficientes para o controle da dispersão de doenças altamente transmissíveis. Conforme a linha do tempo divulgada por (World Health Organization,), em 31 de dezembro de 2019, a Organização Mundial da Saúde foi notificada sobre diversos casos de pneumonia na cidade de Wuhan, província de Hubei. Dias depois, as autoridades chinesas determinaram que o surto estava sendo causado por um novo coronavírus. A partir do dia 13 janeiro de 2020, o vírus havia sido confirmado em outros países da Ásia, como Japão e Coreia do Sul e logo se espalhou para a Europa e América, com casos confirmados nos Estados Unidos e França, por exemplo. Em 26 fevereiro, o vírus foi confirmado no Brasil e pouco tempo depois, a Organização Mundial da Saúde declarou a pandemia do novo coronavírus. Rapidamente o mundo todo já sofria fortes consequências da covid-19, doença causada pelo novo coronavírus.

Além das medidas de isolamento e distanciamento social associado ao uso de máscaras, governos e outras entidades buscam recursos tecnológicos para auxiliar no controle da pandemia. Aplicativos de rastreamento de contato ganharam destaque nesse cenário. O rastreamento de contatos é o processo de identificar, avaliar e gerenciar pessoas que foram expostas a uma doença para evitar a transmissão posterior. Quando aplicado sistematicamente, o rastreamento de contatos quebra as cadeias de transmissão de uma doença infecciosa, portanto, é uma ferramenta essencial de saúde pública para controlar surtos de doenças infecciosas (World Health Organization, 2020). Essa medida atrelada à realização de testes em massa e isolamento social foi extremamente importante para a contenção do vírus na Coreia do Sul (KANG et al., 2021), no entanto, a implementação desse tipo de sistema levanta preocupações sobre privacidade.

Alguns dados que podem ser coletados em sistemas de rastreamento de contato são considerados sensíveis, como nome, CPF, e-mail e localização. Além disso, o fato de uma organização registrar com quem cada indivíduo está tendo contato gera desconforto pela sensação de estarem sendo "vigiados". Outra preocupação é até que ponto os aplicativos podem ser reaproveitados para rastrear seus usuários e como os dados coletados podem ser usados quando a pandemia atual terminar (AHMED et al., 2020). As consequências da coleta massiva de dados por parte das autoridades coreanas foram levadas para Comissão Nacional de Direitos Humanos, que as classificou como uma violação aos direitos humanos (National Human Rights Commission of Korea, 2020).

Devido à alta exposição que um usuário pode ter, a adesão das pessoas à utilização de sistemas de rastreamentos de contato pode ser muito pequena, prejudicando a contenção

e análise da dispersão de um vírus altamente transmissível. Pensando nisso, este projeto tem o propósito de desenvolver um sistema de rastreamento de contato totalmente anônimo que preserva a privacidade dos usuários, tanto no contexto do servidor, quanto em relação aos usuários entre si.

1.1 Motivação e Justificativa

Sistemas de rastreamento de contato se mostraram úteis para controlar a propagação da covid-19. Na verdade, a pandemia do coronavírus não é a primeira que os seres humanos enfrentam e pode não ser a última também. Implementar sistemas que auxiliam na quebra das cadeias de transmissão de um vírus que dispersa muito rapidamente entre pessoas é muito relevante para mitigar pandemias e também amenizar a dimensão que uma pode chegar. No entanto, as pessoas devem se sentir seguras em relação aos seus dados, para terem o interesse em utilizá-los e, assim, sua efetividade seja mantida.

As arquiteturas de sistemas de rastreamento de contato existentes apresentam suas peculiaridades. Na arquitetura centralizada, os servidores têm acesso a todas as informações dos usuários e caso haja um ataque, os dados de todos os usuários são revelados, além de nem sempre ser possível confiar totalmente na organização que gerencia o servidor. Na arquitetura descentralizada, o servidor não tem acesso a nenhuma informação dos usuários, então o usuário fica responsável por identificar se teve contato com alguém infectado, o que pode acabar expondo o infectado, além de privar que os dados de contato possam ser utilizados para analisar a dispersão do vírus. Pesquisadores têm focado seus esforços em propor uma arquitetura híbrida, mas que também pode levantar questões de privacidade.

Dessa forma, para um rastreamento confiável, a quantidade de dados coletados deve ser a mínima possível. Em adição, é pertinente que os usuários tenham controle sobre os seus dados, de forma que saibam o que está sendo coletado e para que estão sendo utilizados. Este trabalho visa o desenvolvimento de um sistema que foca na anonimidade dos usuários, para que eles se sintam seguros em utilizá-lo, enquanto os dados coletados possam servir para estudos correlatos.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um sistema de rastreamento de contato que preserva a privacidade do usuário por meio da anonimidade no contexto do servidor e em relação aos indivíduos no qual teve contato, de forma que não seja possível identificar quem são os usuários infectados, além de não revelar nenhuma informação sensível dos usuários, para que se sintam confiantes em utilizar a aplicação e ainda seja possível realizar análises da rede de

contatos formada.

1.2.2 Objetivos Específicos

- Estudar arquiteturas centralizadas, descentralizadas e híbridas para sistemas de rastreamento de contato;
- Levantar problemas de privacidade em sistemas de rastreamento de contato;
- Definir arquitetura de um sistema de rastreamento de contato;
- Desenvolver sistema com base na arquitetura definida;
- Implantar sistema online para usuários utilizarem;
- Analisar dados coletados pelo sistema.

1.3 Método de Desenvolvimento do Trabalho

Para o desenvolvimento desse trabalho, inicialmente é necessário estudar sobre sistemas de rastreamento de contato e os tipos de arquiteturas existentes de maneira geral, através da leitura de artigos científicos e pesquisas em ferramentas de busca. Após o entendimento, serão levantados problemas de privacidade que podem existir em cada arquitetura.

Em continuação, com base no que foi estudado e avaliado na literatura, uma arquitetura será definida, envolvendo uma aplicação *mobile* e um servidor. Para a aplicação *mobile*, a identificação dos contatos será feita com a tecnologia *beacon*, que utiliza o recurso *Bluetooth Low Energy* (BLE). Também será utilizado um serviço de mensagens assíncrono para o envio e processamento de mensagens no servidor. Assim, o próximo passo é a implementação do sistema com base no que foi definido na arquitetura.

Por fim, com o sistema implementado, o servidor deverá ser hospedado online para o que a aplicação *mobile* possa ser testada por alguns usuários e assim, permitir a análise do funcionamento do sistema na prática. Após isso, será realizada uma análise dos dados coletados pelo sistema, onde também serão levantadas as limitações e oportunidades de melhoria.

1.4 Organização da Monografia

Além desta introdução, este modelo de monografia é composto por outros cinco capítulos:

- O Capítulo 2 apresenta os aspectos relativos ao conteúdo teórico relevante para o trabalho;
- O Capítulo 3 apresenta a descrição da proposta de um sistema de rastreamento de contatos e detalhes da implementação dele;
- O Capítulo 4 apresenta a análise dos dados coletados pelo sistema proposto;
- O Capítulo 5 apresenta as considerações finais do trabalho;

2 Referencial Teórico e Tecnologias Utilizadas

Este capítulo apresenta a fundamentação teórica e tecnologias utilizadas. Na Seção 2.1 é descrito o conceito de rastreamento de contato. Na Seção 2.2 é apresentada a tecnologia *Bluetooth Low Energy*, muito utilizada para detectar contatos em aplicativos de rastreamento. Na Seção 2.3 são destacados métodos de criptografia que podem ser utilizados para garantir a anonimidade e privacidade dos dados de um usuário no sistema. Na Seção 2.4 é apresentado sobre comunicação assíncrona, conceito importante presente na solução proposta para o envio e processamento dos dados enviados pelos usuários. A Seção 2.5 descreve trabalhos relacionados ao sistema proposto.

2.1 Sistema de Rastreamento de Contato

Sistemas de rastreamento de contato, ou *contact tracing*, são responsáveis pelo processo de identificar, avaliar e gerenciar pessoas que foram expostas a uma doença para evitar a transmissão posterior (World Health Organization, 2020). A ideia é localizar pessoas que tiveram contato com alguém infectado, ou seja, pessoas com potencial de também estarem infectadas, e passar recomendações, como fazer testagem e isolamento social. A (World Health Organization, 2020) orienta como definir um contato no contexto da covid-19, e dentre os citados, podemos destacar a pessoa que esteve menos de 1 metro de distância a um caso de covid-19 por mais de 15 minutos, onde o início da doença foi confirmado entre 2 a 14 dias atrás.

Aplicativos existentes utilizam dados de geolocalização ou sinais de *bluetooth* para identificar contatos. Nesse cenário, por questões de privacidade, serão consideradas apenas arquiteturas que utilizam sinais de *bluetooth*. Há três tipos de arquiteturas comumente utilizadas em sistemas de rastreamento de contato, a centralizada, a descentralizada e a híbrida, a qual combina conceitos das arquiteturas anteriores.

2.1.1 Arquitetura centralizada

Em (AHMED et al., 2020), é descrito que, de maneira geral, inicialmente, o usuário deve registrar-se no servidor central, provendo informações que possam ser utilizadas para identificá-los e contatá-los, caso necessário.

O servidor fica responsável por gerar identificadores (ID) temporários para cada dispositivo. A ideia de se compartilhar identificadores temporários é dificultar o mapeamento de um identificador recebido a um usuário específico da rede. Esses identificadores são criptografados com uma chave secreta (conhecida apenas pela autoridade do servidor cen-

tral) e são enviados aos dispositivos que posteriormente as compartilham entre si. Quando um usuário é confirmado com covid-19, ele deve voluntariamente enviar ao servidor todos os IDs temporários recebidos, onde eles serão descriptografados no servidor e mapeados em indivíduos que receberão uma mensagem com as recomendações necessárias. A Figura 1 mostra como é o funcionamento.

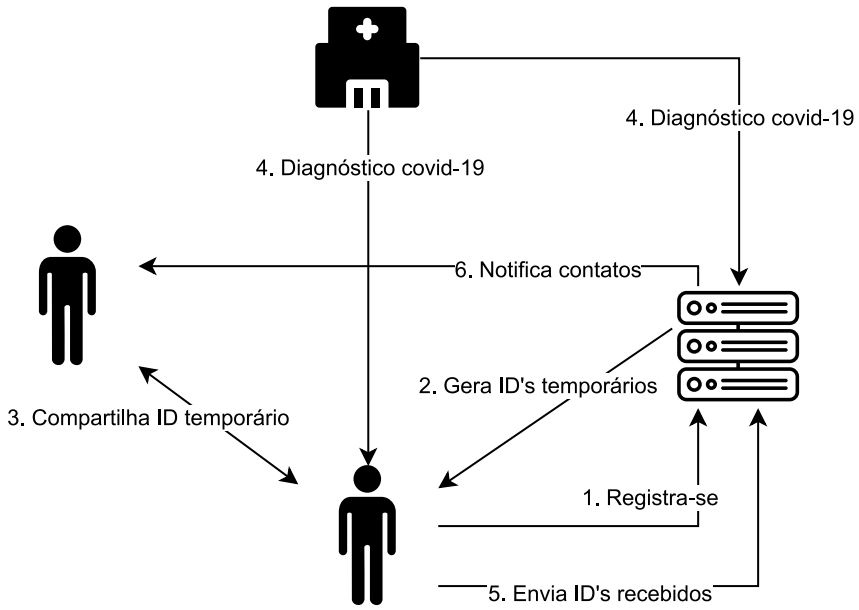


Figura 1 – Arquitetura centralizada de um sistema de rastreamento de contato. Elaborado pelo autor (2022)

Essa arquitetura requer confiança na autoridade central em relação a como os dados serão utilizados ou assegurados. Pela natureza centralizada, os dados ainda ficam vulneráveis a ataques que podem revelar a identidade das pessoas.

2.1.2 Arquitetura descentralizada

Já na arquitetura descentralizada, a principal ideia é passar a maior parte da responsabilidade aos usuários, onde o servidor tem pouca participação. Sendo assim, o usuário fica responsável por gerar os ID's temporários e também identificar se teve contato com um infectado.

Para exemplificar o funcionamento de uma arquitetura descentralizada, é tomado como base o protocolo *Private Automated Contact Tracing* definido em (RIVEST et al., 2020). Diferente da arquitetura centralizada, não é necessário fazer nenhum registro no servidor central, portanto nenhuma informação pessoal é compartilhada.

Os dispositivos são responsáveis por gerar sementes aleatórias que, combinadas com o tempo atual, produzem um pseudônimo ou ID temporário que é compartilhado com outros usuários no momento em que estiverem próximos. Quando um usuário é detectado com covid-19, ele pode enviar suas sementes e as informações de tempo relevantes para o

servidor central, que as disponibiliza para *download*. Os outros usuários, então, ao obterem as sementes e as informações de tempo do usuário infectado, reconstroem os pseudônimos e verificam se algum deles se igualam aos que têm armazenado no seu próprio dispositivo. A Figura 2 ilustra esse processo.

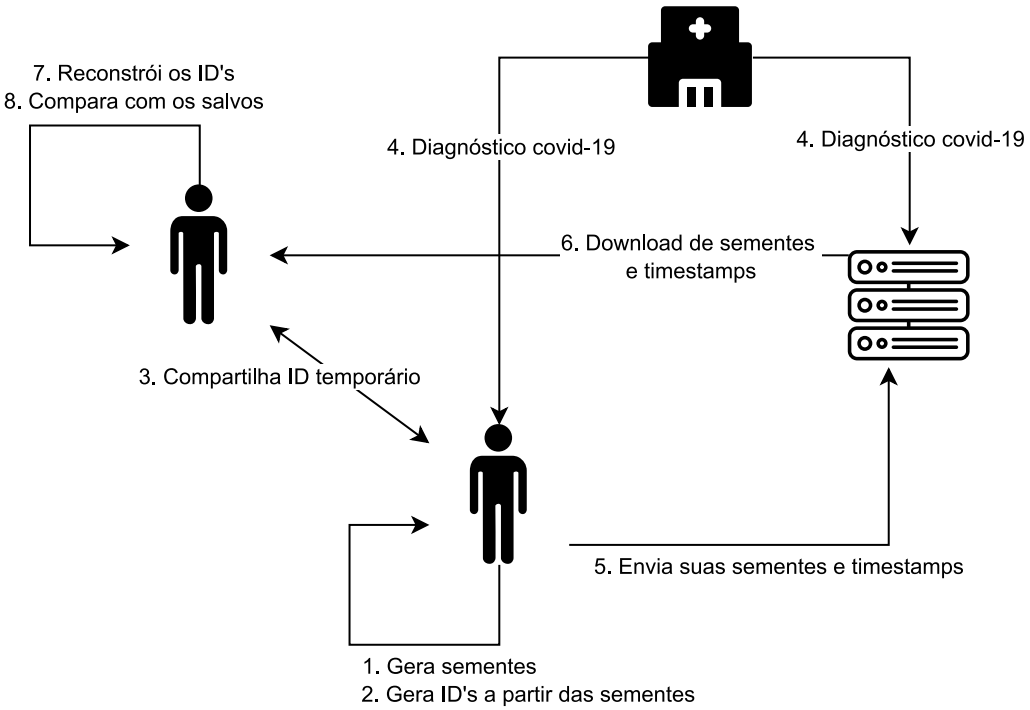


Figura 2 – Arquitetura descentralizada de um sistema de rastreamento de contato. Elaborado pelo autor (2022)

O servidor central não armazena nenhuma informação de contato dos usuários, nem mesmo outras informações pessoais, além de não performar nenhuma função muito relevante. No entanto, deixar a responsabilidade totalmente no lado dos usuários pode trazer problemas, tanto na questão da performance, quanto na questão da privacidade. Em questão de performance, é relevante considerar o uso da memória pra armazenar todos os pseudônimos de contatos e bateria, já que reconstruir todos os pseudônimos e compará-los com os que armazenou requer um custo computacional. Em relação a privacidade, pode-se citar a possibilidade de identificar exatamente quem foi a pessoa infectada na qual entrou em contato, uma vez que as sementes do usuário são disponibilizadas para todos.

2.1.3 Arquitetura híbrida

A arquitetura híbrida propõe que as tarefas sejam divididas entre servidor e dispositivo. A geração e o gerenciamento dos ID's temporários ficam sob responsabilidade dos dispositivos para garantir a privacidade e anonimização, enquanto a análise de risco e notificação de usuários devem ser de responsabilidade do servidor centralizado. (AHMED et al., 2020) cita dois motivos para que o rastreamento dos contatos seja feito no servidor.

A primeira é que, quando a identificação do contato com um infectado é feita por cada usuário individualmente, o servidor desconhece o número de usuários em risco, o que impossibilita qualquer análise estatística que facilite a identificação de *clusters* expostos. A segunda é que, as informações de contatos de usuários infectados não são disponibilizadas para os outros usuários, ficam retidas apenas no servidor. Isso evita que o usuário possa ser identificado. Essa arquitetura pode ser implementada de variadas formas.

2.2 Bluetooth Beacon

A exposição de um indivíduo ao coronavírus se da pela proximidade com uma pessoa infectada durante um período estimado em 15 minutos. Aplicativos de rastreamento devem ser responsáveis por identificar esses contatos. As tecnologias geralmente utilizadas são GPS (*Global Positioning System*) e *Bluetooth*. O GPS pode fornecer informações de localização razoavelmente precisas dentro da margem de erro, especialmente quando usado ao ar livre. No entanto, o armazenamento de informações de localização absoluta vem com custos de privacidade, se transferidos para o servidor (AHMED et al., 2020). Além disso, não funciona muito bem considerando ambientes internos.

2.2.1 Bluetooth Low Energy

A interface *Bluetooth* é um protocolo de comunicação sem fio que permite capturar valores do indicador de intensidade do sinal recebido (RSSI) por um dispositivo e assim estimar uma distância. No entanto, para identificar um contato não é necessário haver uma grande troca de dados e nem uma conexão que dure por muito tempo, logo utilizar *Bluetooth* gera um gasto de energia desnecessário. É daí que surge a oportunidade de se utilizar a tecnologia *Bluetooth Low Energy* (BLE).

De acordo com (HEYDON; HUNN, 2012), o BLE foi projetado para complementar o *Bluetooth* clássico e para ser a tecnologia sem fio de menor potência possível que pode ser projetada e construída. A transmissão de dados é geralmente em rajadas curtas que não precisam ser muito frequentes. É mais adequado para dispositivos que não exigem alta taxa de transferência ou fluxo de dados, além de permitir uma conexão mais rápida com outro dispositivo (GUPTA, 2016).

BLE usa as mesmas ondas de rádio que o *Bluetooth* e permite que dois dispositivos compartilhem dados de muitas maneiras similares. A diferença é que dispositivos BLE permanecem adormecidos entre conexões. Eles também são projetados para se comunicar apenas por alguns segundos, o que acaba economizando bateria.

2.2.2 Beacon

Na tecnologia sem fio, um *beacon* é o conceito de transmitir pequenos pedaços de informação, também chamados de anúncios de *beacon*. Os dados transmitidos são tipicamente estáticos, mas também podem ser dinâmicos e mudar ao longo do tempo (GUPTA, 2016). É utilizando o BLE que dispositivos conseguem atuar como *beacons*. A transmissão é feita por meio de ondas de rádio repetidas e constantes que outros dispositivos com a tecnologia *bluetooth* conseguem detectar. Esse sinal é composto por uma combinação de letras e números, como um identificador único, e pode ser aplicado no varejo, com o intuito de, por exemplo, fazer ofertas especializadas aos clientes dependendo de onde eles estão na loja, e também em projetos de localização *indoor*. Um dispositivo móvel que suporta a tecnologia BLE consegue agir como um *beacon* transmitindo um ID único que pode ser utilizado para detectar o contato com um infectado.

2.2.3 Protocolo AltBeacon

AltBeacon é um protocolo que define um formato de mensagem para anúncios de *beacon* de proximidade. As mensagens transmitidas por um *beacon* AltBeacon tem a finalidade de sinalizar sua proximidade a receptores próximos. O conteúdo da mensagem emitida contém informações que o dispositivo receptor pode usar para identificar o *beacon* e calcular sua distância relativa (ALTBEACON.ORG, 2014). O protocolo define o formato da mensagem transmitida como mostra a Figura 3.

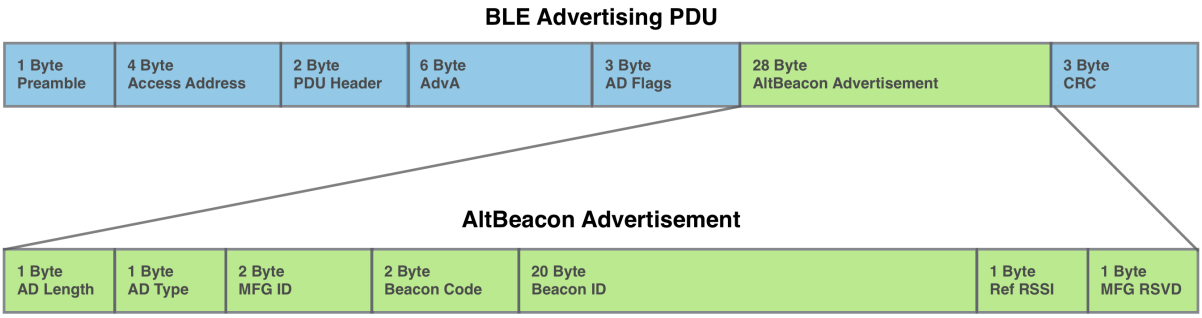


Figura 3 – Formato do protocolo AltBeacon (ALTBEACON.ORG, 2014).

A mensagem no formato AltBeacon é encapsulada como um *payload* na estrutura de dados definida na especificação do *Bluetooth* 4.0 para anúncios via *Bluetooth Low Energy*. Como mostrado na imagem, ela é composta por um campo de comprimento de 1 *byte*, um campo de tipo de 1 *byte* e o identificador da empresa de dois *bytes*, conforme prescrito pelo formato de estrutura de dados de publicidade específicos do fabricante, seguido por 24 *bytes* adicionais contendo os dados do anúncio. Dentre as informações contidas nos dados do anúncio está o RSSI de referência de 1 *byte* e o *beacon* ID de 20 *bytes* que, por questões de interoperabilidade, os primeiros 16 *bytes* devem ser únicos para a empresa

que o fabricou e os *bytes* remanescentes podem ser subdivididos da forma que for mais conveniente para o caso em que for aplicado.

2.3 Criptografia de chave pública

A criptografia de chave pública (PKC) foi inventada em 1976 por Whitfield Diffie, Martin Hellman e Ralph Merkle (RIVEST; SHAMIR; ADLEMAN, 1978). A chave pública é conhecida por todos na rede, porque é usada para criptografia, mas apenas o destinatário autorizado pode descriptografar a mensagem usando sua chave privada, que só ele conhece. Ambas as chaves são necessárias no envio e recebimento da mensagem e devido à implementação de duas chaves, é referido como criptografia assimétrica (GAITHURU et al., 2015). A segurança do PKC é baseada em problemas matemáticos difíceis, ou seja, a existência de funções unidirecionais ou funções matemáticas que são fáceis de calcular, mas sua função inversa é relativamente difícil de obter (AGRAWAL; MISHRA, 2012).

Outro detalhe importante é que, por ser assimétrica, uma mensagem criptografada com a chave privada pode ser descriptografada com a sua chave pública. Isso é importante no contexto de assinatura digital para informar que a mensagem enviada pertence ao remetente informado e a integridade dela não foi comprometida. Sistemas onde um usuário precisa garantir essas questões sem precisar informar ou comprovar uma identidade que é regida por terceiros, pode usufruir da criptografia de chave pública. Como exemplo, é válido citar *blockchains* não permissionadas, como *Bitcoin* e *Ethereum*, nas quais as transações são assinadas seguindo o conceito de chave pública. Isso se dá especialmente pela natureza descentralizada desses sistemas, onde qualquer um pode realizar transações sem ser propriamente identificado, ou seja, anonimamente. Dois algoritmos de criptografia de chave pública são bastante populares, *Rivest Shamir and Adleman* (RSA) e *Elliptic Curve Cryptography* (ECC).

2.3.1 Rivest Shamir and Adleman

O RSA foi o primeiro método de criptografia de chave pública inventado e ainda é um dos algoritmos mais populares. A dificuldade do algoritmo se dá pela dificuldade em fatorar o produto de dois números primos grandes. No entanto, a capacidade dos computadores de fatorar grandes números melhorou muito, facilitando atacá-lo. (GAITHURU et al., 2015). O ideal é que os números sejam grandes o suficiente para diminuir esse risco. A Figura 4 mostra como o algoritmo de criptografia RSA funciona.

Interlocutor **A** escolhe, secretamente, dois números primos, p e q, e publica $n = pq$. Então, **A** aleatoriamente escolhe b tal que b e $\Phi = (p - 1)(q - 1)$ são primos relativos ou primos entre si. **A** calcula a tal que $ab = 1(mod\ \Phi(n))$. Sua chave secreta é a, enquanto que b é revelado publicamente.
Interlocutor **B** cifra sua mensagem x computando:

$$y = x^b \bmod n$$

e envia y para **A**. **A** então decifra e obtém x calculando:

$$x = y^a \bmod n$$

Figura 4 – Sistema de criptografia RSA (PORTNOI, 2005)

2.3.2 Elliptic Curve Cryptography

Os pontos em uma curva elíptica que são definidos sobre um campo finito têm propriedades favoráveis para uso em aplicações criptográficas (GAITHURU et al., 2015). Como toda criptografia de chave pública, a Elliptic Curve Cryptography, ou ECC, é baseada em funções matemáticas que são fáceis de calcular em uma direção, mas muito difíceis de reverter. No caso de ECC, esta dificuldade reside na inviabilidade de calcular o logaritmo discreto de um elemento de curva elíptica aleatório em relação a um ponto de base conhecido publicamente. Portanto, a criptografia de curva elíptica fornece um nível de segurança comparável com parâmetros de sistema significativamente menores, como tamanho de chave menor, proporcionando eficiência nas implementações de software e hardware (BIEHL; MEYER; MÜLLER, 2000). Por exemplo, um tamanho de chave ECC de 160 *bits* tem nível de segurança equivalente a RSA com módulo de 1024 *bits* (GAITHURU et al., 2015). Por ser menor, é mais rápido e exige um esforço computacional inferior, além de menor espaço de armazenamento, sendo ideal para dispositivos com recursos mais limitados, como *smartphones*.

2.3.3 Algoritmo de Assinatura Digital de Curva Elíptica

O ECDSA é o algoritmo de assinatura digital baseado na criptografia de curva elíptica. Foi proposto inicialmente por Scott Vanstone em 1992 para o NIST (National Institute of Standard and Technology). Ele foi aceito em 1998 pelo ISO (International Standards Organization) [ISO 14888-3], aceito em 1999 pela ANSI (American National Standards Institute) [ANSI X69.2] e em 2000 pelo IEEE (Institue of Eletrical and Eletronic Engineers) [IEEE P1363]. Como o ECDSA foi aceito por tantas instituições de padronização, podemos assumir que ele está estável e é bastante robusto (BORGES; MOREIRA; FERREIRA, 2015).

De maneira simplificada, a chave privada `privKey` é gerada como um inteiro aleató-

rio no intervalo $[0...n-1]$. A chave pública `pubKey` é um ponto na curva elíptica, calculado pela multiplicação do ponto: $pubKey = privKey * G$ (a chave privada, multiplicada pelo ponto gerador G) (NAKOV, 2018).

Para assinar uma mensagem, de acordo com (NAKOV, 2018), o seguinte procedimento é realizado:

1. Calcula a *hash* da mensagem, no caso deste projeto utiliza-se o algoritmo SHA-1.
2. Gera um número aleatório `k` de forma segura no intervalo $[1..n-1]$
3. Calcula o ponto aleatório $R = k * G$, onde `G` é o ponto gerador, e pega a coordenada `x` dele: $r = R.x$.
4. Calcula a prova de assinatura: $s = k^{-1} * (h + r * privKey)(mod(n))$
5. Retorna a assinatura `{r,s}`

A assinatura calculada `{r,s}` é um par de inteiros, cada um no intervalo $[1...n-1]$. Ele codifica o ponto aleatório $R = k * G$, com uma prova `s`, confirmando que o signatário conhece a mensagem e a chave privada `privKey`. O remetente envia a mensagem com a assinatura ao receptor, que deve ter conhecimento da chave pública do usuário para realizar o seguinte procedimento de verificação da assinatura (NAKOV, 2018).

1. Calcula a *hash* da mensagem com o mesmo algoritmo usado durante a assinatura, no caso deste projeto utiliza-se o algoritmo SHA-1.
2. Calcula o inverso modular da prova de assinatura $s1 = s^{-1}(mod(n))$
3. Recupera o ponto aleatório usado durante a assinatura: $R' = (h * s1) * G + (r * s1) * pubKey$
4. Pega a coordenada x do ponto `R'`: $r' = R'.x$
5. Compara se $r' = r$

Se a comparação for verdadeira, então a mensagem é válida.

2.4 Comunicação assíncrona

A realização de uma tarefa síncrona ou assíncrona está relacionada ao fluxo de execução do código. No modelo síncrono uma operação precisa ser finalizada para que outra possa ser executada. No modelo assíncrono, uma operação não precisa esperar a outra ser finalizada, ao contrário disso, elas alternam o controle da execução entre si.

Dessa forma, processos ou threads executam assincronamente e precisam de mecanismos de comunicação para transferirem dados entre si, sem esperar um retorno um do outro.

2.4.1 Channels

Go é uma linguagem de programação criada pela Google e lançada em código livre em novembro de 2009. É uma linguagem compilada e focada em produtividade e programação concorrente (WIKIPEDIA, 2022). A linguagem possui recursos nativos para concorrência, como *channels* e *goroutines*. Um *channel* é um mecanismo de comunicação que conecta *goroutines* e possibilita o envio e recebimento de valores. As *goroutines* são *lightweight threads* gerenciadas pelo *runtime* do Go. São chamadas de *lightweight* devido ao custo de criação ser muito menor comparada com uma *thread* de verdade. É comum uma única *thread* conter milhares de *goroutines* sendo executadas. Caso alguma dessas *goroutines* bloqueie a execução das outras por estar aguardando alguma entrada do usuário, por exemplo, o *runtime* do Go irá criar uma nova *thread* e mover as outras goroutines para essa thread (TEMPORIN, 2021).

O tipo de valor que pode ser passado por um *channel* deve ser definido no momento da criação dele. Também é possível definir se o *channel* é *buffered* ou não, informando o tamanho do *buffer* no momento da criação do *channel*. Se o *channel* não possuir um *buffer*, o remetente é bloqueado toda vez que o receptor não estiver pronto para receber um valor. Quando o *channel* possui um *buffer*, o remetente só é bloqueado quando o *buffer* estiver cheio. A Figura 5 ilustra um *buffered channel*.

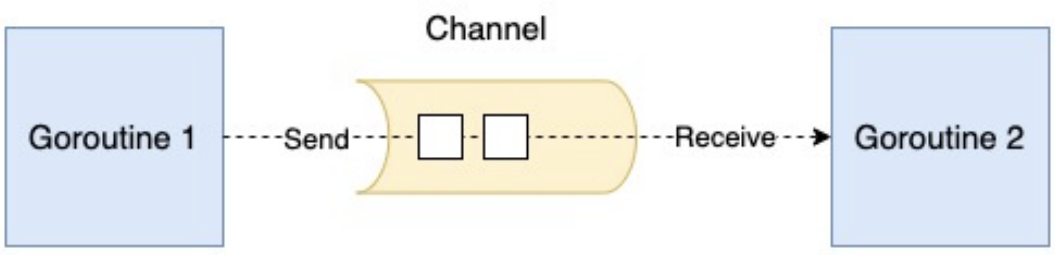


Figura 5 – *Buffered channel* em Golang (AN, 2021).

2.4.2 Protocolo Message Queuing Telemetry Transport (MQTT)

O MQTT é um protocolo de comunicação do tipo *Publisher/Subscriber* criado pela IBM e pela Eurotech em 1999. É um protocolo aberto projetado para ser simples, leve e de fácil implementação: características adequadas para dispositivos embarcados com recursos de memória e/ou processamento limitados. O pequeno *overhead* de transporte (cabeçalho de tamanho fixo de 2 *bytes*) faz do MQTT uma solução interessante para redes não-confiáveis com recursos limitados, como largura de banda e alta latência (MAZZER; FRIGIERI; PARREIRA, 2015).

O protocolo é baseado em *publishers*, *subscribers* e um *broker*. O *broker* funciona como um *middleware*, responsável por distribuir as mensagens publicadas pelas aplicações que funcionam como *publishers* para os clientes que atuam como *subscribers*, permitindo o desacoplamento entre os sistemas, em uma estrutura de comunicação assíncrona. Os *subscribers* se inscrevem em tópicos particulares, relacionados a um tipo de mensagem, já os *publishers* produzem mensagens para os tópicos.

O MQTT V3.1 suporta 3 níveis de Qualidade de Serviço (QoS), que representam a confiabilidade de entrega das mensagens (MAZZER; FRIGIERI; PARREIRA, 2015). A seleção do nível de QoS depende do caso de uso do sistema (SONI; MAKWANA, 2017).

- QoS0 (No máximo uma vez): A mensagem é enviada no máximo uma vez e não oferece garantia de entrega de uma mensagem;
- QoS1 (Pelo menos uma vez): Os dados são enviados pelo menos uma vez e é possível entregar uma mensagem mais de uma vez, definindo o valor do sinalizador duplicado em 1.
- QoS2 (Exatamente uma vez): A mensagem é enviada exatamente uma vez por meio de *handshake* de 4 vias.

Se uma aplicação cliente quiser que todos os *subscribers* recebam uma mensagem, inclusive aqueles que se inscreveram no tópico depois da publicação dela, ele deve marcar a mensagem como *retained*. No entanto, só é possível que 1 mensagem seja marcada como *retained* em um tópico e, caso outra também seja marcada, a mensagem anterior será sobrescrita.

Ambientes restritos, como dispositivos embarcados com capacidade de processamento limitada ou dispositivos conectados a uma rede instável são mais adequados para o protocolo MQTT. (SONI; MAKWANA, 2017) cita algumas aplicações que utilizam o protocolo:

- Uma organização de comunicação interpessoal de longo alcance enfrentou problemas de latência durante a transferência de informações. A estratégia que a organização utilizou para fornecer dados era estável, mas demorada e, se continuasse utilizando mecanismos similares, as soluções seriam restritas. Usando o protocolo MQTT, a organização conseguiu realizar a distribuição de dados em microssegundos, em vez de vários minutos, por meio de um *broker* MQTT.
- É importante acompanhar pacientes fora das clínicas. Uma organização trabalhou com a IBM para alcançar esse objetivo. Um cliente MQTT foi inserido em uma máquina de observação doméstica que coleta diagnósticos. Em seguida, ela encaminha

as informações indicativas pela *web* para o domínio principal, que é entregue a um aplicativo que analisa as medidas e avisa a equipe de saúde se houver indícios de que o paciente possa estar com problemas. Isso economiza dinheiro para a associação e também para os pacientes, pois há uma exigência restrita para os pacientes irem ao hospital para *check-ups* regulares.

O *broker* é o principal componente da arquitetura do protocolo MQTT. No mercado há várias opções como HiveMQ (HiveMQ,), Mosquitto (Mosquitto,) e EMQX (EMQX, 2022), a qual é a opção escolhida para este trabalho. O EMQX é um *broker* MQTT de código aberto com um mecanismo de processamento de mensagens em tempo real de alto desempenho, potencializando o *streaming* de eventos para dispositivos IoT em grande escala (EMQX, 2022). Consegue escalar 100 milhões de conexões com um único *cluster* EMQX na versão 5.0 e processar milhões de mensagens por segundo. Além disso, possui baixa latência, alta disponibilidade e permite a escalabilidade horizontal através de uma arquitetura distribuída sem mestre, ou seja, é possível adicionar novos nós facilmente para lidar com o aumento do fluxo de mensagens.

O EMQX *broker* é mais como um roteador de rede ou um *switch* em conceito, em vez da tradicional fila de mensagens de nível empresarial. Comparado aos roteadores de rede que roteiam pacotes por endereço IP, o EMQX *broker* roteia mensagens MQTT entre nós do *cluster* por modelo de *publish/subscribe Topic Trie*. Ele funciona como um único nó ou conjunto de nós, chamado *cluster* (EMQX, 2022).

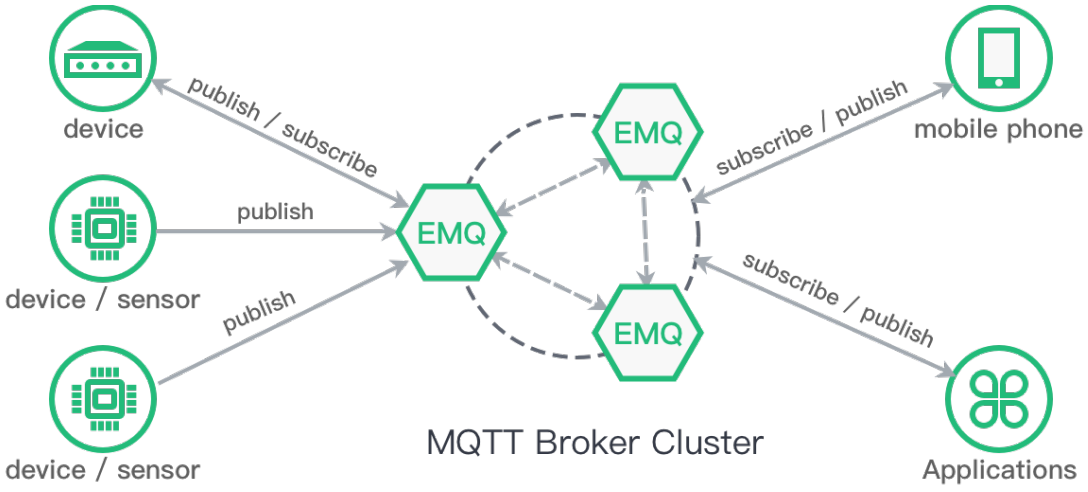


Figura 6 – Arquitetura EMQX (O'HARA et al., 2006)

Conforme a documentação, o design de roteamento segue duas regras:

- Uma mensagem só é encaminhada para outros nós do cluster se um nó do *cluster* estiver interessado nela. Isso reduz tremendamente o tráfego de rede, pois evita que os nós encaminhem mensagens desnecessárias;

- Assim que um cliente em um nó se inscreve em um tópico, ele se torna conhecido no *cluster*. Se um dos clientes em algum lugar do *cluster* estiver publicando neste tópico, a mensagem será entregue ao seu assinante, independentemente de qual nó do *cluster* ele esteja conectado.

2.5 Trabalhos relacionados

Esta seção apresenta os trabalhos relacionados na área que utilizam o *Bluetooth Low Energy* para identificar contatos e buscam a privacidade do usuário.

O artigo (LEE et al., 2021) compara abordagens centralizadas e descentralizadas de protocolos de rastreamento de contatos e propõe um protocolo hibrído. Nesse protocolo, cada usuário gera localmente uma chave e envia para uma autoridade central. A autoridade central gera uma nova chave usando a chave do usuário e a chave da autoridade e a envia para o usuário, o autor chama essa chave de *re-key*. Os usuários computam *tokens* usando a chave do usuário e a *re-key* juntas e mantém localmente a lista dos *tokens* recebidos quando os usuários entram em contato. Se um usuário for diagnosticado positivo, o usuário envia a lista de *tokens* recebidos para o servidor, que publica para o público. O usuário que receber a lista, localmente deriva os seus *tokens* e checa se os *tokens* estão na lista que recebeu. Se algum *token* bater, então significa que o usuário entrou em contato com alguém infectado.

De maneira similar, o artigo (BAY et al., 2020) propõe o protocolo chamado *BlueTrace* e uma diferença é que a autoridade central fica responsável por identificar se o usuário teve contato com algum infectado. Basicamente, os usuários devem se registrar no sistema fornecendo o número de celular. O servidor gera um identificador único e aleatório e o associa ao número de telefone. Quando dois usuários se encontram, eles compartilham entre si identificadores temporários. Os identificadores alternam com frequência para evitar que terceiros rastreiem os usuários e são gerados a partir do identificador do usuário, horário de criação e tempo de expiração, criptografados com AES-256-GCM e, em seguida, codificado em Base64. O histórico de encontro do usuário é armazenado localmente no dispositivo do usuário. Se um usuário estiver infectado ou sujeito a rastreamento de contato, ele será solicitado a compartilhar seu histórico de contato com a autoridade de saúde relevante com o uso de um PIN. Somente a autoridade de saúde consegue descriptografar o histórico de encontro compartilhado para obter e usar informações de identificação pessoal para filtrar contatos próximos e contatar usuários potencialmente infectados.

(AHMED et al., 2020) apresentam a diferença entre o protocolo *BlueTrace* e ROBERT (*ROBust and privacy-presERving proximity Tracing protocol*), protocolo centralizado proposto por pesquisadores em INRIA (França) e Fraunhofer (Alemanha) (CASTELLUCCIA et al., 2020). O primeiro ponto levantado é sobre os tipos de dados de usuário que são

armazenados no servidor. Enquanto o *BlueTrace* armazena informações pessoais (número de telefone), ROBERT armazena apenas identificadores anônimos referidos como *EphIDs*, fornecendo, então, um nível de privacidade. Os protocolos também diferem no processo de notificação, ou seja, como os usuários em risco são notificados. ROBERT requer que todos os usuários verifiquem frequentemente seus *EphIDs* usados com o servidor para determinar se eles são sinalizados como em risco. Em contraste, com o *BlueTrace*, as autoridades de saúde podem notificar proativamente os usuários em risco. O processo de notificação é possível porque o BlueTrace pode mapear os TempIDs para as informações pessoais coletadas.

O projeto proposto neste trabalho retira a responsabilidade do usuário de identificar se o mesmo se encontra em risco, diferente de (LEE et al., 2021), já que é uma tarefa que requer recursos computacionais e pode expor dados de contatos de usuários, uma vez que todos os usuários fazem *download* dos *tokens* do contaminado. Além disso, o sistema não coleta informações pessoais, como faz o protocolo *BlueTrace*, visando a anonimidade, mas mesmo assim consegue notificar os usuários, sem que eles tenham que ficar verificando constantemente se estão em risco, como apresentado no protocolo ROBERT.

3 Proposta de um sistema de rastreamento de contatos

Este capítulo apresenta a proposta de um sistema de rastreamento de contatos, que visa preservar a privacidade dos usuários por meio da anonimidade. A Seção 3.1 apresenta uma visão geral do sistema. A Seção 3.2 detalha o processo de registrar um usuário no sistema. A Seção 3.3 detalha o processo de detectar pessoas próximas. A Seção 3.4 descreve o processo de reportar covid-19. A Seção 3.5 descreve o processo de processar um contato recebido do usuário. A Seção 3.6 descreve como os dados são modelados e gerenciados. A Seção 3.7 apresenta como foi feito o *deploy* da aplicação na nuvem e a implementação do aplicativo.

3.1 Visão geral

O sistema envolve um aplicativo móvel, um servidor *web* e um servidor MQTT. O aplicativo móvel é responsável por identificar pessoas próximas, enquanto a função do servidor *web* é rastrear os contatos de um usuário infectado para detectar pessoas em potencial risco e notificá-las. O envio das mensagens de contato e de notificação são feitas de maneira assíncrona através de um servidor MQTT, chamado EMQ. A Figura 7 apresenta, de maneira simplificada, a arquitetura do sistema proposto, onde é possível identificar as fases do fluxo de funcionamento do sistema. O servidor EMQ deste projeto, para simplificar, contém apenas um nó, no entanto, é interessante utilizar um *cluster* de nós EMQ para lidar com uma demanda real de mensagens.

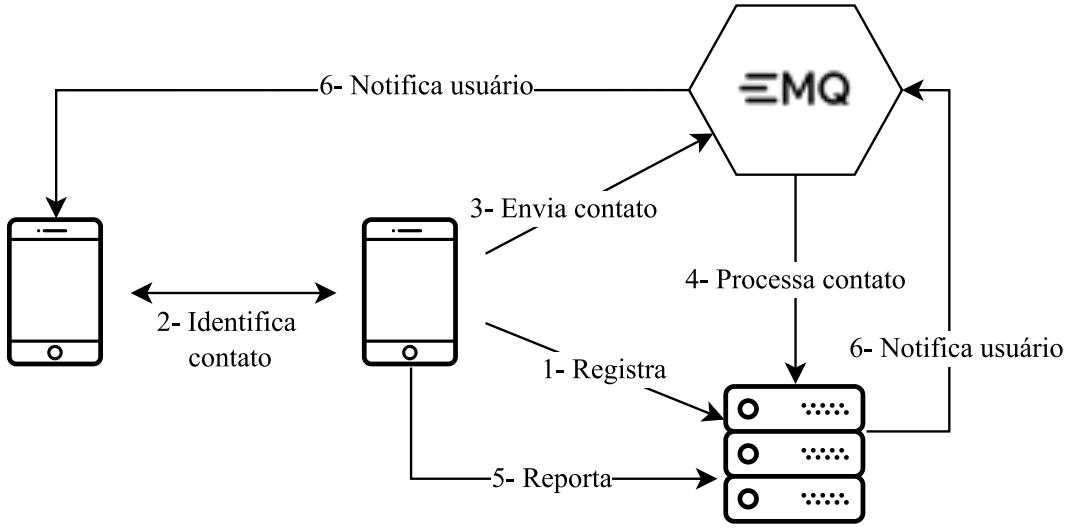


Figura 7 – Visão geral do sistema proposto. Elaborado pelo autor (2022)

O primeiro passo para utilizar o sistema é **registrar o usuário**. O sistema não deve coletar e/ou transmitir nenhuma informação que possibilite a identificação de um usuário, portanto, esse registro não pode solicitar informações usuais como e-mail ou número de telefone. O vínculo é feito através de uma informação única pertencente ao dispositivo, que não está ligada a qualquer informação da pessoa por trás daquele aparelho. No caso de dispositivos com sistema operacional android, é possível obter um identificador do sistema.

Ao fim do registro é gerado um identificador único universal (UUID) do usuário. Esse identificador é essencial na fase de **identificar pessoas próximas ou contatos**. Através do recurso BLE, contido em aparelhos com *Bluetooth* a partir da versão 4.0, cada *smartphone* pode transmitir e escanear identificadores com um formato definido. Com a intensidade do sinal recebido, é possível estimar a distância entre esses dispositivos. Eles devem armazenar informações relativas ao contato e a cada intervalo predefinido de tempo, enviar esses dados para o *middleware* pub/sub EMQ, por meio do protocolo MQTT, para ser, posteriormente, processado pelo servidor *web*.

A próxima fase do fluxo é **processar as mensagens de contato** recebidas do servidor MQTT, que contém as informações do encontro entre duas pessoas. Primeiro, o sistema confere a integridade da mensagem e descarta se for inválida. Em seguida, salva esse contato em um banco de dados. Um dos usuários em contato pode ter sido diagnosticado com covid-19 recentemente, portanto, é importante verificar essa informação a cada contato recebido. Caso um dos dois esteja contaminado, a duração do contato seja maior ou igual a 15 minutos e a distância entre eles menor que 2 metros, a outra pessoa deve ser notificada.

Ao receber um diagnóstico positivo, é de responsabilidade do usuário reportá-lo ao sistema. O sistema não valida o diagnóstico, portanto, está sujeito a falsos positivos. Ao **reportar covid-19**, o usuário deve informar a data do diagnóstico e enviá-la ao servidor *web*. Assincronamente, o servidor rastreia os contatos desse usuário nos últimos 15 dias salvos no banco de dados, seguindo as regra de distância e duração mínima citadas anteriormente.

Ao identificar um usuário em risco de contágio, o servidor é responsável por **enviar uma notificação de risco** para que a pessoa siga as recomendações da OMS e evite a propagação do vírus. Essa notificação é enviada através de uma mensagem publicada em um tópico do servidor MQTT específico para o usuário em risco. O dispositivo do usuário fica aguardando por mensagens desse tópico, mesmo *offline* no momento, deve receber ao conectar-se a internet e exibir uma notificação visual.

O aplicativo móvel foi desenvolvido em Java para o sistema operacional Android, por ser uma linguagem com ampla documentação disponível e ter suporte nativo no Android. A escolha de desenvolver apenas para Android deve-se à restrição de recurso

físico, uma vez que a autora não possui acesso a um dispositivo iOS. Enquanto o servidor foi desenvolvido na linguagem Go, por ter recursos nativos para concorrência, utilizando banco de dados relacional PostgreSQL e Redis como cache.

3.2 Registro

O usuário não precisa prover nenhuma informação para iniciar e completar o registro no sistema. Os dados necessários são obtidos pela própria aplicação, são eles: um par de chaves assimétricas e o identificador do sistema. O par de chaves é necessário para autenticação do dispositivo. Como não há nenhuma informação do usuário, o dispositivo utilizado por ele que é de fato registrado no sistema. O ID do sistema, *Service Set Android Identifier* (SSAID) ou também chamado de Android ID, é a única informação associada ao *smartphone* utilizada.

A aplicação gera um par de chaves com o algoritmo de curva elíptica utilizando a biblioteca `java.security` do Java e armazena na memória do aplicativo. A chave pública é posteriormente compartilhada com o servidor no processo de registrar, enquanto a chave privada é utilizada pelo dispositivo para assinar as mensagens enviadas. Ao verificar a assinatura com a chave pública armazenada do usuário, o servidor consegue atestar que a mensagem é de quem se diz ser, garantindo o controle de acesso e integridade da mensagem.

Conforme a documentação do Android, o SSAID é um número de 64 *bits* expresso como uma *string* hexadecimal, exclusivo para cada combinação de chave de assinatura de aplicativo, usuário e dispositivo. O Android exige que todos os aplicativos sejam assinados digitalmente com um certificado antes de serem instalados em um dispositivo ou atualizados. Portanto, o identificador é único para a aplicação que o obtém, mesmo que ele seja desinstalado e reinstalado. Esse identificador está disponível a partir do Android 8.0 (API 26). O valor pode mudar caso o usuário formate o celular para o padrão de fábrica ou a chave de assinatura do aplicativo mude. Dessa forma, se o usuário trocar de dispositivo ou formatá-lo, um novo ID do sistema é obtido, requerendo um novo registro no sistema de rastreamento de contatos. Por questões de segurança, a documentação do Android não recomenda a utilização de identificadores de *hardware*, por exemplo, endereço MAC, como identificadores exclusivos pela aplicação.

A Figura 8 mostra o processo de registro do usuário no sistema. Ao abrir o aplicativo, a aplicação busca na memória um par de chaves salvo e um identificador gerado pelo servidor a partir do seu SSAID. As situações em que o aplicativo não encontrará essas informações são: primeiro acesso do usuário no aplicativo após o *download*, tendo registro ou não, e após apagar os dados do aplicativo sem desinstalar.

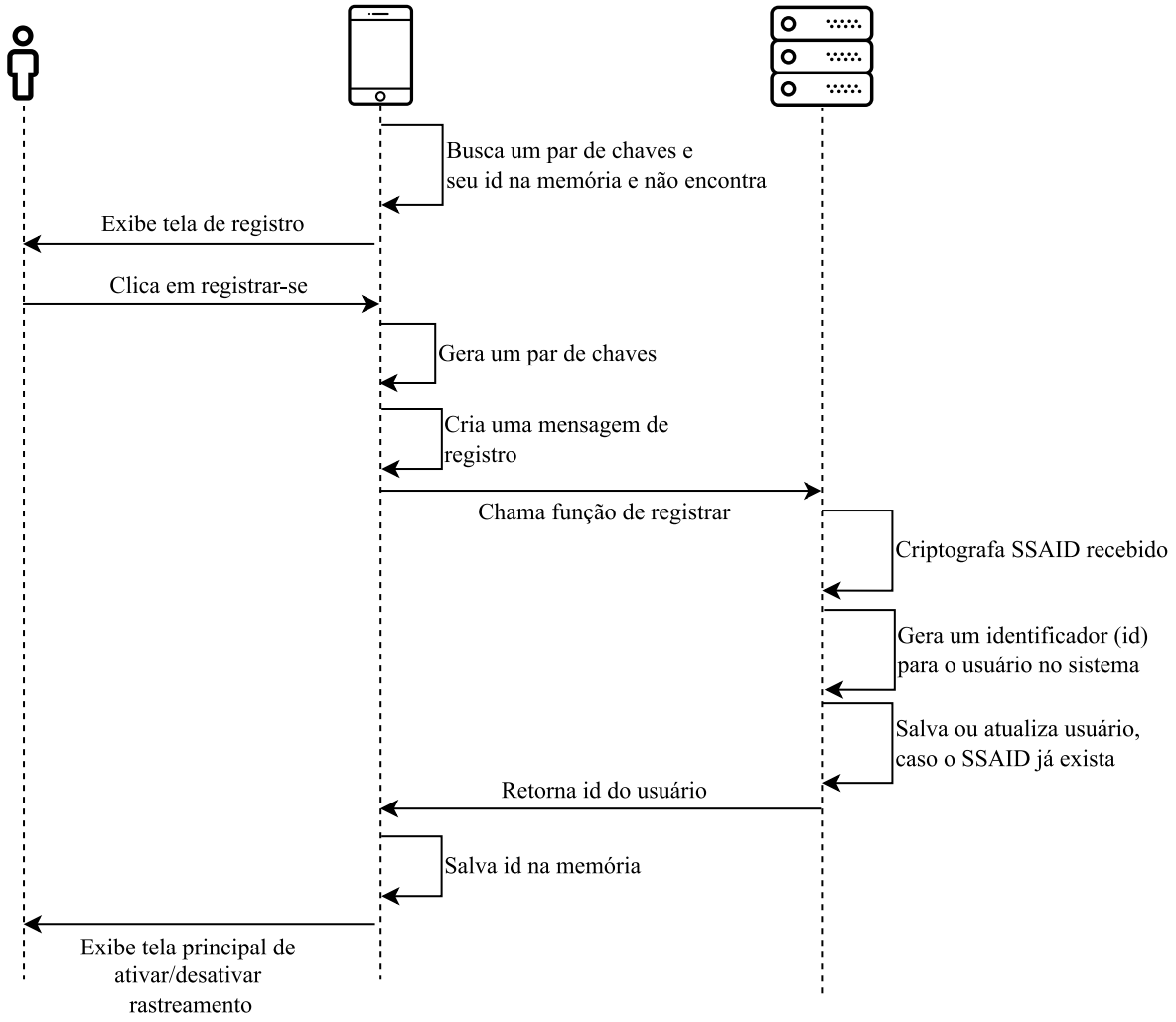


Figura 8 – Processo de registrar um usuário no sistema. Elaborado pelo autor (2022)

Quando não encontra o par de chaves e/ou o identificador, uma tela solicitando registro é apresentada para o usuário, que confirma sua inscrição no sistema clicando em um botão. Com a confirmação do usuário, o aplicativo gera o par de chaves assimétricas e salva na memória. A mensagem de registro necessita de 2 informações: o SSAID (deviceId) e a chave pública (pk), ambos em hexadecimal, como no exemplo abaixo 3.1.

Listagem 3.1 – Mensagem de registro

```
{
  "pk": "3059301306072a8648..."
  "deviceId": "65ab8c72d904..."
}
```

Esses dados são enviados para o servidor por meio de uma chamada de procedimento remoto (gRPC) da API do servidor *web*. O servidor, então, criptografa o SSAID e gera um identificador único (UUID) de 16 *bytes*, que será usado pelo aplicativo móvel no momento da detecção de pessoas próximas. O SSAID não pode ser compartilhado com outros usuários, já que é uma informação privada usada para registrar ou atualizar um

usuário. Basicamente, é análoga a uma senha que seria informada pelo usuário em outros mecanismos de autenticação.

O identificador gerado, o SSAID criptografado e a chave pública são salvos no banco de dados. Caso já exista um registro com o mesmo SSAID criptografado, a chave pública é atualizada, porque indica que o usuário se mantém com o mesmo dispositivo, mas desinstalou o aplicativo ou limpou a memória dele, o que fez com que fosse necessário gerar um novo par de chaves. Como o SSAID é único para aquele aplicativo e usuário naquele dispositivo, pode-se assumir ser a mesma pessoa. Nesse caso, o UUID gerado é descartado e o identificador já contido no banco é mantido.

O servidor, se tudo der certo, retorna o identificador para o usuário, que será persistido na memória do aplicativo. Agora, com as chaves geradas e o UUID do usuário, a aplicação apresenta a tela em que é possível ativar o rastreamento de contatos.

3.3 Detectar contato

As chances de contágio aumentam com contato próximo e prolongado com uma pessoa infectada. A interface *Bluetooth* permite capturar a força do sinal recebido do outro dispositivo e assim, estimar uma distância que auxiliará na avaliação de risco. Neste projeto, os dispositivos móveis atuam como *beacons*, através da tecnologia *Bluetooth Low Energy*, transmitindo um identificador único através do protocolo *AltBeacon* utilizando a biblioteca *Android Beacon Library*. O usuário registrado possui um UUID de 16 bytes, que será utilizado pelo serviço *beacon* da aplicação.

Ao habilitar o rastreamento de contatos, o dispositivo inicia o serviço beacon de transmissão e escaneamento em *foreground*, mantendo uma notificação fixa enquanto o serviço estiver ativo, para que o aplicativo não seja morto, caso o usuário o feche. A Figura 9 mostra o fluxo de detectar e enviar contatos para o servidor MQTT.

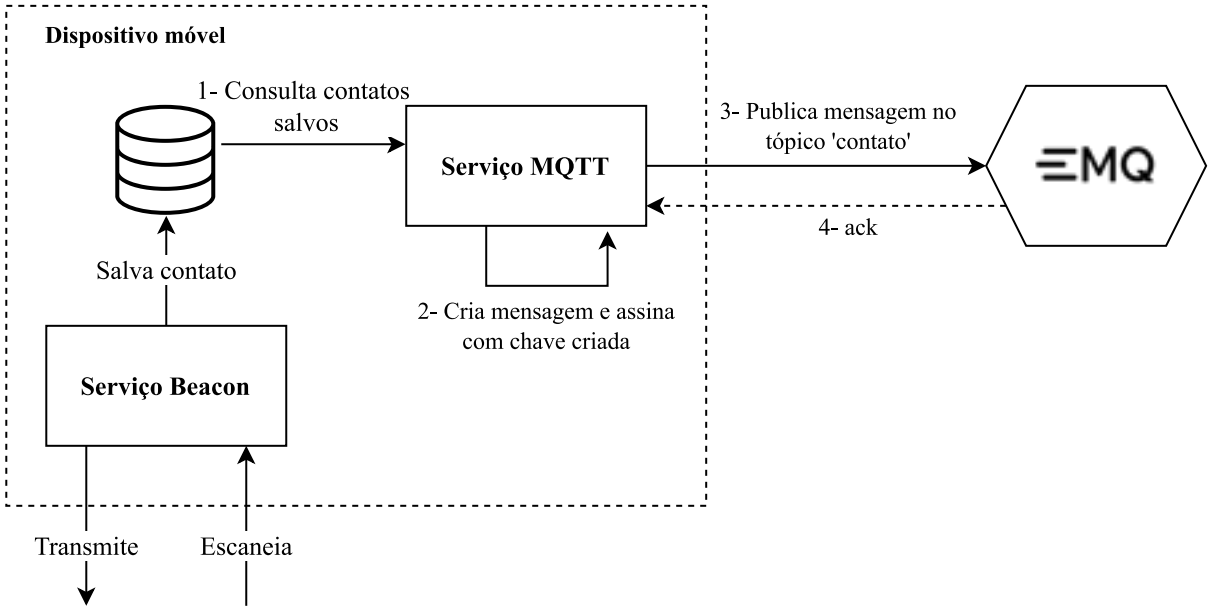


Figura 9 – Processo de detectar proximidade e enviar contato. Elaborado pelo autor (2022)

O serviço escaneia *beacons* próximos a cada 15 segundos e armazena-os como contatos em um banco de dados local SQLite. Um contato contém o identificador recebido, o *timestamp* do início do contato, o *timestamp* do fim do contato, a distância média aproximada, a intensidade do sinal recebido (RSSI) e o nível de bateria do usuário que recebeu aquele *beacon* no momento. Para poupar memória, se o usuário receber *beacons* com o mesmo identificador dentro de um intervalo de 2 minutos com uma variação de distância de no máximo 1 metro entre as mensagens dos *beacons*, então o último registro de contato salvo no banco de dados referente àquele identificador é atualizado com um novo *timestamp* de fim de contato e a distância média recalculada. Caso o tempo entre as mensagens de *beacons* seja maior que 2 minutos ou a variação de distância maior que 1 metro, é necessário adicionar um novo registro de contato na tabela de contatos do banco.

Ao habilitar o rastreamento de contatos, também é iniciado o serviço MQTT em *foreground* com um cliente MQTT que publica mensagens de contato no tópico `contato` do servidor EMQ, atuando como um *publisher*. A mensagem deve conter os dados do contato, o ID do contato no banco e o identificador do usuário que está enviando o contato.

Listagem 3.2 – Mensagem de contato

```
1    "id": 1,
2    "contact": {
3        "otherUser":  "123e4567-e89b-12d3-a456-426655440000",
4        "firstContactTimestamp": 2435465634,
5        "lastContactTimestamp": 2435465815,
6        "distance": 120, // centímetros
7        "rssi": -15,
8        "batteryLevel": 51
9    },
10   "user": "3d0ca315-aff94fc2-be61-3b76b9a2d798"
```

```
11 }
```

O sistema de autenticação se baseia em uma assinatura feita com a chave privada do usuário para cada mensagem enviada, garantindo autenticidade e integridade da mensagem. Neste caso, a parte da mensagem assinada é o campo `contact`, que contém todas as informações do contato. Essas informações são formatadas para uma *string* JSON, a qual é assinada com a chave privada salva no dispositivo do usuário. Dessa forma, o campo `signature` é adicionado na mensagem para ser validado pelo servidor, o qual contém um *array* de byte relativo à assinatura.

Listagem 3.3 – Mensagem de contato com assinatura

```
{
    "id": 1,
    "contact": {
        "otherUser": "123e4567-e89b-12d3-a456-426655440000",
        "firstContactTimestamp": 2435465634,
        "lastContactTimestamp": 2435465815,
        "distance": 120, // centímetros
        "rssi": -15,
        "batteryLevel": 51
    },
    "user": "3d0ca315-aff94fc2-be61-3b76b9a2d798",
    "signature": [48,69,2,33,0,204,95,194,149,2,35,126,50,19,34,248, ...]
}
```

A cada 30 minutos, o serviço consulta no banco de dados no máximo 60 registros de contatos para publicar no tópico. Esse número foi escolhido arbitrariamente para limitar o envio de mensagens, evitando que o serviço passasse muito tempo realizando essa tarefa. Ao receber uma confirmação do envio da mensagem de contato, um *acknowledgment*, o serviço remove o registro de contato do banco de dados. É importante destacar que, todos os *beacons* recebidos são salvos como registros de contatos, mesmo aqueles em que a distância é superior àquela considerada arriscada.

3.4 Reportar covid-19

Quando uma pessoa é diagnosticada com covid-19, as pessoas com quem teve contato nos últimos 15 dias por pelo menos 15 minutos devem ser notificadas sobre um possível risco de contágio. No entanto, nenhuma informação pessoal da pessoa infectada pode ser revelada, nem mesmo ao servidor *web*. É por isso que não é de responsabilidade deste sistema validar o diagnóstico positivo, sendo, portanto, vulnerável a falsos positivos. Desse modo, no contexto da aplicação, a única validação feita é se o diagnóstico reportado é realmente daquele usuário que se diz ser, através da validação da assinatura da mensagem. A Figura 11 apresenta o fluxo de reportar um diagnóstico positivo.

Para reportar covid-19, o usuário deve informar a data de início dos sintomas e a data em que recebeu o diagnóstico positivo (1). Caso seja assintomático, é recomendado informar a data de início dos sintomas igual à data do diagnóstico. A mensagem de diagnóstico positivo contém as datas informadas pelo usuário e a data em que o usuário reportou, ou seja, a data do dia em que enviou a mensagem ao servidor. A mensagem é enviada pela API gRPC do servidor *web*, portanto o tipo da data enviada está no formato `google.protobuf.Timestamp`(3). A assinatura é referente ao conteúdo do campo `report`.

Listagem 3.4 – Mensagem de reportar covid-19

```
{
    "report": {
        "userId": "f234454c-f6d4-a0fa-df2f-4911ba9ffa6",
        "dateStartSymptoms": {
        "seconds": 853548512,
        "nanos": 3435453434
      },
      "dateDiagnostic": {
        "seconds": 853548512,
        "nanos": 3435453434
      },
      "dateReport": {
        "seconds": 853548512,
        "nanos": 3435453434
      }
    },
    "signature": [48,69,2,33,0,204,95,194,149,2,35,126,50,19,34,248, ...]
}
```

No servidor, ao receber a mensagem, o primeiro passo é validar a assinatura da mensagem com a chave pública do usuário que se diz ser (4). Caso seja válida, o diagnóstico é salvo no banco de dados (5) e também na cache (6). É importante salvar esse diagnóstico na cache com uma expiração de 15 dias para que, posteriormente, quando um usuário enviar um contato com uma pessoa infectada, o acesso ao diagnóstico seja rapidamente consultado e ele, seja notificado do risco de contágio. Por fim, o servidor retorna sucesso na chamada da função (8). No entanto, o processamento desse diagnóstico continua acontecendo assincronamente.

Este processo é composto por duas rotinas que processam de maneira assíncrona tarefas específicas colocadas em um *channel* com tamanho máximo de 50 objetos. São elas: rastreador de contatos e notificador.

O rastreador de contatos é responsável por buscar os contatos enviados pela pessoa infectada e verificar se há risco de contágio. Essa rotina recebe tarefas por meio de um *channel*, onde um objeto é passado com as seguintes informações: ID do diagnóstico no banco de dados, UUID do usuário infectado e a data do diagnóstico. Portanto, antes de propriamente retornar sucesso na chamada da função da API, uma tarefa é adicionada no *channel* do rastreador de contatos (7). A tarefa é o objeto com as informações apresentadas

anteriormente.

A primeira ação no processamento dessa tarefa é performar uma *query* no banco de dados PostgreSQL (9) filtrando os contatos daquela pessoa no período de 15 dias antes da data do diagnóstico até o dia atual, em que a distância é menor que 2 metros. Esses dados podem ser agregados para definir uma duração de contato com cada usuário. Sendo assim, o segundo passo é agregar esses contatos (10) de forma que, se a diferença entre dois contatos entre mesmos usuários for menor que 20 minutos, então pode-se assumir um contato constante, ou seja, a diferença de tempo entre esses dois contatos será incluída na duração do contato, portanto, esses dois contatos são agregados em apenas 1. A Figura 10 ilustra um exemplo deste procedimento.

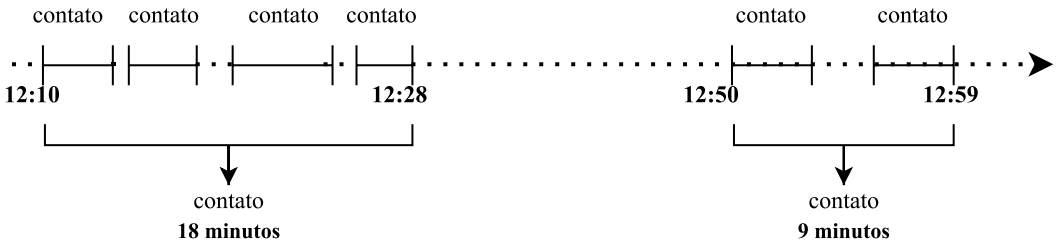


Figura 10 – Agregação de contatos entre duas mesmas pessoas. Elaborado pelo autor (2022)

O próximo passo é, então, avaliar o risco de contágio de cada usuário a partir dos contatos. Se um contato tem duração menor que 15 minutos, este usuário é considerado fora de risco, caso contrário deve-se criar uma tarefa para a rotina de notificação. Caso algo dê errado durante o processamento da tarefa de rastrear contatos, por exemplo, uma falha na conexão com o banco de dados, a tarefa é recolocada no *channel* para ser reprocessada depois de um tempo definido, tendo um limite máximo de tentativas.

O notificador é a rotina de notificação responsável por notificar usuários em risco de contágio, portanto, uma das funções é enviar mensagens para o tópico de notificação de risco de contágio de um determinado usuário. Ela recebe tarefas por meio de um *channel* (11). A tarefa é representada como um objeto, que contém seguintes informações: ID do diagnóstico no banco de dados, ID do usuário em risco de contágio e data do contato em que usuário foi exposto.

Ao receber uma tarefa, o notificador consulta a cache para verificar se há uma notificação para aquele usuário referente àquele diagnóstico (12). Se existir, o usuário não precisa ser notificado novamente. Veja que, se existir uma notificação referente ao diagnóstico de outra pessoa e não àquela da tarefa em questão, o usuário deve ser notificado novamente, já que se refere a outra pessoa infectada em que ele entrou em contato, significando um novo risco. Agora, se não existir uma notificação na cache, então o usuário deve ser alertado do risco. Sendo assim, além de salvar uma notificação no banco de dados (14) e também na

cache (15), o notificador deve publicar uma mensagem no tópico de notificação do usuário (13). O tópico segue o formato `notificacao/<userId>`, onde `userId` é o ID do usuário no servidor. Por exemplo, o usuário com ID `f234454c-f6d4-a0fa-df2f-4911ba9ffa6` subscreve no tópico `notificacao/f234454c-f6d4-a0fa-df2f-4911ba9ffa6` para receber notificações (1). Sendo assim, se esse usuário está em risco, então é nesse tópico que o servidor vai publicar uma mensagem similar a mostrada abaixo.

Listagem 3.5 – Mensagem de risco de contágio

```
{
  "risk": true,
  "message": "Você esteve em contato com uma pessoa diagnosticada com covid-19
     nos ultimos 15 dias."
}
```

Pode acontecer do usuário reportar covid-19 depois de mais de 15 dias do diagnóstico positivo. Nesse caso, os usuários em contato não devem receber uma notificação, já que não se pode considerar mais em risco. No entanto, é interessante que uma notificação seja salva apenas no banco de dados. Desse modo, o notificador, antes de salvar na cache e publicar no tópico, deve verificar se o usuário ainda está em risco.

A cache é utilizada para evitar consultas no banco de dados a toda tarefa que for recebida, o que demandaria um tempo maior do que uma consulta direta nela. É possível determinar um tempo de expiração para os dados salvos na cache, no entanto, o usuário é considerado em risco por 15 dias e, após isso, deve ser notificado como fora de risco, caso ele não tenha entrado em contato com mais nenhum outro infectado. Para isso, há também outra rotina que diariamente remove notificações expiradas na cache e notifica os usuários fora de risco. Basicamente, essa rotina verifica se, para cada notificação contida na cache, há mais de 15 dias desde o dia em que ela foi criada, em caso positivo, remove-a da cache e, se não existir mais nenhuma notificação na cache para aquele usuário, envia uma mensagem para o tópico de notificação do usuário indicando que não está mais em risco, como a mensagem abaixo.

Listagem 3.6 – Mensagem de fora de risco de contágio

```
{
  "risk": false,
  "message": ""
}
```

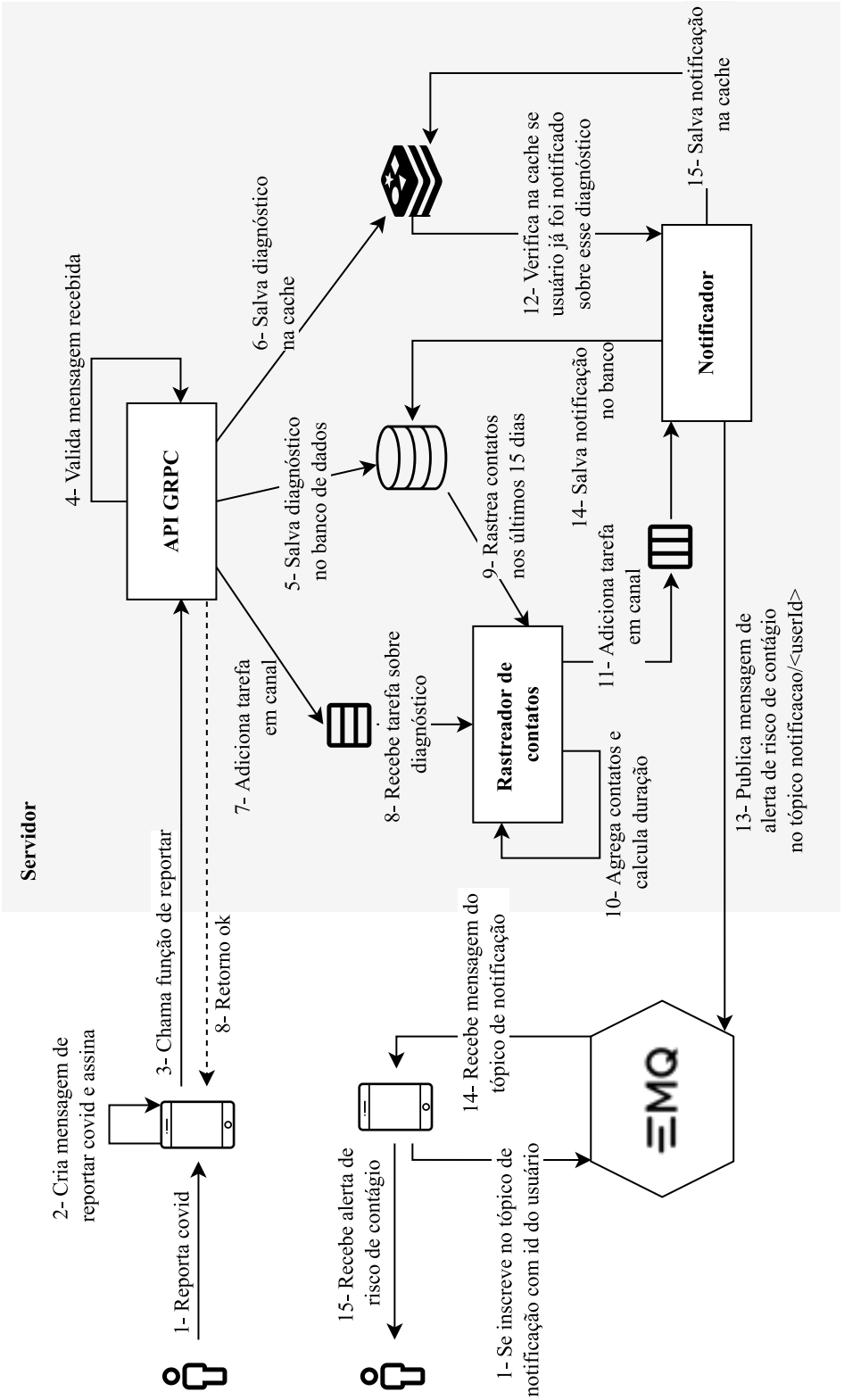


Figura 11 – Processo de reportar covid-19. Elaborado pelo autor (2022)

3.5 Processar contato

Cada contato publicado por um usuário no tópico `contato` deve ser processado pelo servidor de forma assíncrona. Para isso, o servidor age como um cliente MQTT que subscreve nesse tópico, registrando um *callback* ou *handler* para processar as mensagens recebidas. Vamos chamar esse *handler* de processador de contatos. A Figura 12 mostra o que acontece no servidor quando um contato é recebido.

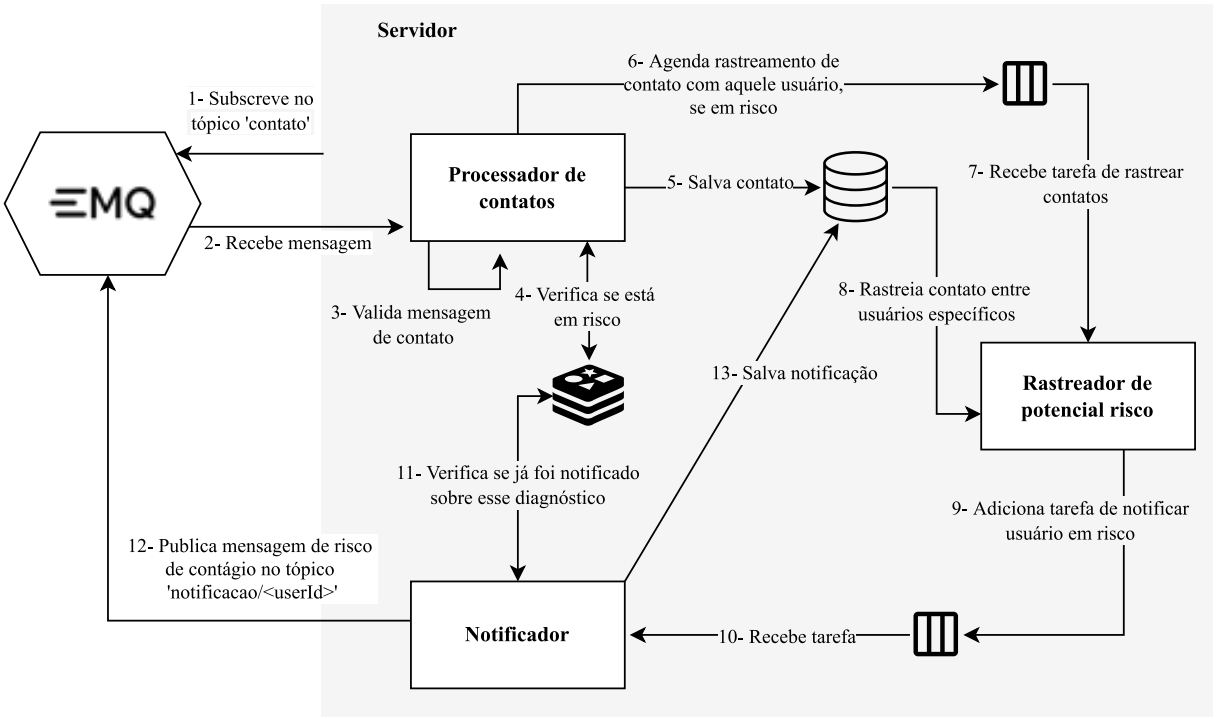


Figura 12 – Diagrama do processamento de um contato recebido no servidor. Elaborado pelo autor (2022)

Tal qual o processo de reportar covid-19, o processador de contatos valida a mensagem recebida a partir da assinatura contida nela. Se é válida, o contato é salvo no banco de dados e o processador verifica se o usuário pode estar em risco para que um rastreamento seja agendado. Além do processador de contatos, duas rotinas atuam nesse processo: notificador e rastreador de potencial risco. A rotina chamada notificador é a mesma apresentada na Seção 3.4. O rastreador de potencial risco é similar ao rastreador de contatos também apresentado na seção anterior, no entanto, sua tarefa é rastrear o contato entre dois usuários específicos entre o dia do contato e a data do diagnóstico do usuário infectado. Essa rotina recebe tarefas por meio de um *channel*, onde um objeto é passado com as seguintes informações: ID do diagnóstico no banco de dados e ID do usuário infectado. Assim como o rastreador de contatos, os contatos rastreados são agregados para verificar se a duração de algum contato foi superior a 15 minutos e caso isso aconteça, uma tarefa para o notificador é criada.

O potencial risco é avaliado se há um diagnóstico para algum dos dois usuários do contato salvo na cache. O usuário em risco não deve ser notificado nesse momento, uma vez que o contato pode ter uma duração muito pequena, insignificante para alertar do risco. Outro fato é que, se o usuário receber uma notificação quase que no mesmo momento em que estiver em contato com um infectado, a anonimidade do infectado seria violada, já que o usuário em contato poderia facilmente identificá-lo. Dessa forma, após 2 horas deve ser realizada uma busca dos contatos entre esses dois usuários no período da data do diagnóstico até o momento atual. Isso é feito através do agendamento de uma tarefa para o rastreador de potencial risco. Se já existir um agendamento, uma nova tarefa não pode ser criada. Caso seja encontrado um contato com duração maior que 15 minutos, uma tarefa de notificação é criada para o notificador. Esse fluxo de tomada de decisão pode ser visualizado na Figura 13.

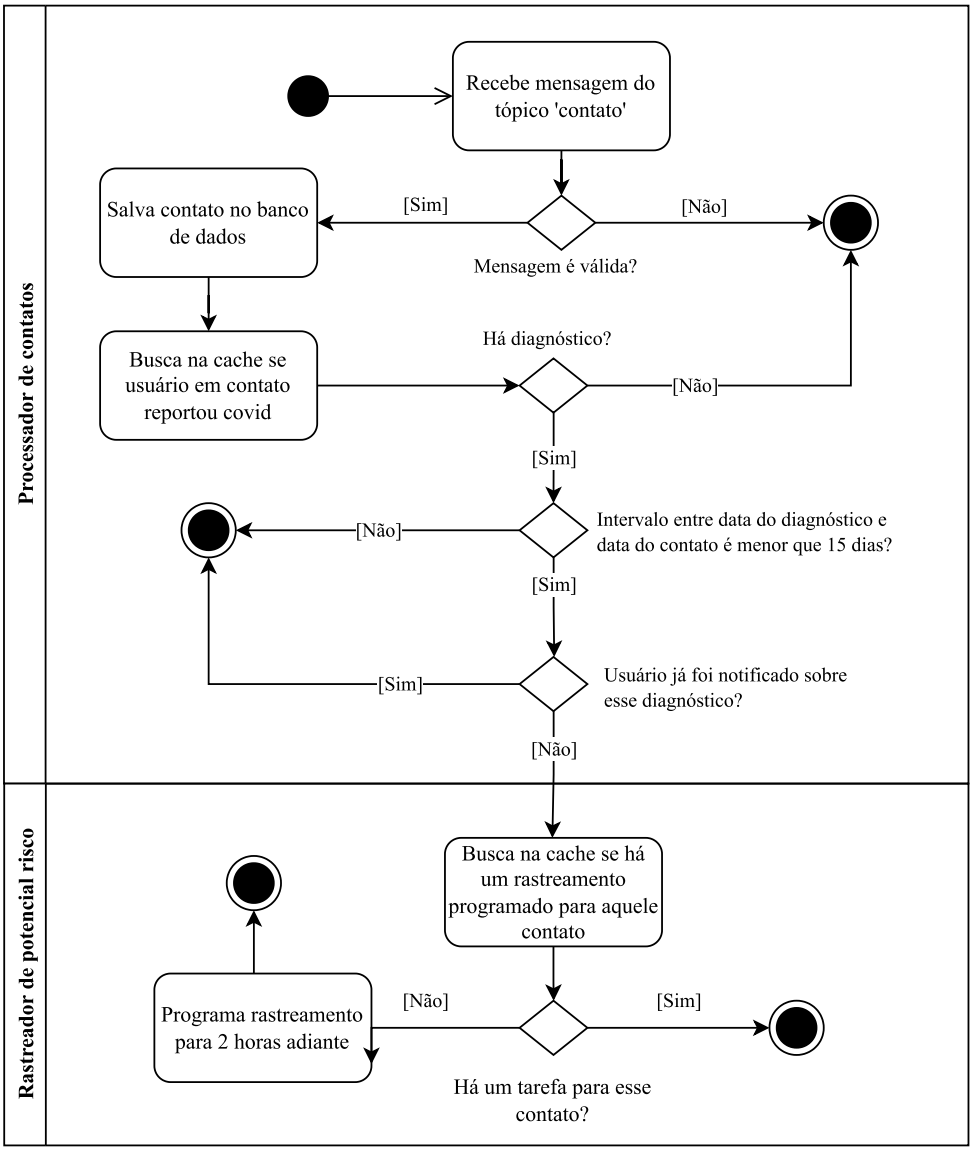


Figura 13 – Fluxo de decisão do processamento de um contato. Elaborado pelo autor (2022)

3.6 Gerenciamento dos dados

Em cada fase do fluxo do sistema são produzidos e compartilhados dados. Cada componente do sistema é responsável por gerenciar uma parte dos dados, mas é no servidor onde todos os dados são definitivamente mantidos. Na fase de registro, a responsabilidade de geração de dados é dividida entre servidor e dispositivo. Na fase de detecção de proximidade, os dispositivos móveis geram os dados de contato a partir das informações compartilhadas entre eles. Nas fases onde é possível identificar um risco de contágio, ou seja, reportar covid-19 e enviar contato, além das informações obtidas do usuário, é gerado um novo dado no banco de dados para cada notificação de risco.

As informações que devem ser armazenadas no sistema são: informações de usuário, informações de contato, informações de diagnóstico e, por fim, informações de uma notificação. Esses dados, apresentados em cada fase do sistema, são armazenados em um banco de dados relacional PostgreSQL. A Figura 14 mostra o diagrama entidade-relacionamento do sistema.

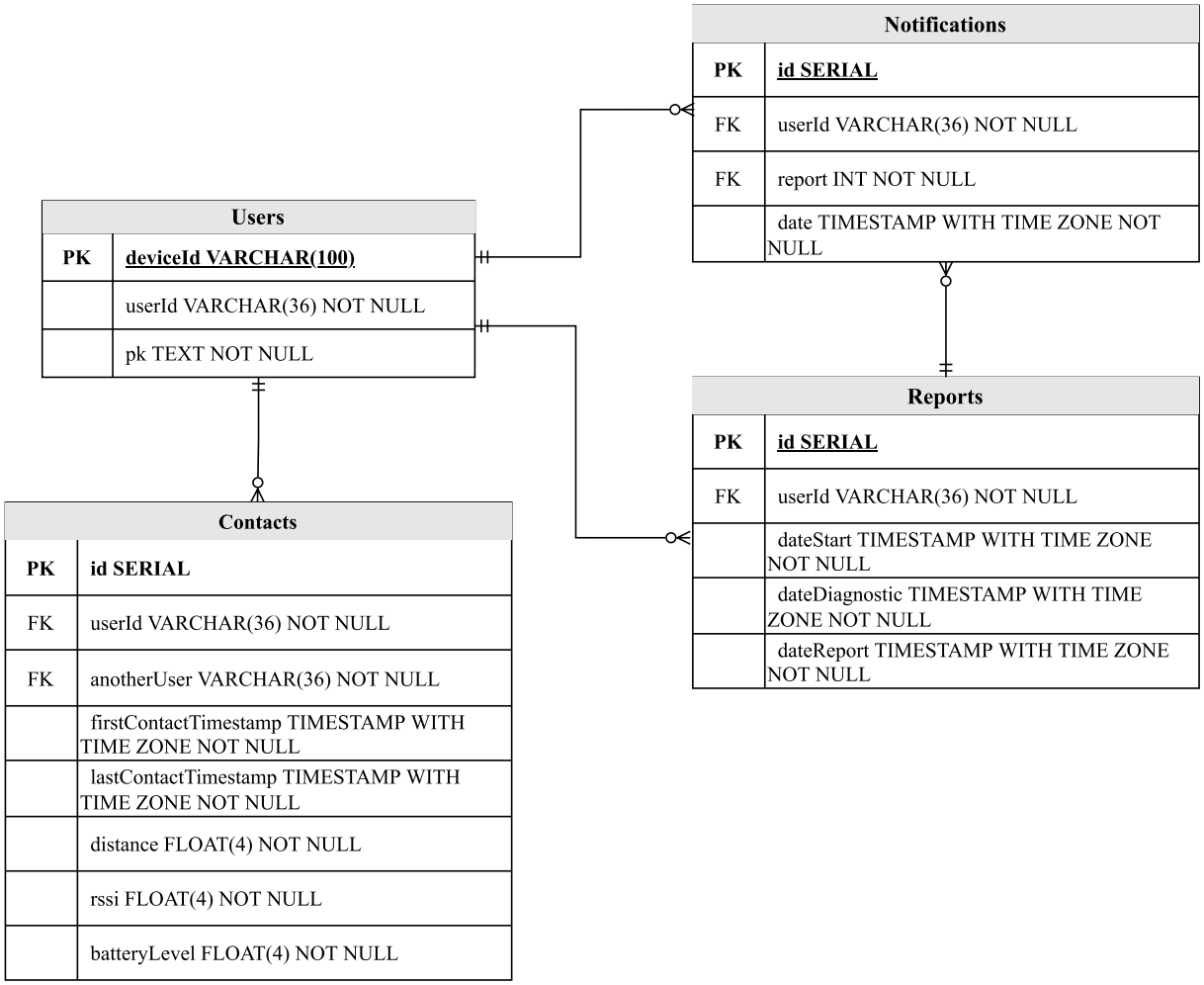


Figura 14 – Diagrama entidade-relacionamento. Elaborado pelo autor (2022)

3.7 Apresentação da aplicação

3.7.1 *Deploy* dos servidores na nuvem

Para que o aplicativo móvel se comunique com servidor *web* e servidor MQTT, é necessário que eles estejam acessíveis na internet. O *deploy* da aplicação foi realizado na plataforma de serviços de computação em nuvem *Amazon Web Services* (AWS). Os recursos utilizados foram Amazon RDS para o banco de dados PostgreSQL e uma instância Amazon EC2 para executar os servidores. Por questões econômicas, apenas uma instância EC2 foi utilizada, mas o ideal é que os servidores fiquem em máquinas diferentes. A Figura 15 apresenta a arquitetura da aplicação na nuvem.

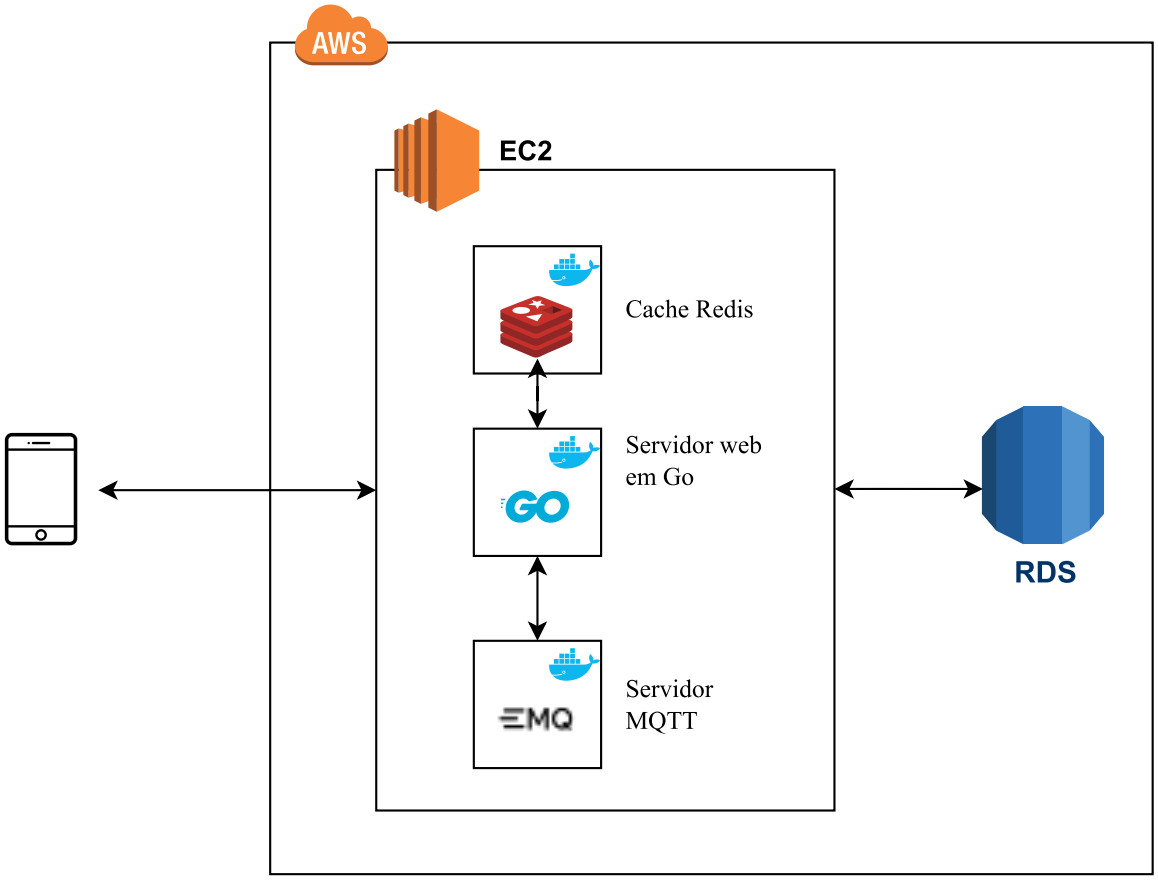
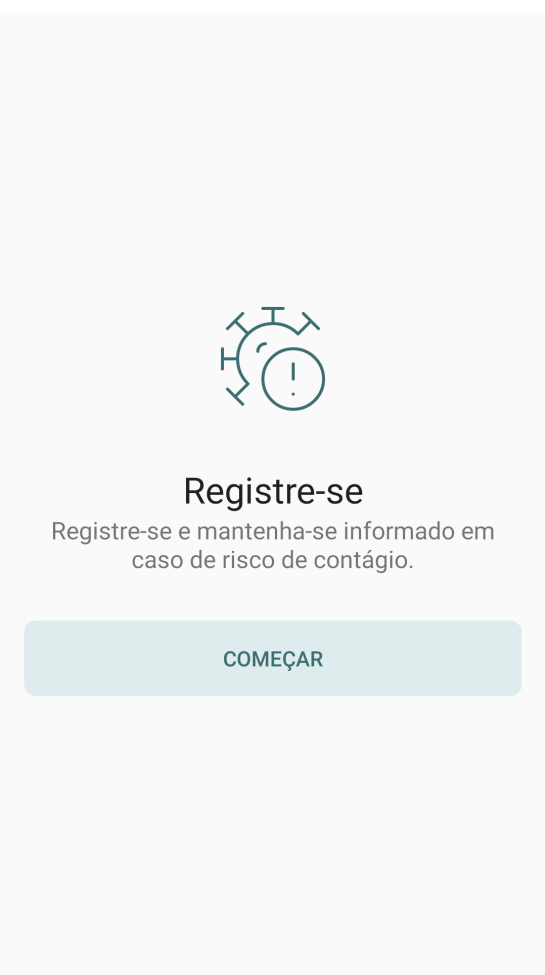


Figura 15 – Arquitetura da solução implantada na nuvem AWS. Elaborado pelo autor (2022)

O servidor *web*, a cache Redis e servidor MQTT foram virtualizados com *containers Docker*. A imagem do servidor *web* foi criada e salva no *Dockerhub* para que pudesse ser utilizada dentro instância EC2 e facilmente atualizada.

3.7.2 Aplicativo móvel

O aplicativo foi desenvolvido com a linguagem Java e um arquivo APK foi gerado para os usuários instalarem. As Subfiguras 16(a) e 16(b) apresentam as telas de registro de usuário e de rastreamento desativado, respectivamente, a qual é mostrada logo após se registrar e também nas outras vezes em que o usuário abrir o aplicativo sem ter ativado o rastreamento de contatos.

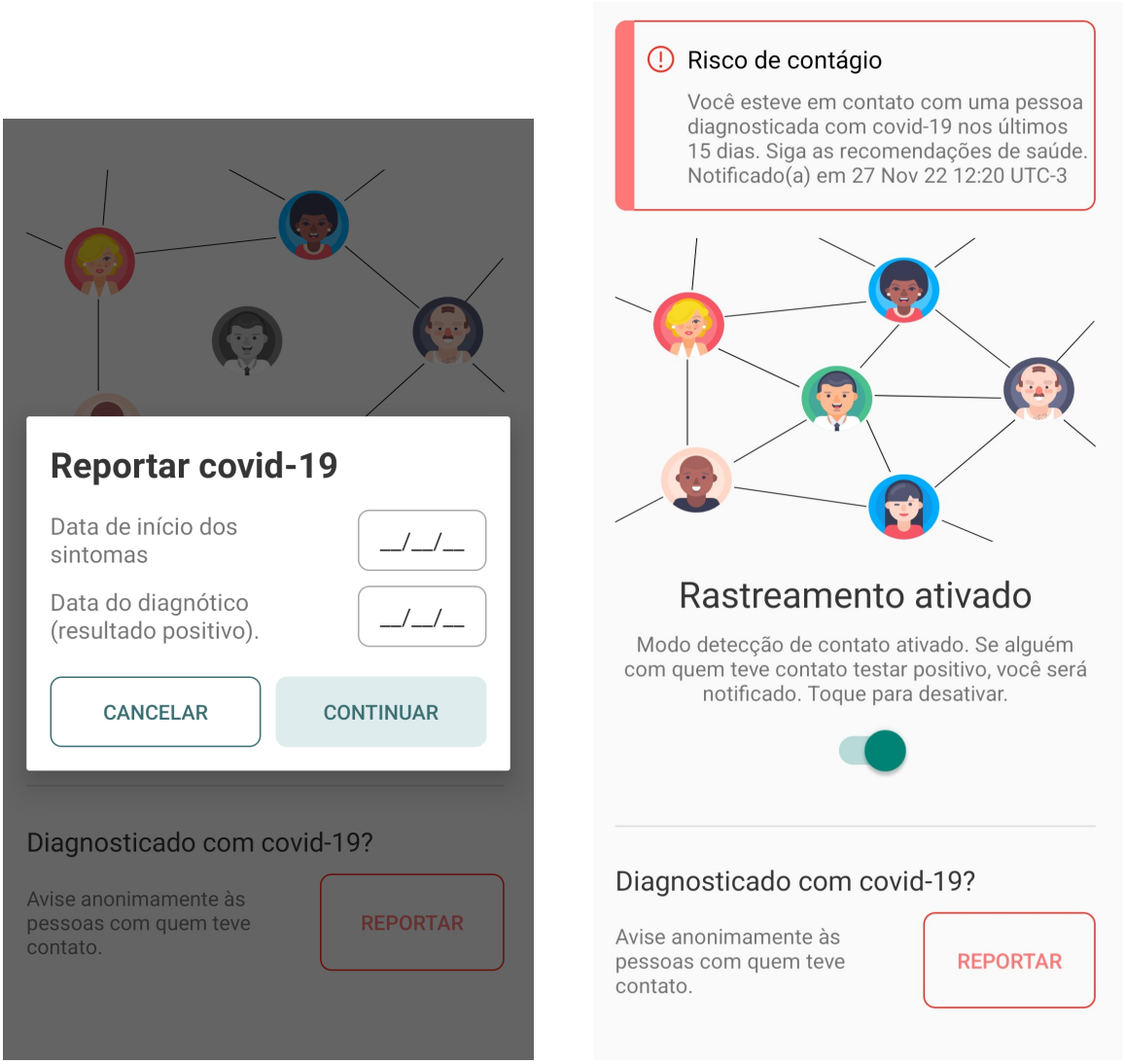


(a) Registro.

(b) Rastreamento desativado.

Figura 16 – Telas de registrar usuário e rastreamento desativado. Elaborado pelo autor (2022)

Na tela de rastreamento desativado é possível ver o botão de reportar covid-19. A Subfigura 17(a) exibe o que é solicitado do usuário ao clicar nesse botão: a data de início dos sintomas e a data em que recebeu o diagnóstico positivo. A Subfigura 17(b) mostra a tela de rastreamento ativado onde o usuário foi notificado de risco de contágio.



(a) Reportar covid-19.

(b) Rastreamento ativado com notificação.

Figura 17 – Telas de reportar covid-19 e rastreamento ativado. Elaborado pelo autor (2022)

4 Análise dos dados coletados

Este capítulo apresenta a análise dos dados coletados pelo sistema proposto durante o teste realizado. A Seção 4.1 descreve a metodologia utilizada para a realização do teste. A Seção 4.2 apresenta os resultados das análises dos dados coletados. A Seção 4.3 apresenta as considerações finais do capítulo.

4.1 Metodologia

4.1.1 Metodologia da coleta de dados

Um formulário foi criado explicando o propósito e funcionamento do sistema e esclarecendo quais dados são coletados durante o uso do aplicativo. Nele, os voluntários devem concordar com a coleta dos dados informados para que o aplicativo seja disponibilizado para *download*. Após concordar, é fornecido um *link* para o Google Drive, onde o usuário pode baixar e instalar o aplicativo, no formato APK, em seu dispositivo. Foi orientado que, por não ser um aplicativo registrado na Play Store, é normal que o Google Drive e/ou o sistema operacional Android apresente um alerta ao tentar baixá-lo, no entanto, não é motivo para preocupação. Além disso, o formulário pede que os voluntários informem um e-mail para contato, que não é associado ao sistema, mas é utilizado somente em uma etapa do teste. É importante dizer que um e-mail para reportar problemas e/ou tirar dúvidas foi disponibilizado pela autora do projeto.

O formulário foi divulgado entre alguns alunos da Universidade Federal do Espírito Santo e entre amigos e parentes da autora deste projeto. Os usuários foram instruídos a utilizar o sistema por algumas horas diariamente durante 2 semanas, destacando a importância de utilizá-lo no momento em que outras pessoas próximas também estiverem utilizando-o. Ao final das 2 semanas, dois voluntários foram sorteados para reportar um diagnóstico positivo. Essa seleção foi feita a partir dos e-mails fornecidos no preenchimento do formulário. Dado que os diagnósticos positivos não foram reais, as notificações não indicaram um risco de verdade.

4.1.2 Visualização e análise dos dados

Os dados coletados podem ser visualizados e analisados a partir de métricas. A seguir, são apresentadas algumas técnicas de visualização utilizadas e métricas para a análise dos dados coletados pelo sistema proposto no teste realizado.

4.1.2.1 Grafo de contatos

Contatos entre pessoas podem ser representados como uma rede e uma rede pode ser representada como um grafo. Nesse grafo, um nó indica uma pessoa e uma aresta um contato entre duas pessoas. Quando um nó da rede se infecta, a transmissão pode acontecer através dessas arestas. Um nó com muitas arestas tem mais chances de se contaminar. No entanto, esse não deve ser o único ponto a se considerar. Arestas de um grafo podem ter pesos e, no caso do grafo de contatos, o peso pode ser a duração do contato e/ou a distância do contato, como também a quantidade de vezes que aquela pessoa se aproximou da outra, o que também influencia na análise do risco de contágio.

É possível destacar nós com alguma característica específica. No caso do experimento realizado, pode-se destacar os nós infectados e também aqueles que se encontram em risco e devem ser ou foram notificados. O grafo de contatos ainda permite visualizar o comportamento social dos usuários e realizar análises mais profundas sobre a dispersão do vírus entre as pessoas.

4.1.2.2 Gráfico de dispersão

Um gráfico de dispersão é uma representação gráfica da associação entre pares de dados, onde cada par se torna um ponto no gráfico (FM2S, 2020). No contexto do rastreamento de contatos, podem ser avaliadas diversas características em pares, uma delas é a duração e a distância média de um contato. Quando se consideram todos os contatos de um grupo de usuários, é possível analisar o comportamento social daquele grupo pelos tipos de contatos que o sistema conseguiu coletar. Por exemplo, é possível visualizar se aquele grupo tem contatos mais rápidos, porém mais próximos ou o contrário. Agora, quando se consideram apenas os contatos entre dois indivíduos, também é possível visualizar o risco de contágio entre eles pela quantidade de pontos que se encontram na área do gráfico que representa contatos mais próximos e mais longos.

4.1.2.3 Função de Distribuição Acumulada

A Função de Distribuição Acumulada (FDA) calcula a probabilidade acumulada para um determinado valor, ou seja, a partir dela é possível determinar a probabilidade de que um dado retirado da amostra seja menor ou igual um determinado valor (Minitab, 2021). Assim como o gráfico de dispersão, é possível visualizar o comportamento dos dados coletados pelo sistema referente a um grupo.

É interessante utilizar a FDA para dados de contato, como duração e distância média, porque são informações essenciais para a identificação de risco de contágio da covid-19. No entanto, outra informação que também pode ser relevante é o grau dos nós da rede, obtido através do grafo de contato. A FDA do grau dos nós nos traz a probabilidade

de um indivíduo da amostra ter tido contato com até `n` outras pessoas, o que tem bastante influência no espalhamento do vírus.

4.2 Resultados

Ao todo, 20 usuários foram registrados no sistema e 866 dados de contatos foram coletados entre os dias 06/11/2022 e 20/11/2022. O sorteio dos voluntários para diagnosticar covid aconteceu no dia 27/11/2022, onde 2 (dois) diagnósticos positivos foram reportados e 3 (três) notificações de risco de contágio foram geradas. Apesar de 20 usuários terem se registrado no sistema, os dados de contatos se referem a 17 usuários, o que pode ter acontecido devido à má utilização do aplicativo, ou seja, nem todas as pessoas ativaram o rastreamento próximo à alguém que também estava com o rastreamento ativado ou simplesmente algumas pessoas não ativaram o rastreamento em momento algum.

Os dados dos contatos foram agregados considerando um contato constante, aquele em que a diferença de um registro de contato e outro foi menor que 15 minutos, similar a como foi feito no servidor, descrito na Seção 3.4. Portanto, doravante, um contato se refere a um conjunto de registros de contatos que foram agregados, os quais são dados enviados pelos dispositivos ao encontrar alguém próximo. A Tabela 1 apresenta um resumo dos dados coletados.

Tabela 1 – Resumo dos dados coletados pelo sistema.

Usuários registrados	20
Usuários com contatos	17
Registros de contatos	866
Contatos constantes	341

A Figura 18 mostra a quantidade de registros de contatos enviados ao servidor por dia ao longo das 2 semanas de teste. Veja que houve dias em que nenhum contato foi enviado, indicando que, provavelmente, os usuários não se lembraram de ativar o rastreamento por algumas horas, como recomendado. É possível perceber que o dia 13/11/2022 foi o que teve mais contatos reportados.

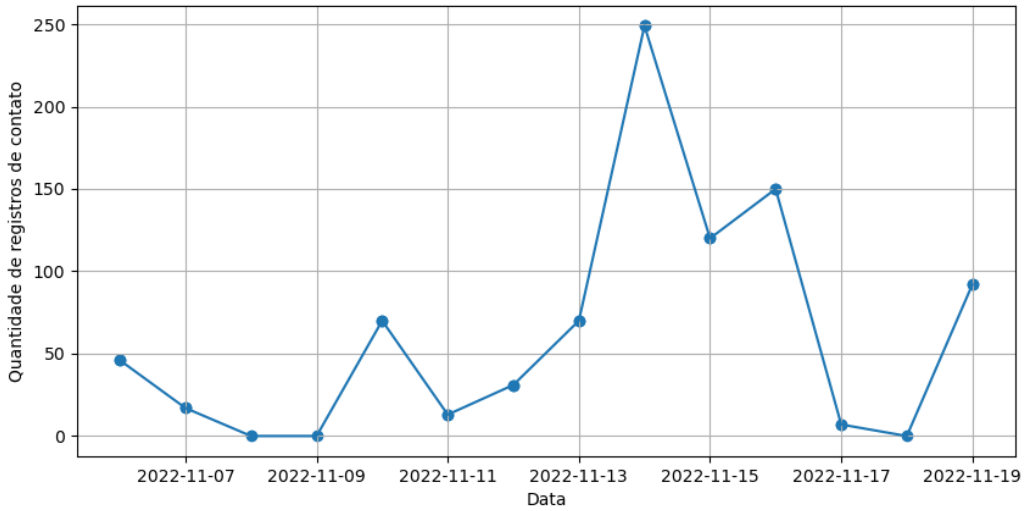


Figura 18 – Quantidade de registros de contato por dia. Elaborado pelo autor (2022)

Além disso, a partir dos contatos constantes, foi possível obter as métricas das distâncias, duração e também da quantidade de contatos constantes por usuário, apresentadas na Tabela 2.

Tabela 2 – Caracterização dos dados de contatos

Dado	**Média**	**Desvio Padrão**	**Mínimo**	**Máximo**
Distância (cm)	152,918	145,75	0,0	713.0
Duração (min)	5,085	7,933	0,0	55,203
Contatos constantes por usuário	5,879	5,127	0	21

É possível observar que a distância média dos contatos coletados foi inferior a 200 cm, ou seja, uma distância considerada de risco, no entanto, o desvio padrão, por apresentar um valor elevado, mostra que os valores de distância variaram bastante em relação à média. A distância mínima igual a zero indica que os usuários estavam tão próximos, que a distância foi desconsiderada. Já em relação à duração dos contatos, a média foi baixa, sendo de apenas cinco minutos, com um desvio relativamente pequeno. No entanto, a grande diferença entre valor mínimo e valor máximo de duração, mostra que o sistema conseguiu capturar tanto contatos curtos, quanto de grande duração. Na verdade, os contatos de duração igual a zero poderiam ser até desconsiderados, uma vez que os usuários não permaneceram tempo suficiente próximos para receberem mais de um sinal de *beacon* do outro. Por fim, a última linha da Tabela 2 diz respeito à quantidade de contatos constantes que foi possível extrair dos dados enviados ao servidor. A média foi de quase seis, entretanto isso inclui até aqueles contatos com duração igual a zero. Usuários que fizeram *download* do aplicativo e se registraram, mas não ficaram próximas de pessoas que também utilizavam-no, não conseguiram enviar nenhum contato para o

servidor. A Figura 19 mostra o grafo de contatos geral, onde é possível ver as interações dos voluntários entre si ao longo das duas semanas.

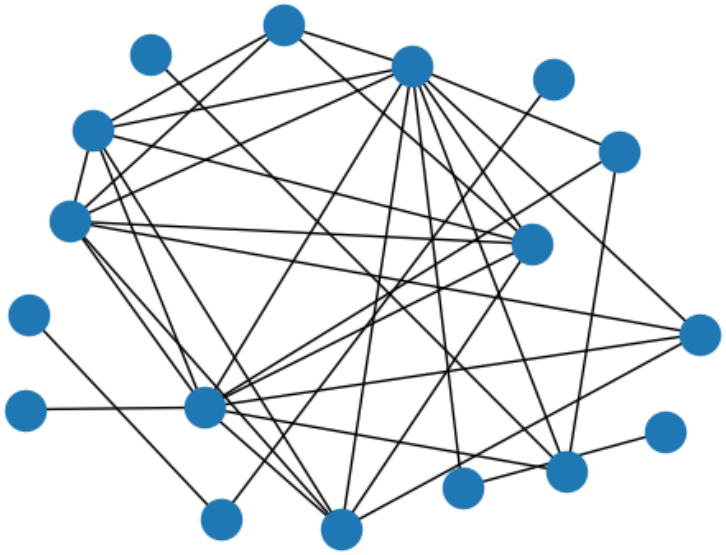
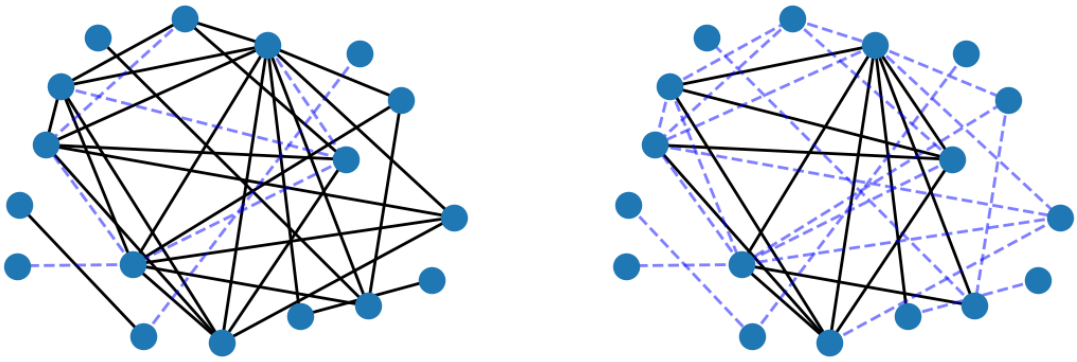


Figura 19 – Grafo de contatos geral. Elaborado pelo autor (2022)

Agora, a Figura 20 apresenta os grafos de contatos considerando as arestas com peso de distância e duração de contato. Dois indivíduos podem ter mais de um contato constante durante um intervalo de tempo, portanto, para gerar os grafos foi considerada a média das distâncias e a maior duração dos contatos entre dois indivíduos. No caso da Subfigura 20(a), uma aresta tracejada na cor azul indica que o contato entre os dois indivíduos teve distância média superior a 200 cm, enquanto as arestas pretas indicam contatos com distância média inferior a 200 cm. Da mesma forma, para a Subfigura 20(b), onde as arestas tracejadas indicam contatos com duração menor que 15 minutos e as arestas na cor preta indicam contatos com duração maior que 15 minutos.



(a) Distância média. (b) Duração máxima.

Figura 20 – Grafos de contato ponderados por distância e duração. Elaborado pelo autor (2022)

O risco de fato aparece quando as duas variáveis, duração e distância são avaliadas, desse modo, a Figura 21 mostra o grafo de contato em que as arestas pretas indicam contatos em que a duração foi maior ou igual a 15 minutos e a distância média foi menor que 2 metros. Sendo assim, é possível visualizar os indivíduos que podem ser notificados caso um deles reporte um diagnóstico positivo.

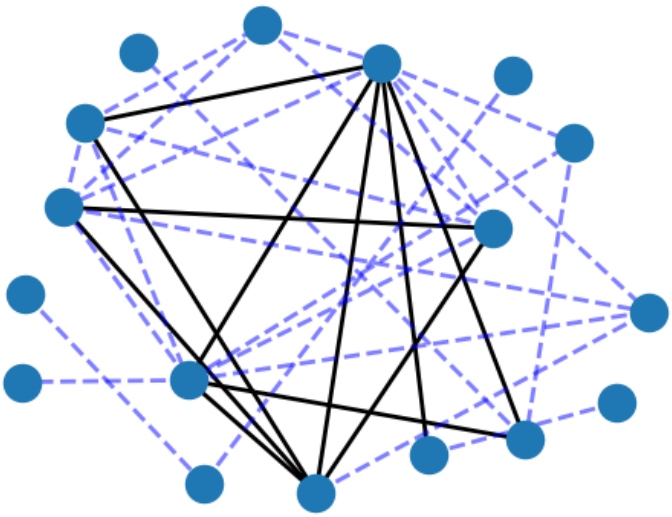


Figura 21 – Grafo de contatos ponderado por distância e duração simultaneamente. Elaborado pelo autor (2022)

Como citado anteriormente, dois voluntários foram selecionados para reportar um diagnóstico positivo. Então, a partir do grafo anterior, se marcarmos os nós infectados de vermelho e os nós notificados de amarelo, obtemos o grafo da Figura 22. Veja que, os nós notificados pelo servidor foram realmente os nós que tiveram contatos considerados de risco com o infectado nos últimos 15 dias.

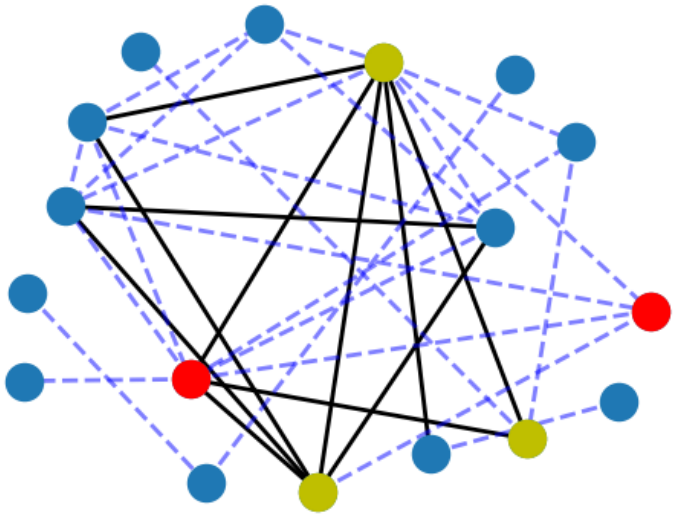


Figura 22 – Grafo de contatos com nós infectados e notificados. Elaborado pelo autor (2022)

Também é importante analisar os tipos de contatos coletados durante o experimento. A Figura 23 mostra o grafo de dispersão dos contatos de todos os usuários. Observe que a maioria dos pontos se encontra próxima à origem, indicando contatos de curta duração e também de curta distância. Há contatos em que a distância foi muito superior a 200 cm, ou 2 metros, podendo indicar que os indivíduos estavam em cômodos diferentes ou simplesmente bem distantes, o que não expressa um possível risco de contágio no caso de um usuário estar infectado.

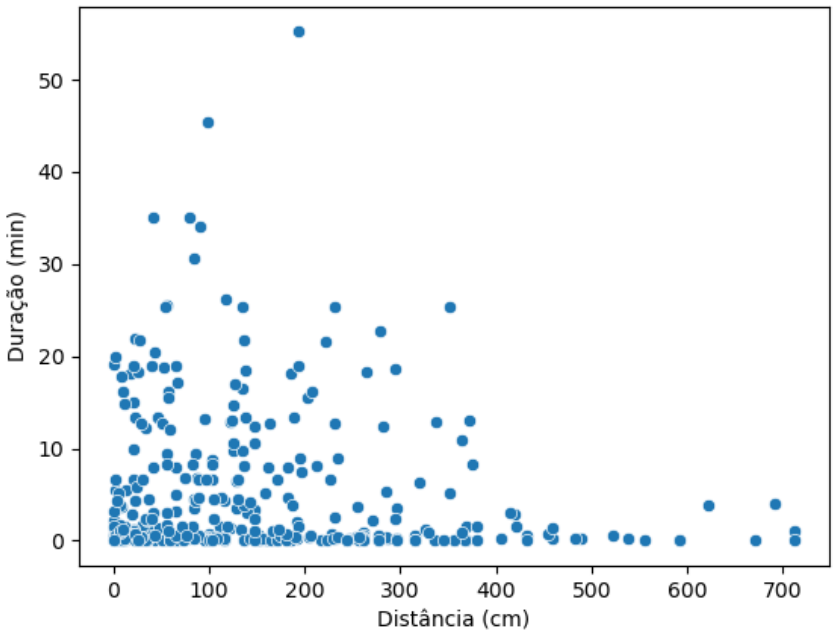


Figura 23 – Gráfico de dispersão dos contatos de todos os usuários. Elaborado pelo autor (2022)

O contato de maior duração possui duração superior a 50 minutos e uma distância por volta de 200 cm. É possível observar diversos contatos com duração igual a zero, o que indica que os usuários receberam apenas um único *beacon* no dispositivo vindo de outro e demoraram pelo menos cerca de 15 minutos para receberem outro daquele mesmo usuário ou simplesmente não receberam mais.

Outra forma de analisar os contatos coletados é observando as funções de distribuição acumulada de um tipo de informação. As Figuras 24 e 25 e 26 apresentam o gráfico função de distribuição acumulada da distância média, da duração e do grau dos nós do grafo de contato, respectivamente.

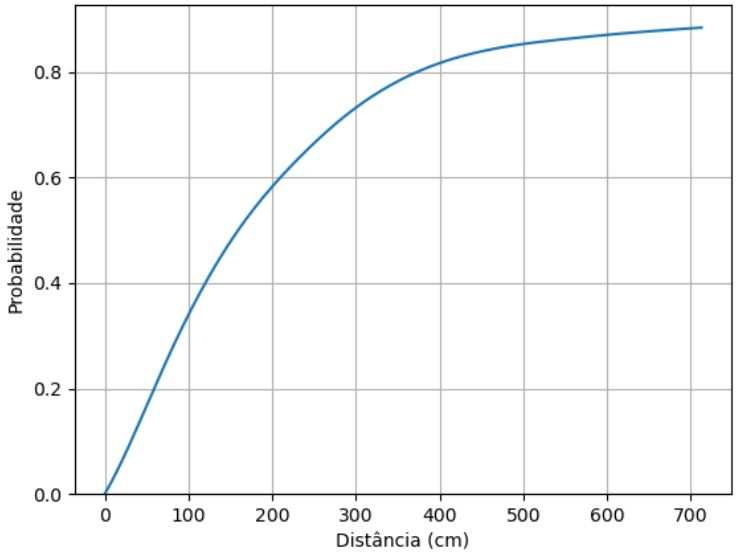


Figura 24 – FDA das distâncias dos contatos. Elaborado pelo autor (2022)

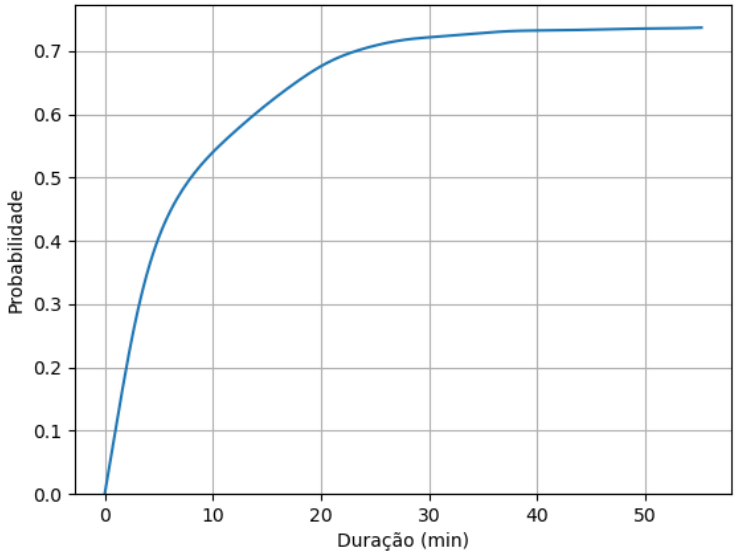


Figura 25 – FDA da duração dos contatos. Elaborado pelo autor (2022)

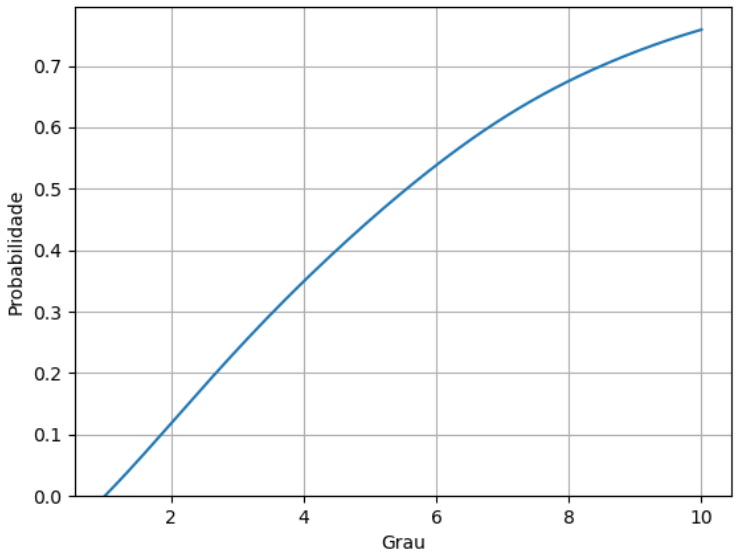


Figura 26 – FDA dos graus dos nós do gráfico de contatos. Elaborado pelo autor (2022)

A Figura 24 mostra que por volta de 60% dos contatos registrados aconteceram em uma distância média de até 2 metros, enquanto apenas um pouco mais de 30% foram inferior a 1 metro de distância. Já a Figura 25 mostra que cerca de 72% dos contatos duraram até 30 minutos. Além disso, pelo gráfico de dispersão da Figura 23, nota-se que grande parte dos contatos durou até 10 minutos, o que é evidenciado na FDA exibida em 25, onde é possível extrair que 54% dos contatos duraram até o tempo mencionado de 10 minutos. Por fim, a Figura 26 diz respeito à probabilidade de um voluntário ter tido contato com uma certa quantidade de pessoas, já que mostra a FDA dos graus dos nós do grafo de contato. Sendo assim, percebe-se que por volta de 67% dos nós do grafo possui até 8 arestas, ou seja, contato com até 8 pessoas distintas.

4.3 Considerações

A realização do experimento permitiu, de certa forma, validar o funcionamento do sistema proposto, uma vez que os voluntários que reportaram um diagnóstico positivo tiveram seus contatos de risco identificados e os usuários notificados, seguindo o que se esperava pelos tipos de contatos coletados pelo sistema. Além disso, permitiu analisar os dados coletados, onde foi possível identificar contatos com potencial de espalhamento do vírus e observar o comportamento social dos usuários, dentro das limitações do experimento, sem revelar informações pessoais dos usuários que participaram.

De fato, é importante destacar que as limitações influenciam no resultado das análises dos dados coletados. A primeira questão a se pontuar é o fato do sinal *Bluetooth* capturado pelos dispositivos poder ser influenciado por vários fatores além da distância,

como objetos no meio do caminho e paredes que dividem cômodos. Além de que, outros sinais sem fio usando a mesma frequência também podem causar interferência e atenuar ainda mais o sinal. Por fim, também há o fato de diferentes dispositivos transmitirem sinais *Bluetooth* com diferentes níveis de potência, sendo ideal realizar uma calibração, o que não foi feito para o experimento.

Outra questão é que os usuários foram instruídos a utilizarem o aplicativo preferencialmente quando outros usuários também estiverem utilizando, mas era esperado que nem sempre isso seria possível e por isso, mais contatos podem ter acontecido e não foram coletados pelo sistema, além da dificuldade dos voluntários lembrarem de ativar o rastreamento por algumas horas no dia. Em suma, conseguir pessoas para participarem do teste também foi uma das dificuldades que surgiram, principalmente motivá-las a utilizarem e garantir que o aplicativo é confiável, uma vez que um alerta aparece ao tentar instalá-lo, por não estar registrado na Play Store e, portanto, o sistema não reconhecer o desenvolvedor.

5 Conclusão

Neste trabalho, foi proposto e desenvolvido um sistema de rastreamento de contatos anônimo, que teve como principal objetivo preservar a privacidade dos usuários para que se sentissem seguros em utilizá-lo. O trabalho fez uma análise das arquiteturas existentes e os problemas de privacidade que cada uma tem. Além disso, também foi levantada a questão da necessidade de poder analisar dados de contatos de usuários para estudar sobre o espalhamento do vírus, sem expor os usuários. Ao fim, voluntários foram convidados a utilizar o sistema por 2 semanas e os dados coletados foram analisados.

A Seção 5.1 apresenta as considerações finais do trabalho, limitações, contribuições e experiência adquirida. A Seção 5.2 traz sugestões de trabalhos futuros.

5.1 Considerações Finais

O sistema cumpre com o objetivo base de rastrear os contatos de usuários infectados e notificá-los, uma vez que todos os usuários que tiveram contatos considerados de risco com um infectado foram notificados no experimento realizado e detalhado no Capítulo 4. Também é importante destacar a anonimidade alcançada, os usuários notificados não receberam nenhuma informação que pudesse identificar quem era o infectado e como o servidor não coletou nenhuma informação pessoal, não era possível associar somente os dados que estavam armazenados no banco de dados a uma pessoa em específico, portanto, o conceito de anônimo foi atingindo nesse contexto. Além disso, foi possível analisar os tipos de contatos coletados e o potencial risco de contágio, sem que qualquer usuário fosse propriamente identificado pelo servidor.

Quanto ao objetivo de motivar as pessoas a utilizá-lo, a dificuldade foi em instruir as pessoas de que o aviso de perigo que aparece no dispositivo ao tentar instalá-lo deve-se ao fato do aplicativo não estar registrado na Play Store, portanto, o desenvolvedor é desconhecido pelo sistema e com isso, é esperado que uma notificação apareça. Com a publicação do aplicativo, esse problema seria resolvido, no entanto, devido ao tempo que leva para a aplicação ser revisada e liberada para *download*, a publicação do aplicativo na Play Store foi impossibilitada.

Uma limitação que pode ter influenciado no resultado obtido é a precisão da distância estimada pelos dispositivos a partir do sinal de *Bluetooth* recebido. A Seção 4.3 comenta desse desafio na análise dos dados coletados, uma vez que a potência do sinal pode ser influenciada por diversos fatores desconsiderados na implementação do sistema. Outro ponto é que os voluntários, apesar de instruídos a utilizarem o aplicativo diariamente por

algumas horas e, preferencialmente, próximo a outras pessoas que também estivessem utilizando, isso pode não ter sido seguido, já que alguns dias foram registrados zero contatos coletados pelo servidor.

Outra questão que surgiu durante a implementação da solução, foi o fato do sistema operacional Android finalizar processos que estavam rodando em *foreground* quando havia necessidade de liberar espaço de memória, sem aviso, o que podia prejudicar o serviço de rastreamento, que deveria continuar ativo. No entanto, como uma notificação era mantida informando que o rastreamento estava ativo, caso o sistema operacional removesse o processo, o usuário poderia notar que o serviço não estava mais em execução e assim, ativar novamente, mas nem sempre isso acontece.

Por fim, a não validação do diagnóstico positivo pode ocasionar falsos positivos registrados no sistema e, com isso, pessoas notificadas erroneamente. Isso já era esperado pelo que foi proposto pelo sistema. Durante a realização do teste, apresentado no Capítulo 4, apenas os usuários selecionados reportaram covid-19. No entanto, em uma implementação real, esse problema poderia facilmente acontecer.

Do ponto de vista acadêmico, este trabalho permitiu reforçar conceitos aprendidos ao longo do curso, como programação, lógica e sistemas distribuídos, atrelando conceitos teóricos à prática.

5.2 Trabalhos Futuros

A limitação a respeito da imprecisão da estimativa de distância a partir da intensidade de sinal abre uma oportunidade de melhoria. Uma distância mal estimada pode ocasionar contatos de risco não identificados e com isso, usuários não notificados quando deveriam ou até mesmo o contrário, notificações para usuários que não tiveram contatos de risco. A respeito das diferentes potências de sinal Bluetooth emitidas por diferentes dispositivos, é possível configurar a força de transmissão de sinal *Bluetooth* na biblioteca utilizada *Android Beacon Library*, permitindo fornecer um valor ideal para tipos de dispositivos, o que requer uma pesquisa e análise sobre os melhores valores para cada modelo de *smartphone*, linha ou fabricante.

Na fase de teste, o formulário enviado às pessoas para selecionar voluntários, citado no Capítulo 4, poderia questionar o que desestimulou utilizar o aplicativo, caso a pessoa marcasse que não concorda com as coleta dos dados informados, permitindo avaliar outras questões que deixaram as pessoas inseguras para poder melhorar a implementação do sistema ou a apresentação dele.

Uma funcionalidade que agregaria na análise do espalhamento e comportamento do vírus é o acompanhamento dos sintomas do infectado. Um trabalho futuro envolveria

adicionar a funcionalidade de enviar sintomas diariamente no aplicativo móvel. Além disso, seria importante buscar uma forma de validar o diagnóstico positivo do usuário, para evitar falsos positivos e notificações desnecessárias. Como a proposta do sistema é não coletar nenhuma informação pessoal, o mais adequado seria utilizar uma validação por terceiros, como uma entidade de saúde, de forma que não revele os dados pessoais do infectado, apenas se o diagnóstico é válido ou não.

Com relação à análise dos dados coletados, um trabalho futuro abrange construir um modelo de predição para identificar possíveis usuários infectados a partir do comportamento social deles, ou seja, a partir dos contatos enviados ao servidor. Como também identificar grupos com maior potencial de espalhamento do vírus.

Outra oportunidade seria generalizar o sistema para qualquer doença similar na forma de transmissão, parametrizando dados que definem um risco de contágio. Dessa forma, o sistema poderia ser utilizado para rastrear contatos de outras doenças, como a influenza. Além disso, ainda é possível adicionar a funcionalidade de rastreamento de contágio, buscando identificar a origem de uma ou mais transmissões e com isso, analisar mais profundamente o comportamento de um vírus, além de permitir notificar pessoas infectadas que estão contribuindo para o aumento da taxa de transmissão.

Referências

AGRAWAL, M.; MISHRA, P. A comparative survey on symmetric key encryption techniques. *International Journal on Computer Science and Engineering*, Citeseer, v. 4, n. 5, p. 877, 2012. Citado na página 21.

AHMED, N. et al. A survey of covid-19 contact tracing apps. *IEEE access*, IEEE, v. 8, p. 134577–134601, 2020. Citado 5 vezes nas páginas 12, 16, 18, 19 e 27.

ALTBEACON.ORG. *AltBeacon Protocol Specification v1.0*. 2014. Disponível online em <https://github.com/AltBeacon/spec>, Último acesso em 09/11/2022. Citado 2 vezes nas páginas 6 e 20.

AN, J. *How Does Golang Channel Works*. 2021. Disponível online em <https://levelup.gitconnected.com/how-does-golang-channel-works-6d66acd54753>, Último acesso em 12/11/2022. Citado 2 vezes nas páginas 6 e 24.

BAY, J. et al. Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders. *Government Technology Agency-Singapore, Tech. Rep*, v. 18, 2020. Citado na página 27.

BIEHL, I.; MEYER, B.; MÜLLER, V. Differential fault attacks on elliptic curve cryptosystems. In: SPRINGER. *Annual International Cryptology Conference*. [S.l.], 2000. p. 131–146. Citado na página 22.

BORGES, F. G.; MOREIRA, M. A. R.; FERREIRA, R. M. Ecdsa (elliptic curve digital signature algorithm). In: *6º CONTECSI-International Conference on Information Systems and Technology Management*. [S.l.: s.n.], 2015. Citado na página 22.

CASTELLUCCIA, C. et al. Robert: Robust and privacy-preserving proximity tracing. 2020. Citado na página 27.

EMQX. *Design*. 2022. Disponível online em <https://www.emqx.io/docs/en/v4.4/design/design.html#foreword>, Último acesso em 13/11/2022. Citado na página 26.

FM2S. *O que é e para que serve o Gráfico de Dispersão?* 2020. Disponível online em <https://www.fm2s.com.br/blog/grafico-de-dispersao>, Último acesso em 06/02/2023. Citado na página 47.

GAITHURU, J. N. et al. A comprehensive literature review of asymmetric key cryptography algorithms for establishment of the existing gap. In: IEEE. *2015 9th Malaysian Software Engineering Conference (MySEC)*. [S.l.], 2015. p. 236–244. Citado 2 vezes nas páginas 21 e 22.

GUPTA, N. K. *Inside Bluetooth low energy*. [S.l.]: Artech House, 2016. Citado 2 vezes nas páginas 19 e 20.

HEYDON, R.; HUNN, N. Bluetooth low energy. *CSR Presentation, Bluetooth SIG*, 2012. Citado na página 19.

HiveMQ. *HiveMQ MQTT Broker - Enterprise ready MQTT broker to move IoT data*. Disponível online em <https://www.hivemq.com/hivemq/mqtt-broker/>, Último acesso em 06/02/2023. Citado na página 26.

KANG, S.-J. et al. Successful control of covid-19 outbreak through tracing, testing, and isolation: Lessons learned from the outbreak control efforts made in a metropolitan city of south korea. *Journal of Infection and Public Health*, Elsevier, v. 14, n. 9, p. 1151–1154, 2021. Citado na página 12.

LEE, E. et al. Locally testable privacy-preserving contact tracing protocol without exposing secret seed. In: *2021 IEEE International Conference on Consumer Electronics (ICCE)*. [S.l.: s.n.], 2021. p. 1–5. Citado 2 vezes nas páginas 27 e 28.

MAZZER, D.; FRIGIERI, E.; PARREIRA, L. Protocolos m2m para ambientes limitados no contexto do iot: Uma comparação de abordagens. *Inatel. Br*, p. 24, 2015. Citado 2 vezes nas páginas 24 e 25.

Minitab. *Uso da função de distribuição acumulada (FDA)*. 2021. Disponível online em <https://support.minitab.com/pt-br/minitab/20/help-and-how-to/probability-distributions-random-data-and-resampling-analyses/supporting-topics/basics/using-the-cumulative-distribution-function-cdf>, Último acesso em 06/02/2023. Citado na página 47.

Mosquitto. *Eclipse Mosquitto - An open source MQTT broker*. Disponível online em <https://mosquitto.org/>, Último acesso em 06/02/2023. Citado na página 26.

NAKOV, S. *Practical Cryptography for Developers*. 2018. Disponível online em <https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages>, Último acesso em 09/11/2022. Citado na página 23.

National Human Rights Commission of Korea. *NHRCK Chairperson's Statement on Excessive Disclosure of Private Information of COVID-19 Patients*. 2020. Disponível online em <https://www.humanrights.go.kr/site/program/board/basicboard/view?boardtypeid=7003&boardid=7605315&menuid=002002001>, Último acesso em 02/06/2022. Citado na página 12.

O'HARA, J. et al. *Advanced Message Queuing Protocol–Protocol Specification*. 2006. Citado 2 vezes nas páginas 6 e 26.

PORTNOI, M. Criptografia com curvas elípticas. Núcleo de Pesquisa Interdepartamental em redes de computadores, Universidade de Salvador – UNIFACS, p. 14, 2005. Citado 2 vezes nas páginas 6 e 22.

RIVEST, R. L. et al. *The PACT protocol specification*. [S.l.], 2020. v. 0.1. Citado na página 17.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, ACM New York, NY, USA, v. 21, n. 2, p. 120–126, 1978. Citado na página 21.

SONI, D.; MAKWANA, A. A survey on mqtt: a protocol of internet of things (iot). In: *International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017)*. [S.l.: s.n.], 2017. v. 20, p. 173–177. Citado na página 25.

TEMPORIN, T. *O que são e como funcionam as Goroutines*. 2021. Disponível online em <https://aprendagolang.com.br/2021/11/19/o-que-sao-e-como-funcionam-as-goroutines>, Último acesso em 12/11/2022. Citado na página 24.

WIKIPEDIA. *Go (linguagem de programação) — Wikipédia, a enciclopédia livre*. 2022. Disponível online em <https://pt.wikipedia.org/wiki/Go_(linguagem_de_programaÃğÃčo)>, Último acesso em 06/02/2023. Citado na página 24.

World Health Organization. *Timeline: WHO's COVID-19 response*. Disponível online em <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/interactive-timeline>, Último acesso em 02/06/2022. Citado na página 12.

World Health Organization. *Contact tracing in the context of COVID-19*: Interim guidance. 2020. Disponível online em <https://apps.who.int/iris/bitstream/handle/10665/332049/WHO-2019-nCoV-Contact_Tracing-2020.1-eng.pdf>, Último acesso em 31/05/2022. Citado 2 vezes nas páginas 12 e 16.