UNIVERSIDADE FEDERAL DO PARÁ

CAMPUS UNIVERSITÁRIO DE CASTANHAL

FACULDADE DE COMPUTAÇÃO

CLEITON EWERTON QUARESMA DOS SANTOS

PLATAFORMA PARA ARMAZENAMENTO E GERENCIAMENTO DE INFORMAÇÕES APLICANDO IOT (*INTERNET OF THINGS*)

Estudo de caso voltado para a medicina veterinária.

Castanhal - PA

2024

CLEITON EWERTON QUARESMA DOS SANTOS

PLATAFORMA PARA ARMAZENAMENTO E GERENCIAMENTO DE INFORMAÇÕES APLICANDO IOT (*INTERNET OF THINGS*): Estudo de caso voltado para a medicina veterinária.

Trabalho de Conclusão de Curso apresentado à Universidade Federal do Pará, como requisito parcial para a obtenção do grau de Bacharelado em Engenharia de Computação, pela Universidade Federal do Pará.

Orientador: Prof. Dr. Bruno Souza Lyra Castro.

Castanhal - PA

2024

CLEITON EWERTON QUARESMA DOS SANTOS

Plataforma para armazenamento e gerenciamento de informações aplicando Iot (*Internet Of Things*): Estudo de caso voltado para a medicina veterinária.

Este trabalho foi julgado adequado em 08 de Novembro de 2024 para obtenção do Grau de ENGENHEIRA DE COMPUTAÇÃO de CLEITON EWERTON QUARESMA DOS SANTOS, aprovado em sua forma final pela banca examinadora, a qual atribuiu o seguinte conceito: EXCELENTE.

Banca examinadora:

_____

Prof. Dr. Bruno Souza Lyra Castro.
Orientador – UFPA

_____

Prof. Dr. Igor Ruiz Gomes.
Examinador(a) Interno – UFPA

_____

Prof. Dr. Tássio Costa de Carvalho.
Examinador(a) Interno – UFPA

AGRADECIMENTOS

Primeiramente, minha eterna gratidão a Deus, que ouviu minhas preces e jamais deixou que me faltasse a força para seguir em frente. Sua presença me deu coragem nos momentos de dúvida e motivação nos dias mais desafiadores.

Aos meus pais, João dos Santos Filho e Santana do Socorro Quaresma Do Santos, agradeço por serem o alicerce de tudo. Com amor incondicional, apoio e sacrifício, vocês me deram tudo que eu precisava e muito mais. Pelo companheirismo, pela confiança, pelos conselhos e pela paciência, sou eternamente grato. Vocês me moldaram com seu exemplo e dedicação, e tudo que conquistei reflete o que aprendi com vocês.

Ao meu orientador Bruno Lyra, meu sincero agradecimento. Obrigado por abrir as portas com tanto entusiasmo e confiança quando decidi abraçar esse projeto. Sua paciência, seus ensinamentos e sua clareza nos momentos de dúvida foram fundamentais para meu crescimento e para o sucesso desta jornada.

Aos amigos que encontrei durante o curso, agradeço por tornarem essa trajetória única, mais leve e repleta de boas memórias. Vocês me apresentaram novos horizontes, me inspiraram e trouxeram risadas em momentos que poderiam ser difíceis. Um agradecimento especial ao Akaz Marinho e ao Ciro Kyushima: as noites de "*call*" no Discord para trabalharmos juntos sempre se transformavam em algo mais que estudo, virando momentos descontraídos, até competições sobre qual de nós vive na cidade que tem o melhor açaí. E aos amigos de longa data, Paulo Guedelha e Poliana Camilo, obrigado por dedicarem seu tempo na elaboração e revisão deste trabalho. Vocês são parte essencial de tudo isso.

À Williane Brasil, secretária do curso de computação, que foi como uma mãe para nossa turma. Sempre presente, lutou ao nosso lado, apresentando soluções rápidas para nossas necessidades, com atenção e cuidado, e acreditando em cada um de nós. A sua dedicação é parte do que nos trouxe até aqui.

Por fim, agradeço profundamente a todos que, de alguma forma, fizeram parte desta caminhada. Sem o apoio, a confiança e a generosidade de cada um, eu jamais teria chegado até aqui.

Muito obrigado a todos, de coração!

EPÍGRAFE

“Nunca tema a escuridão. As árvores mais fortes estão enraizadas nos lugares escuros da Terra.
(George R.R. Martin)

RESUMO

Este trabalho apresenta o desenvolvimento de uma plataforma web capaz de gerar gráficos com base em dados enviados por um dispositivo, o ESP8266. Esse dispositivo coleta dados de temperatura e ECG – a partir dos quais é possível estimar a BPM – de animais de pequeno e grande porte. O sistema foi desenvolvido no Visual Studio Code, um editor de código aberto criado pela Microsoft, utilizando as linguagens CSS, JavaScript, HTML e PHP. As informações de login e dos sinais vitais são armazenadas em um banco de dados MySQL, permitindo o acesso pela aplicação para consultar dados de diferentes períodos. Além disso, o sistema incorpora os conceitos do protocolo MQTT, viabilizando a visualização de gráficos em tempo real, o que auxilia pesquisadores veterinários na análise e diagnóstico de zoonoses. O projeto também conta com um sistema de autenticação via login baseado no método CRUD (Create, Read, Update, Delete), garantindo que apenas usuários autorizados acessem a plataforma e protegendo o servidor contra possíveis ataques de hackers em trabalhos futuros.

Palavras-chave: IoT, MQTT, programação web, segurança da informação.

ABSTRACT

This study presents the development of a web platform capable of generating graphs based on data transmitted by the ESP8266 device. The device collects temperature and ECG data, from which BPM can be estimated, for large and small animals. The system was developed in Visual Studio Code, an open-source editor created by Microsoft, utilizing CSS, JavaScript, HTML, and PHP languages. Login information and vital signs data are stored in a MySQL database and accessible through the application, allowing for data consultation across different historical periods. Additionally, the system integrates MQTT protocol concepts, enabling real-time graph visualization, which assists veterinary researchers in the analysis and diagnosis of zoonotic diseases. The project also includes an authentication system via login based on the CRUD method (Create, Read, Update, Delete), ensuring that only authorized users access the platform and protecting the server from potential hacker attacks in future work.

Keywords: IoT, MQTT, web programming, information security.

LISTA DE FIGURAS

Figura 1 – Esquema de dispositivos IoT conectados à internet. 15

Figura 2 – Fluxo entre clientes e broker. 17

Figura 3 – Menu de Feeds do adafruit IO. 19

Figura 4 – Comparação de conexão WebSockets vs HTTP. 20

Figura 5 – Interface gráfica da tabela de usuários com dados de teste para login. 21

Figura 6 – Esquema geral do projeto proposto por Yew et al (2020). 23

Figura 7 – Implementação do sistema proposto pelos autores Singh e Jasuja (2017). 25

Figura 8 – Tela de login. 32

Figura 9 – Página inicial com barra lateral responsiva. 33

Figura 10 – Tela de cadastro de novo usuário. 34

Figura 11 – Tela de edição de dados do usuário. 35

Figura 12 – Tela de exclusão de usuário logado. 36

Figura 13 – Tela de exibição do painel de gráfico de temperatura. 37

Figura 14 - Segmentos de ondas de ECG. 38

Figura 15 – Página de exibição de temperatura, BPM e ECG em tempo real via MQTT. 39

Figura 16 - Exemplo de ECG em tempo real. Captura de 14 picos de ECG em um intervalo de 5 segundos resultando em uma estimativa de 168 BPM. 40

Figura 17 – Gráficos de temperatura carregados do BD simulando a temperatura corporal de um coelho. 41

Figura 18 – Gráficos de ECG carregados do BD, com informações calculadas de BPM. 42

Figura 19 – Comportamento do navegador no gereciador de tarefas enquanto carrega os dados de ECG. 43

Figura 20 – Comportamento do navegador após a leitura ser finalizada. 43

LISTA DE ABREVIATURAS E SIGLAS

API .....*Application Programming Interface*

BD.....Banco de Dados

BPM .....Batimentos Por Minuto

CRUD .....*Create*, *Read*, *Update* e *Delete*

CSS .....*Cascading Style Sheets*

ECG .....Eletrocardiograma

FIFO .....*First In, First Out*

HTML .....*Hyper Text Markup Language*

HTTP .....*Hypertext Transfer Protocol*

IA .....Inteligência Artificial

ID .....*Identity*

IDE .....*Integrated Developmente Enviroment*

IoT .....*Internet of Things*

JSON .....*JavaScript Object Notation*

LIFO .....*Last In, First Out*

M2M .....Máquina a Máquina

MQTT .....*Message Queuing Telemetry Transport*

MQTT - SN .....*Message Queuing Telemetry Transport - Sensor Network*

ms .....Milissegundo

PHP .....*Hypertext Processor*

PK .....*Primary Key*

QoS .....*Quality of Service*

s .....Segundos

TCP.....*Transmition Control Protocol*

TCC .....Trabalho de Conclusão de Curso

UDP .....*User Datagram Protocol*

UFPA .....Universidade Federal do Pará

VS Code .....*Visual Studio Code*

SUMÁRIO

1. INTRODUÇÃO 12

1.1. Contextualização e Motivação 12

1.2. Objetivos Gerais 13

1.3. Objetivos Específicos 13

1.4. Relevância para a Comunidade Científica 14

2.1. Internet das Coisas (IoT) 14

2.2. Linguagens de programação 15

2.2.1. CSS (Cascading Style Sheets) 16

2.2.2. HTML (Hyper Text Markup Language) 16

2.2.3. Javascript (JS) 16

2.2.4. PHP (Hypertext Processor) 16

2.3. Projeto Paho 16

2.4. Protocolo MQTT (Message Queuing Telemetry Transport) 17

2.4.1. Broker 17

2.4.2. Cliente 18

2.4.3. Qualidade do serviço (QoS) 18

2.5. Plataforma em Nuvem – Adafruit IO 19

2.6. Tecnologia WebSockets 20

2.7. XAMPP 21

3. TRABALHOS CORRELATOS 22

3.1. IoT Based Real-Time Remote Patient Monitoring System 22

3.2. An IoT-Cloud Based Health Monitoring Wearable Device For Covid Patients 23

3.3. IoT based low-cost distant patient ECG monitoring system 25

3.4. Comparações 26

4. DESENVOLVIMENTO 27

4.1. Testes iniciais 27

4.2. Versão 1 28

4.3. Versão 2 28

4.4. Versão 3 29

4.5. Versão 4 29

4.6. Versão 5 30

4.7. Versão 6 30

4.8. Versão 7 31

4.9. Versão 8 36

4.9.1. Consulta de dados armazenados no BD (Temperatura e ECG) 37

4.9.2. Dados em tempo real via MQTT 39

5. RESULTADOS E DISCUSSÕES 41

6. CONCLUSÃO 45

REFERÊNCIAS 47

APÊNDICES 51

1. INTRODUÇÃO

1.1. Contextualização e Motivação

A saúde é um alicerce essencial para a preservação e o desenvolvimento de todas as formas de vida, abrangendo desde os seres humanos até as mais variadas espécies animais. Contudo, o conceito de saúde, frequentemente associado à medicina humana, encontra na medicina veterinária uma dimensão igualmente indispensável, estendendo-se ao cuidado de seres vivos e à proteção da saúde pública. A medicina veterinária desempenha um papel crucial na prevenção de zoonoses, doenças transmissíveis entre animais e seres humanos, destacando sua importância no âmbito da saúde global e no equilíbrio das interações entre espécies.

A medicina veterinária contemporânea divide-se, de modo geral, em duas grandes áreas de atuação: a Medicina Veterinária Preventiva e a Medicina Veterinária voltada para a Saúde Pública. A primeira baseia-se em conhecimentos epidemiológicos que auxiliam na prevenção de doenças que afetam populações animais e que, em muitos casos, possuem impacto direto na saúde humana. A segunda concentra-se na higiene e segurança alimentar, implementando práticas que evitam a disseminação de patógenos na cadeia alimentar, protegendo as populações humanas (PFUETZENREITER et al., 2004).

Esse cenário destaca a abrangência da medicina veterinária não apenas no tratamento de doenças em animais, mas também na manutenção do equilíbrio ecológico, na melhoria da produção de alimentos e na salvaguarda das populações humanas. A atuação veterinária se estende desde a administração de grandes rebanhos em fazendas e reservas ecológicas até o cuidado de animais de companhia que integram o cotidiano familiar.

É nesse contexto que se insere o presente Trabalho de Conclusão de Curso (TCC), que documenta e analisa o desenvolvimento do projeto de pesquisa intitulado “Internet das Coisas (IoT) para o Monitoramento de Saúde de Animais de Pequeno e Grande Porte”. Fruto de uma colaboração entre o aluno autor deste trabalho, o bolsista Nathannael Fonseca e o professor orientador Bruno Lyra, o projeto visa explorar o potencial da IoT para o monitoramento eficiente, contínuo e remoto da saúde animal.

Utilizando tecnologias de baixo custo, como o módulo ESP8266, amplamente reconhecido por sua capacidade de conectar dispositivos à internet (ROBOCORE, 2024), o sistema coleta dados vitais como eletrocardiograma (ECG), temperatura

corporal e batimentos cardíacos por minuto (BPM). Esses dados são transmitidos utilizando o protocolo MQTT (*Message Queuing Telemetry Transport*), uma solução confiável e leve para comunicação em projetos de IoT, permitindo que sejam armazenados e visualizados em uma plataforma web. Essa plataforma oferece tanto a análise em tempo real quanto consultas históricas com base em períodos específicos definidos pelo usuário.

O projeto busca atender às necessidades de médicos veterinários e pesquisadores ao propor uma solução prática, acessível e escalável para o monitoramento da saúde animal. A tecnologia desenvolvida apresenta potencial para beneficiar desde pequenos animais de estimação até grandes rebanhos ou populações de animais selvagens, reforçando a importância da integração de tecnologias emergentes no campo da medicina veterinária e no monitoramento ambiental.

Com essa abordagem, o trabalho contribui para a inovação no setor, aliando conhecimentos de saúde animal, tecnologia e sustentabilidade. A pesquisa evidencia o papel da IoT na ampliação das capacidades de monitoramento e no suporte à tomada de decisões, promovendo, assim, benefícios tanto no campo da saúde quanto no desenvolvimento sustentável da interação entre humanos e animais.

1.2. Objetivos Gerais

Desenvolver uma ferramenta web que possibilite o monitoramento remoto e contínuo da saúde de animais de pequeno e grande porte, fornecendo dados em tempo real e históricos sobre eletrocardiograma (ECG), temperatura corporal e batimentos por minuto (BPM). A solução utiliza o protocolo MQTT para transmitir os dados coletados pelo dispositivo ESP8266, integrando-os a uma plataforma de armazenamento em nuvem e à página web por meio de um broker.

1.3. Objetivos Específicos

1.3.1. Implementar uma plataforma web que permita a visualização dos dados coletados em tempo real, assim como a consulta de dados armazenados, permitindo a comparação de períodos distintos para futuros diagnósticos.

1.3.2. Explorar o uso do protocolo MQTT, otimizando o envio e recebimento de dados para garantir a eficiência do sistema de monitoramento em tempo real.

1.3.3. Proporcionar suporte ao armazenamento em nuvem, facilitando o acesso a dados históricos e a possibilidade de análise retroativo para pesquisa e acompanhamento de diagnósticos.

1.3.4. Auxiliar pesquisadores e profissionais de medicina veterinária oferecendo uma solução tecnológica que integra os dados coletados com ferramentas de análise e comparação, otimizando a prática clínica.

1.4. Relevância para a Comunidade Científica

A relevância deste projeto reside na crescente aplicação de tecnologias IoT no campo da saúde, destacando-se por expandir seu uso para a medicina veterinária, tradicionalmente menos explorada em comparação às aplicações voltadas para seres humanos. Este trabalho foca no desenvolvimento de uma solução prática e economicamente viável para o monitoramento da saúde animal, atendendo à demanda por ferramentas que otimizem esse processo, especialmente em ambientes como fazendas e centros de pesquisa.

A utilização de dados em tempo real, aliados ao histórico de medições ao longo de períodos prolongados, permite análises mais detalhadas e precisas sobre o bem-estar animal. Essa abordagem não apenas aprimora os diagnósticos e tratamentos veterinários, como também contribui para o desenvolvimento de práticas de manejo mais eficazes.

Ao unir a tecnologia IoT à medicina veterinária, o projeto promove uma integração inovadora, reforçando o potencial dessas tecnologias no monitoramento contínuo e remoto da saúde animal. Além disso, estabelece uma base sólida para pesquisas futuras no campo da IoT aplicada à saúde veterinária, ampliando horizontes para soluções tecnológicas mais avançadas.

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os principais componentes, equipamentos e protocolos utilizados no projeto, detalhando os dispositivos físicos, tecnologias e metodologias que sustentam a estrutura do sistema. O foco é fornecer uma visão técnica da integração entre hardware e software que garante o funcionamento adequado da solução.

2.1. Internet das Coisas (IoT)

O termo IoT está relacionado a um conceito tecnológico com foco em conectar objetos do cotidiano, facilitando também a comunicação entre os mesmos e a internet, possibilitando a troca de dados ou informações. Um sistema básico de IoT funciona através da coleta e troca de dados em tempo real, tendo em vista 3 principais componentes, segundo a Amazon Webservices (2023):

I. **Dispositivo inteligentes:**

Trata-se de um dispositivo com recursos computacionais, como câmeras de segurança, televisores, *smartwatches* ou equipamentos de exercícios, capaz de coletar dados do ambiente, como sinais vitais ou padrões de uso. Esses dados são transmitidos via internet para a aplicação IoT, conforme ilustrado na Figura 1.

Figura 1 – Esquema de dispositivos IoT conectados à internet.

Fonte: Otávio Monsato de Paula (2017).

II. **Aplicação IoT:**

Uma aplicação consiste em um conjunto de serviços que integram os dados recebidos de dispositivos IoT, podendo utilizar tecnologias como machine learning ou inteligência artificial para analisar as informações e executar decisões previamente programadas.

III. **Interface gráfica ao usuário:**

Dispositivos IoT podem ser gerenciados por meio de interfaces gráficas intuitivas e de fácil uso, disponíveis em aplicativos móveis, como “IoT MQTT Panel” e “MQTT Box”, ou em plataformas web que permitem registro e controle.

Na área da saúde, o IoT desempenha um papel essencial ao possibilitar o monitoramento remoto de pacientes, transmitindo dados em tempo real para profissionais de saúde. Isso facilita o acompanhamento contínuo e aprimora a resposta em situações de emergência (HADIS et al, 2020).

2.2. Linguagens de programação

O desenvolvimento do projeto envolveu a criação de uma página web utilizando como base três linguagens fundamentais para o front-end: JavaScript, CSS e HTML. Para o back-end, foi utilizada a linguagem PHP.

2.2.1. CSS (*Cascading Style Sheets*)

Conforme MDN Web Docs (2024), o CSS é uma linguagem declarativa utilizada para controlar a apresentação visual de páginas web em navegadores. Ela aplica estilos predefinidos aos elementos especificados, por meio de declarações que definem propriedades e seus valores, como dimensões, formatos e outras características visuais, garantindo a exibição conforme projetado.

2.2.2. HTML (*Hyper Text Markup Language*)

De acordo com o MDN Web Docs (2024), o HTML é uma linguagem de marcação baseada em texto estruturado, composta por elementos delimitados por tags de abertura e fechamento. Seu principal objetivo é atribuir significado ao conteúdo da página, permitindo a definição de cabeçalhos, tabelas, imagens, vídeos e diversos outros elementos. Além disso, é amplamente utilizado para criar referências a outros textos, organizando e enriquecendo o conteúdo apresentado.

2.2.3. Javascript (JS)

O JavaScript é uma linguagem de programação que permite a criação de conteúdos dinâmicos, controle de multimídia, manipulação de imagens e outras funcionalidades interativas. Segundo o MDN Web Docs (2024), o JavaScript também possibilita a manipulação de dados, interação com dispositivos subjacentes ao navegador por meio de diversas APIs e a criação de gráficos, entre outras aplicações.

2.2.4. PHP (*Hypertext Processor*)

De acordo com a documentação do PHP, trata-se de uma linguagem de script de código aberto, projetada para uso geral, com ênfase no desenvolvimento web. Ela permite ser integrada a arquivos HTML, utilizando tags específicas de início e fim para alternar entre o modo PHP e o HTML.

Uma das principais vantagens do PHP é que sua execução ocorre no servidor, gerando um HTML que é então enviado ao navegador, sem que o usuário tenha acesso ao código fonte. Esse processo garante a segurança, impedindo o acesso a informações sensíveis. No presente projeto, o PHP é utilizado como parte do back-end da aplicação.

2.3. Projeto Paho

O Cliente Paho JavaScript é um projeto Open Source mantido pela Eclipse Foundation, que oferece diversas opções de clientes MQTT para várias linguagens de programação, visando aplicações de Máquina a Máquina (M2M) e Internet das Coisas (IoT) em novos desenvolvimentos ou sistemas em andamento (CRAGGS, 2017).

Ainda de acordo com Craggs (2017), essa ferramenta permite a implementação de um cliente MQTT em JavaScript utilizando WebSockets, possibilitando que a página web do projeto acesse dados MQTT em tempo real.

2.4. Protocolo MQTT (*Message Queuing Telemetry Transport*)

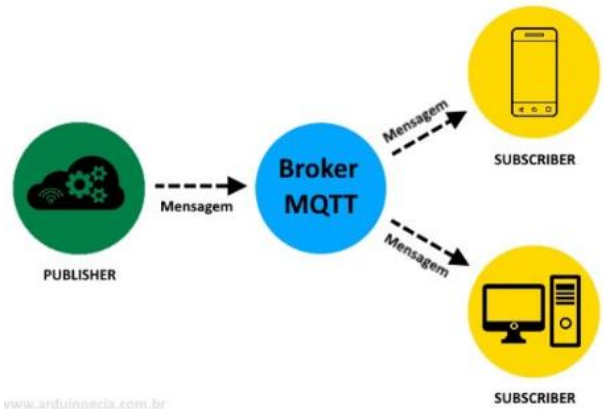
O MQTT é um protocolo de comunicação leve, projetado para a troca eficiente e confiável de dados entre dispositivos. Amplamente adotado no contexto de IoT, o MQTT é especialmente eficaz em ambientes com baixa largura de banda e alta latência. Utilizando o modelo de comunicação "publish/subscribe", ele é ideal para aplicações que demandam escalabilidade e gerenciamento dinâmico de clientes e tópicos (NERI et al., 2019).

O MQTT opera sobre o protocolo TCP/IP, garantindo a confiabilidade no envio e recebimento de mensagens, mesmo em redes com qualidade variável. Sua implementação simples e baixo consumo de energia o tornam adequado para dispositivos embarcados, como sensores e atuadores. A arquitetura do MQTT é composta por dois elementos principais: clientes, que podem publicar ou se inscrever em tópicos, e brokers, responsáveis pela mediação da troca de dados entre os clientes.

2.4.1. *Broker*

O broker é o servidor intermediário no protocolo MQTT, responsável por gerenciar e distribuir as mensagens entre os clientes (Figura 2). Ele recebe informações publicadas por um cliente e as distribui para os outros clientes inscritos no tópico correspondente.

Figura 2 – Fluxo entre clientes e broker.



Fonte: Arduino e Cia (2022).

Um ponto importante é que, caso não haja um cliente inscrito no tópico no momento da publicação, a informação é descartada e não fica armazenada no broker para futuros envios.

Existem diversos brokers disponíveis, tanto comerciais quanto de código aberto, e eles podem ser usados de forma local ou em servidores remotos. No projeto, foi utilizado o broker público do Mosquitto, acessível no servidor, o “test.mosquitto.org”. Este broker permite testes rápidos e acesso livre para desenvolvimento de aplicações.

Ainda com Neri et al (2019), o protocolo MQTT, além de funcionar sobre TCP (*Transmition Control Protocol*), também pode utilizar o protocolo MQTT-SN (*Sensor Network*), que é uma variação otimizada para redes de sensores com baixa potência, como as que utilizam UDP (*User Datagram Protocol*) ou Bluetooth para o transporte de dados.

2.4.2. Cliente

O cliente MQTT desempenha um papel ativo em dois aspectos principais da comunicação: postagem (*publish*) e recebimento (*subscribe*). Ele pode realizar uma dessas funções ou ambas simultaneamente, dependendo das necessidades do sistema (AWS, 2023). Esse modelo permite que os clientes se comuniquem sem saber diretamente quais outros dispositivos ou aplicações estão envolvidos, pois toda a intermediação é feita pelo broker. A comunicação eficiente e a flexibilidade para alternar entre as funções de publicação e assinatura são essenciais para o desenvolvimento de sistemas IoT que exigem monitoramento em tempo real e controle remoto.

2.4.3. Qualidade do serviço (QoS)

O protocolo MQTT oferece três níveis de Qualidade de Serviço (QoS), que definem como a entrega de mensagens é gerenciada:

I. **QoS 0** – No máximo uma vez: Trata-se do nível mais rápido de comunicação, porém menos confiável. Se houver falhas na transmissão, a mensagem pode ser perdida.

II. **QoS 1** – No mínimo uma vez: Nesse caso, a mensagem é retransmitida continuamente até que o remetente receba uma confirmação de recebimento do destinatário. Contudo, esse mecanismo pode resultar no destinatário recebendo mensagens duplicadas devido à ausência de um controle de duplicidade.

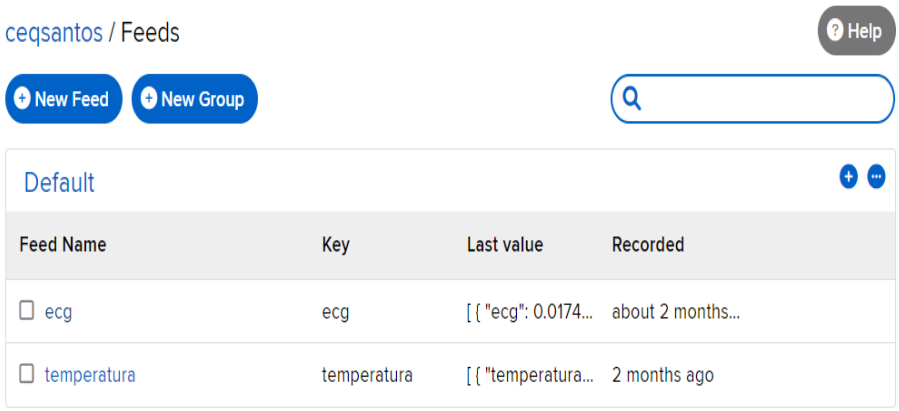
III. **QoS 2** – Exatamente uma vez: Este é o nível mais seguro, pois garante que a mensagem será entregue apenas uma vez, eliminando duplicidades e proporcionando maior confiabilidade, embora com um aumento na latência.

2.5. Plataforma em Nuvem – Adafruit IO

A computação em nuvem é um modelo relativamente recente no campo da tecnologia, permitindo o acesso remoto a recursos computacionais, como armazenamento e processamento de dados, via internet (ARMBRUST et al., 2010).

O Adafruit IO é uma plataforma em nuvem desenvolvida pela Adafruit Industries, projetada especificamente para aplicações de Internet das Coisas (IoT). Essa plataforma oferece uma interface intuitiva para o gerenciamento e visualização de dados gerados por dispositivos conectados à internet. Ela permite que os usuários criem e administrem feeds, que são seções dedicadas ao armazenamento e à visualização desses dados (Figura 3).

Figura 3 – Menu de Feeds do adafruit IO.



Fonte: Adafruit IO

Cada feed no Adafruit IO é composto por um nome único, chaves (*keys*) de autenticação e uma API (*Application Programming Interface*) que facilita a interação com os dados. Ao acessar a API, por padrão, o usuário recebe apenas o último valor registrado. Contudo, para acessar um histórico mais completo dos dados armazenados, é recomendado que o termo "/data" seja adicionado ao final do link da API, conforme indicado na documentação da plataforma (API REFERENCE, 2024). Isso permite que todos os dados contidos no feed sejam recuperados de uma vez, proporcionando maior flexibilidade na análise dos dados ao longo do tempo.

Para utilizar o Adafruit IO, é necessário criar uma conta gratuita na plataforma. Após a criação da conta, os usuários podem gerar feeds para armazenar dados provenientes de seus dispositivos IoT. Cada dado armazenado inclui um campo “Created At”, que marca a data e hora em que o dado foi recebido, junto com o valor correspondente.

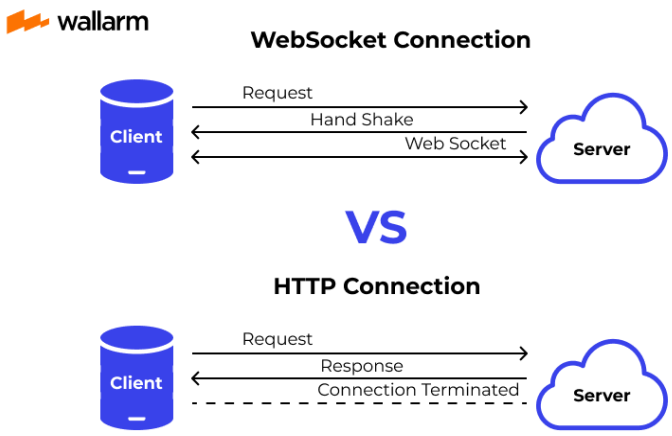
O Adafruit IO também oferece integração com diversos protocolos e dispositivos de IoT, incluindo o MQTT, o que facilita a comunicação em tempo real entre os dispositivos conectados e a plataforma. A utilização de "triggers" também permite automações, como enviar notificações ou executar ações específicas quando determinados valores são atingidos, ampliando as funcionalidades da plataforma para casos de uso em sistemas de monitoramento remoto e controle.

2.6. Tecnologia WebSockets

Os WebSockets constituem uma tecnologia que permite a comunicação bidirecional em tempo real entre um cliente (no contexto deste projeto, o navegador) e um servidor por meio de uma única conexão TCP. Essa tecnologia, possibilita que o servidor envie informações ao cliente sem a necessidade de requisições repetidas, tornando-se, assim, ideal para aplicações que requerem atualizações frequentes e em tempo real (CLEMENTE, 2024).

O funcionamento dos Websockets se inicia com uma conexão estabelecida por meio do protocolo HTTP tradicional, onde o cliente solicita ao servidor um “upgrade” para a tecnologia Websocket. Uma vez que essa requisição é atendida (*Hand shake*), a comunicação se mantém de maneira persistente, permitindo a troca livre de informações até que uma das partes decida encerrar a conexão (Figura 4).

Figura 4 – Comparação de conexão WebSockets vs HTTP.



Fonte: Mukhadin Beschokov (2024).

Segundo Beschokov (2024), a utilização dos Websockets apresenta diversas vantagens, como baixa latência, visto que não é mais necessária a realização de múltiplas requisições como no protocolo HTTP (*Hypertext Transfer Protocol*). Dessa forma, a comunicação se torna mais rápida e eficiente, além de contínua, o que o torna perfeito para aplicações que exigem atualizações constantes ou em tempo real.

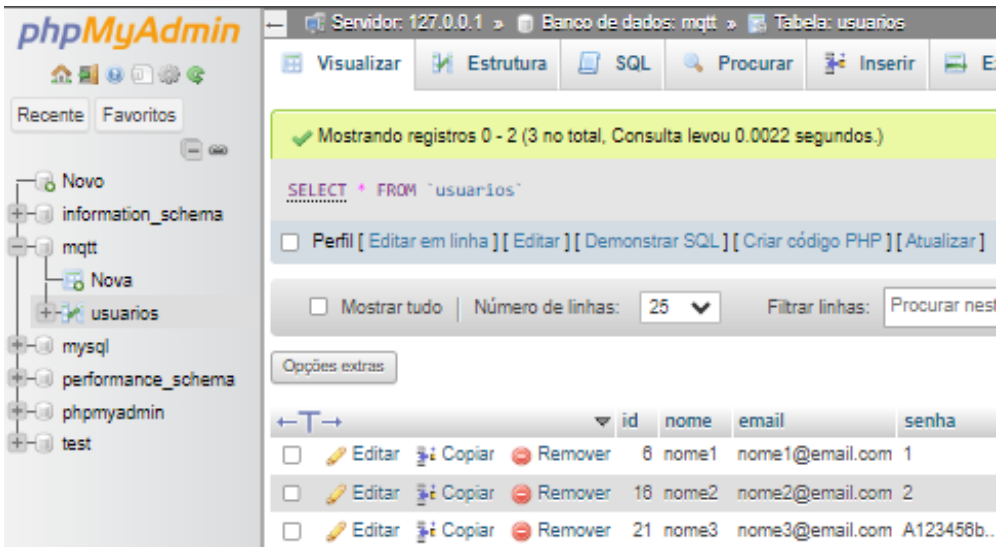
2.7. XAMPP

O XAMPP é uma distribuição *open-source* projetada para simplificar a instalação de um servidor web local, integrando as principais tecnologias utilizadas no desenvolvimento web: Apache, MySQL, PHP e Perl (XAMPP, 2024). Essa solução é especialmente útil para desenvolvedores que necessitam de um ambiente de testes local para criar, modificar e testar websites e aplicações antes de sua publicação online.

O XAMPP permite simular um servidor real, oferecendo um ambiente seguro e de fácil gerenciamento para o desenvolvimento de projetos. Através dessa ferramenta, é possível configurar rapidamente um servidor web completo, o que facilita o aprendizado e a prototipagem de aplicações (GARCIA, 2024).

Uma das características notáveis do XAMPP é sua interface gráfica, que possibilita o gerenciamento do banco de dados MySQL por meio do "phpMyAdmin" (Figura 5). Essa funcionalidade foi crucial para o projeto, pois facilitou o gerenciamento e a visualização do armazenamento de dados dos usuários, como senhas, nomes, e-mails e IDs (*identity*) nas tabelas. Além disso, é uma aplicação de baixo custo computacional em comparação com outras soluções como o MySQL Workbench.

Figura 5 – Interface gráfica da tabela de usuários com dados de teste para login.



Fonte: Autor.

Para acessar o phpMyAdmin, o usuário deve abrir o painel de controle do XAMPP, que se torna disponível após a instalação do software. É necessário ativar os serviços do Apache e do MySQL. No entanto, deve-se ter cautela durante esse processo, pois, se houver outra instância do MySQL em execução no dispositivo, poderá ocorrer um conflito, impedindo a inicialização do XAMPP. Nestes casos, é recomendado verificar o Gerenciador de Tarefas do sistema para identificar se outra instância do MySQL está ativa e encerrá-la antes de continuar o uso do XAMPP.

3. TRABALHOS CORRELATOS

A fim de proporcionar um aprofundamento nos conceitos que serão utilizados no desenvolvimento desse projeto, serão tomados como referência 3 trabalhos previamente realizados e publicados no IEEEXplore. Essas referências são imprescindíveis para a compreensão dos princípios fundamentais e das abordagens utilizadas em projetos semelhantes. Cada um deles possui ênfase na área específica no qual o projeto atual está inserido.

Em vista disso, dentre os trabalhos selecionados temos: "*IoT Based Real-Time Remote Patient Monitoring System*" de Yew et al. (2020), "*An IoT-Cloud Based Health Monitoring Wearable Device For Covid Patients*" de Hafsiya e Rosa (2021), por fim, "*IoT based low-cost distant patient ECG monitoring system*" de Singh, P. e Jasuja (2017).

Utilizando esses trabalhos como referência, será possível usar como base o desenvolvimento no projeto atual, a qual se refere esse TCC, com as pesquisas citadas. A análise das abordagens tomadas, bem como, a tecnologia implementada e os resultados obtidos, fornecerão a percepção necessária para o planejamento e concretização do projeto em questão.

3.1. *IoT Based Real-Time Remote Patient Monitoring System*

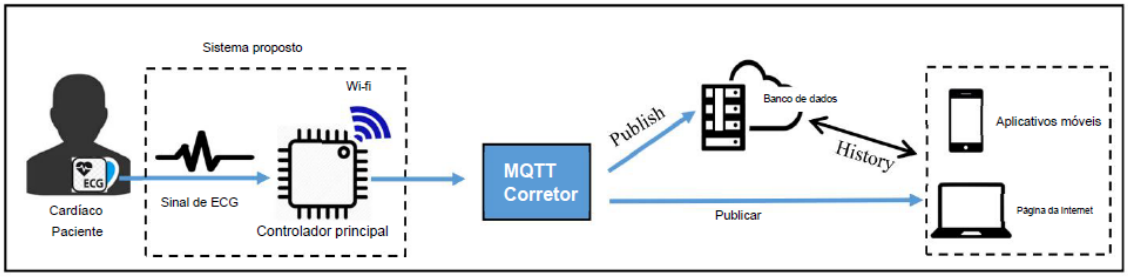
Inicialmente, os autores apresentam a estatística populacional da Malásia, destacando a divisão entre residentes urbanos e rurais. De acordo com um estudo do Ministério do Desenvolvimento Rural da Malásia, aproximadamente 25% da população reside fora dos centros urbanos. Em seguida, é apresentado um levantamento sobre a proporcionalidade entre médicos e a população, abordando as razões pelas quais certas regiões possuem uma menor concentração de médicos.

Em sequência, os autores propõem um sistema de monitoramento remoto de pacientes, justificando que essa abordagem permite que médicos acompanhem a

saúde dos pacientes independentemente da localização geográfica e do momento. Além disso, é sugerido um sistema de monitoramento de ECG em tempo real baseado em IoT, destacando os desafios enfrentados, como a perda de pacotes e os atrasos na transmissão, que afetam a integridade dos dados em tempo real. É enfatizada a diferença em relação aos dados armazenados, que podem ser consultados posteriormente, sem o impacto desses problemas.

O sistema proposto pelos autores é composto por quatro módulos: sensor, controlador, módulo de comunicação e servidor web. O sensor coleta os dados vitais do paciente, enquanto o controlador transmite essas informações, utilizando o protocolo MQTT, tanto para o banco de dados (BD) para armazenamento quanto para a interface do usuário, que pode ser uma aplicação ou página web (Figura 6).O sistema também gera informações de BPM, atualizadas a cada 3 segundos, com base nos dados de ECG.

Figura 6 – Esquema geral do projeto proposto por Yew et al (2020).



Fonte: Yew et al. (2020)

Por fim, os autores apresentam os resultados de seu projeto, afirmando que não houve erros ou perdas de pacotes em redes públicas ou privadas. No entanto, em redes públicas, o atraso pode chegar a até 1 segundo, um valor consideravelmente maior devido às condições de distância, enquanto em redes privadas o atraso foi de até 50,08 ms, considerado aceitável.

Além disso, foi realizada uma comparação entre os dados exibidos em tempo real e os dados armazenados na nuvem, constatando-se que ambos eram semelhantes.

3.2. An IoT-Cloud Based Health Monitoring Wearable Device For Covid Patients

O projeto proposto por Yew et al. (2020) compartilha semelhanças com o sistema descrito, especialmente no contexto do monitoramento remoto de pacientes durante a pandemia de COVID-19. Os autores abordam a problemática de reduzir a

superlotação hospitalar e garantir a integridade dos pacientes ao permitir monitoramento remoto, o que também minimiza a exposição dos profissionais de saúde.

A proposta destaca que, mesmo após testar negativo, um paciente pode desenvolver sintomas inesperadamente, o que justifica a necessidade de vigilância constante, sendo essa vigilância realizada através de um fluxo contínuo de dados em tempo real. Além disso, o sistema é projetado para ser não invasivo, utilizando sensores para monitorar parâmetros como temperatura, frequência respiratória e níveis de oxigênio.

O principal objetivo da solução é responder à ameaça representada pela COVID-19, ao mesmo tempo em que reforça a prontidão dos sistemas nacionais de saúde na Índia, onde a sobrecarga nos hospitais era um risco iminente. O sistema proposto permite que os médicos monitorem a saúde dos pacientes em tempo real e tomem decisões rápidas, reagindo ao primeiro sinal de sofrimento, mesmo quando o paciente já foi liberado da unidade hospitalar.

O artigo também compara o projeto proposto com outros estudos semelhantes, todos focados no monitoramento de saúde baseado em IoT. A metodologia do sistema enfatiza a importância do monitoramento remoto, particularmente durante a pandemia, onde os pacientes sintomáticos poderiam agravar a situação ao se deslocarem até o hospital, contribuindo para o aumento da transmissão do COVID-19. O monitoramento remoto ajuda a determinar se uma consulta presencial é realmente necessária, aliviando a pressão sobre os hospitais e facilitando o distanciamento social.

O sistema proposto é estruturado em três fases essenciais: (1) a fase de transmissão, na qual sensores vestíveis coletam dados dos pacientes; (2) a fase de recebimento, onde um dispositivo no ambiente do paciente coleta esses dados; e (3) a fase de monitoramento, onde a equipe médica visualiza as informações por meio de uma aplicação web.

A arquitetura do sistema é baseada no Raspberry Pi, um minicomputador, que recebe os dados digitalizados do Arduino (que converte os valores dos sensores) e os transmite via Wi-Fi para a seção receptora no Raspberry Pi. Os dados dos sensores são, então, armazenados na nuvem utilizando os protocolos MQTT e HTTP.

A equipe médica pode acessar os dados coletados e visualizá-los em dispositivos móveis por meio de aplicações como MQTT Dash ou IoT MQTT Panel. O

sistema, portanto, proporciona um monitoramento remoto eficaz e acessível para médicos e pacientes.

Por fim, os autores discutem o impacto positivo da tecnologia tanto para os pacientes quanto para a equipe médica, permitindo um monitoramento contínuo que limita a disseminação da COVID-19, especialmente durante a fase inicial de sintomas. A solução também contribui para o alívio da pressão sobre os sistemas de saúde ao permitir que os médicos avaliem a gravidade da condição dos pacientes sem a necessidade de visitas físicas constantes.

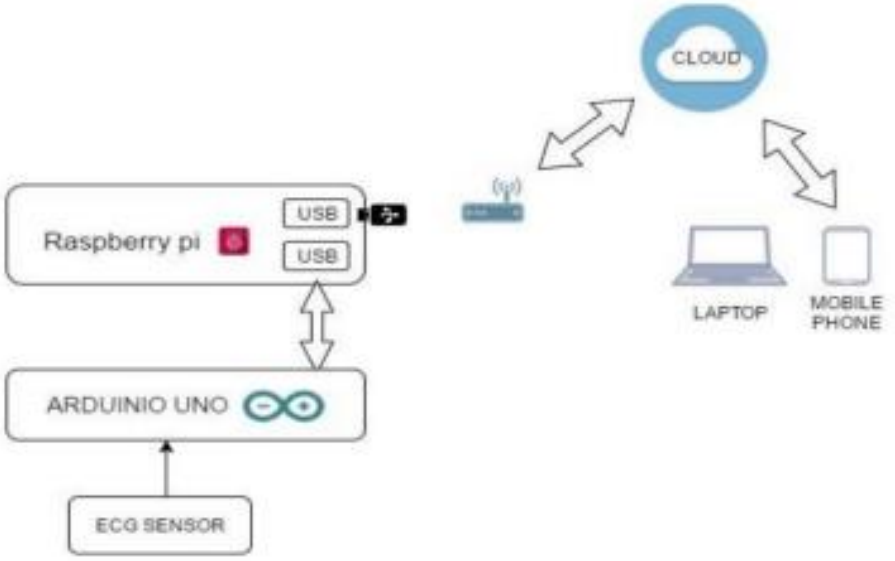
3.3. IoT based low-cost distant patient ECG monitoring system

Os autores iniciam explicando o conceito de Internet das Coisas (IoT) e sua aplicação global, destacando seu potencial para resolver problemas sociais, como o monitoramento remoto de pacientes, especialmente idosos com complicações cardiovasculares.

Apresentam projeções de crescimento populacional idosa até 2060, ressaltando a necessidade de sistemas IoT para reduzir custos com consultas médicas presenciais e hospitalizações. Eles destacam que essas soluções podem facilitar o acesso remoto a cuidados médicos, economizando tempo e evitando deslocamentos.

O projeto foi fundamentado em pesquisas correlatas, as quais orientaram o desenvolvimento da solução proposta, detalhada no fluxo de informações apresentado na Figura 7.

Figura 7 – Implementação do sistema proposto pelos autores Singh e Jasuja (2017).



Fonte: Singh, P. e Jasuja, A. (2017)

Em termos de hardware, o sistema é composto por um Raspberry Pi, uma placa microcontroladora Arduino e um sensor de ECG AD8232. Quanto ao software, o sistema operacional Raspbian, baseado em Linux, é instalado tanto no Raspberry Pi quanto no Arduino.

A aplicação é desenvolvida utilizando o ambiente de desenvolvimento integrado (IDE) do Arduino, enquanto a ferramenta de programação visual Node-RED, baseada em JavaScript, é utilizada no Raspberry Pi para o processamento dos dados. O protocolo MQTT também é discutido como parte integral da comunicação entre os dispositivos.

A configuração experimental do sistema é dividida em três partes principais: (1) Sensoriamento – os dados são coletados diretamente do corpo do paciente utilizando eletrodos, sendo processados para reduzir o ruído e amplificar o sinal útil antes de serem enviados para o próximo estágio; (2) Unidade de Processamento de Sinais – o Arduino recebe os dados do Raspberry Pi, converte-os de analógicos para digitais e os envia para a nuvem; (3) Tráfego em Nuvem – os dados são transmitidos por meio do IoT Bluemix, com segurança nativa garantida por tipo, ID e token do dispositivo, permitindo que apenas médicos autorizados acessem as informações do paciente.

Em relação aos custos financeiros, os autores discutem os investimentos necessários para o desenvolvimento do projeto. Nos resultados, afirmam que obtiveram sucesso na consulta e exibição dos dados em tempo real, utilizando o painel do IoT Bluemix, onde o sinal de ECG foi visualizado em formato gráfico.

Concluem o trabalho com sugestões para pesquisas futuras, focando na tomada de decisões automáticas em casos de anomalias detectadas pelo sistema, aprimorando ainda mais o monitoramento remoto e a resposta médica.

3.4. Comparações

Tabela comparativa entre os estudos relacionados e o presente TCC.

Aspectos	Pesquisas Mencionadas	Este TCC
Papel das Pesquisas	Embasamento teórico para a compreensão das práticas e desafios de conexões em campo real.	Auxílio específico aos médicos veterinários com obtenção de dados clínicos de animais, utilizando IoT.

Objetivos	Desenvolver um sistema IoT capaz de garantir os cuidados necessários para os pacientes e garantir a integridade dos profissionais.	Desenvolver um sistema IoT para médicos veterinários obterem dados clínicos, com potencial aplicação futura em humanos.
Aplicação	Voltado para o cuidado humano, com aplicação hospitalar ou periodicamente por um paciente em seu lar.	Focado inicialmente em veterinária, para pesquisadores da UFPA, com possível expansão futura para humanos.
Motivação Principal	Monitoramento independente de geografia; riscos epidemiológicos; mobilidade e lazer.	Agilidade e baixo custo na obtenção de resultados clínicos para diagnósticos veterinários.
Impacto Esperado	Ajudar a visualizar os desafios e aplicações de conexões reais.	Proporcionar ferramentas ágeis e econômicas para a medicina veterinária.

Fonte: Autor.

4. DESENVOLVIMENTO

Nesta seção do trabalho, será abordado desde a origem do projeto, além de suas respectivas fases até a etapa final, com a integração do ESP8266 ao BD e a página web. Outrossim, será explorado de forma contextualizada o projeto ao qual este trabalho se faz complemento, desde o momento de coleta dos dados, o processamento do sinal e envio dos mesmos tratados para análises ou diagnósticos.

4.1. Testes iniciais

O primeiro contato com o protocolo MQTT, iniciou-se com a tentativa de enviar dados para entender a estrutura de seu funcionamento. A instalação do servidor Mosquitto em uma máquina permitiu a realização dos primeiros testes via prompt de comando, enviando Strings, denominadas "payloads" no contexto do MQTT.

No entanto, foi identificado que um conhecimento básico sobre o protocolo seria fundamental. Para isso, foi realizado um curso gratuito disponível no canal “InterneteCoisas” (Internet e Coisas, 2019) no YouTube, que abordou os princípios de funcionamento do protocolo MQTT, suas implementações e características. O curso

também apresentou ferramentas úteis para os testes, como o “MQTT Box” e o “MQTTIeC”, desenvolvidas pelo próprio canal.

Ao término do curso, o instrutor demonstrou um projeto de exibição de temperatura e umidade em tempo real, que serviu de base para a adaptação e desenvolvimento inicial do projeto. No editor de código Visual Studio Code (VS Code), foi criado o primeiro arquivo front-end, exibindo dados relacionados à temperatura. Os testes foram realizados remotamente, com o aluno Nathannael transmitindo os dados a partir de outra cidade.

Reconhecendo a complexidade das modificações em uma aplicação Web simples, ficou evidente a necessidade de um aprofundamento em lógica de programação e desenvolvimento web. Com isso, o acesso aos cursos da plataforma Danki Code permitiu o aprimoramento das habilidades necessárias, por meio dos cursos de “Lógica de Programação” e “Desenvolvimento Web Completo”.

Até a apresentação deste TCC, o projeto passou por 11 versões. Contudo, serão discutidas apenas as 8 versões que sofreram modificações significativas, as quais impulsionaram o desenvolvimento do projeto até a versão final, que será analisada detalhadamente.

4.2. Versão 1

A página era capaz de exibir os valores de temperatura recebidos via MQTT, utilizando a conexão com o servidor “test.mosquitto.org”, na porta “1883”, com o tópico de publicação/assinatura definido como “teste/temperatura123”.

O intervalo de tempo para atualização do gráfico poderia ser ajustado diretamente no código. Inicialmente, o gráfico era atualizado a cada 1 segundo, exibindo o último valor recebido. Caso o intervalo não fosse alterado, o gráfico gerava uma linha reta ao longo do tempo.

Nesta versão, não havia armazenamento de dados, apenas a exibição gráfica da temperatura. Posteriormente, foi percebida a necessidade de armazenar as informações, o que foi possível com o uso do MySQL, permitindo o armazenamento local dos dados.

4.3. Versão 2

Nesta fase do projeto, foi implementada uma consulta de dados armazenados por períodos específicos, inicialmente utilizando uma API local baseada no framework API Flash, que fazia a interface entre a página web e um banco de dados MySQL hospedado localmente. A consulta era realizada com o usuário especificando o

intervalo de tempo desejado, selecionando a data, hora e minuto de início e fim. Somente os valores de temperatura dentro desse intervalo eram recuperados e exibidos graficamente.

Na versão 4, a API Flash e o banco de dados MySQL foram substituídos pelas APIs dos feeds do Adafruit IO, permitindo o armazenamento dos dados na nuvem. Essa mudança possibilitou a comunicação bidirecional de dados entre o ESP8266 e a página web, independentemente da localização geográfica, aproveitando servidores sempre ativos e requerendo apenas uma conexão com a internet. Essa configuração facilitou significativamente a realização de testes a longa distância.

4.4. Versão 3

Foram implementadas modificações no sistema com o objetivo de reduzir a sobrecarga causada pelo envio individual de dados ao banco de dados. Essa necessidade tornou-se especialmente crítica com a adição do componente de ECG ao projeto, exigindo uma abordagem mais eficiente no processamento e armazenamento das informações.

Como solução, foi adotada a proposta de agrupar os dados no formato JSON (*JavaScript Object Notation*). Nesse formato, os dados eram simulados e inseridos manualmente no banco de dados local, contendo o valor da variável e seu respectivo timestamp, que representava a marca temporal de cada dado. A página web foi configurada para interpretar esse formato e gerar o gráfico correspondente à temperatura.

4.5. Versão 4

Neste momento, o Adafruit IO foi utilizado como banco de dados, exibindo um gráfico para cada componente, com os valores extraídos do feed de temperatura criado na plataforma. Por padrão, o link da API fornecido pelo Adafruit IO carregava apenas o último dado recebido pelo feed. Contudo, através da documentação do Adafruit IO, foi possível contornar essa limitação, acrescentando "/data" ao final do link da API. Dessa forma, todos os dados armazenados no feed eram carregados quando a API era consultada.

No entanto, novas limitações surgiram à medida que as restrições do Adafruit IO eram superadas. A plataforma permitia a leitura de no máximo 100 linhas de dados no feed. Após consulta à documentação, foi possível aumentar esse limite para 1000 linhas. No entanto, essa limitação permaneceu na versão gratuita da plataforma, o

que poderia ser um problema caso o usuário precisasse consultar grandes volumes de dados, já que apenas as 1000 linhas mais recentes seriam exibidas graficamente.

Nesse estágio, o conceito de IoT perdeu parte de sua relevância, pois os dados vitais coletados só podiam ser acessados quando solicitados, sem a exibição em tempo real dos dados.

4.6. Versão 5

Nesta versão, foi implementado um aprimoramento significativo com a integração de uma API dedicada ao ECG, o que possibilitou a geração de um gráfico adicional, ao lado do gráfico de Temperatura. A exibição dos gráficos era condicional: caso apenas um conjunto de dados estivesse disponível, somente o gráfico correspondente era gerado; caso ambos os conjuntos de dados estivessem presentes, ambos os gráficos eram exibidos simultaneamente; caso contrário, nenhum gráfico era renderizado. Essa estrutura foi projetada de forma independente, pois cada gráfico obteve seus dados de APIs distintas.

Uma nova limitação foi identificada nessa etapa, relacionada ao tamanho do *payload* que o Adafruit IO pode processar. O sistema permitia o envio de um *payload* JSON contendo aproximadamente 5 a 10 valores. Caso esse limite fosse excedido, toda a informação seria comprometida. Cada grupo de dados recebido, ou seja, cada linha no feed, incluía um campo "Created At", que indicava o momento de criação da linha de dados. Essa informação era utilizada como referência para o eixo de tempo (eixo X) nos gráficos.

4.7. Versão 6

Na presente fase, tornou-se possível realizar a leitura dos dados JSON contidos no campo "value" nos feeds do Adafruit IO. O período a ser analisado para a consulta passou a ser o timestamp armazenado em cada valor de temperatura ou ECG no JSON, em vez do campo "Created At", que é um campo nativo do Adafruit IO responsável por registrar o momento em que os dados foram recebidos.

Uma das dificuldades encontradas foi a organização dos dados para a geração dos gráficos, uma vez que o JSON é estruturado em formato LIFO (*Last In, First Out*), que é o comportamento padrão de listas. Nesse formato, os dados mais recentes são adicionados ao final, enquanto os dados mais antigos ocupam as primeiras posições.

Por outro lado, o Adafruit IO utiliza o formato FIFO (*First In, First Out*), no qual o último dado inserido fica no topo e é o primeiro a ser lido, resultando em gráficos parcialmente invertidos.

Essa discrepância foi resolvida por meio da função "reverse()", que inverte a ordem de leitura dos feeds, permitindo que os dados sejam processados do mais antigo para o mais recente. Essa abordagem assegura uma organização adequada para a leitura dos dados, sincronizando os índices de leitura do JSON e os do Adafruit IO.

4.8. Versão 7

A autenticidade constitui um dos pilares fundamentais da segurança da informação, garantindo que uma informação tenha sido criada, transmitida, alterada ou excluída exclusivamente por um agente autorizado, seja ele um usuário, sistema, dispositivo ou entidade organizacional.

Nesse sentido, a proteção das credenciais de acesso, como logins e senhas, é imprescindível, pois elas servem como identificadores exclusivos para assegurar a autenticidade dos acessos. Recomenda-se que tais credenciais sejam tratadas com sigilo, evitando o compartilhamento com terceiros e a reutilização em diferentes plataformas.

Em complemento, o uso do e-mail institucional deve ser restrito às atividades corporativas, evitando seu emprego em cadastros pessoais, especialmente em sites não confiáveis. A reutilização de senhas institucionais em outras aplicações também deve ser evitada, uma vez que um eventual vazamento de dados pode comprometer tanto a integridade das informações organizacionais quanto a autenticidade dos usuários (LNCC, 2024).

No desenvolvimento do sistema, foram implementadas funcionalidades de autenticação de usuários baseadas no modelo CRUD (*Create*, *Read*, *Update*, *Delete*). Esse modelo, que compreende as operações de criação, leitura, atualização e exclusão de registros, fundamentou a estruturação da autenticação, atendendo aos seguintes requisitos:

I. Tela de login;
II. Página inicial;
III. Tela de cadastro;
IV. Tela para edição de dados;
V. Exclusão de usuário.

Com o XAMPP configurado para ativar o servidor de banco de dados, foi criada a tabela "usuários" na extensão MySQL local, com os campos: "id" (chave primária), "nome", "login" e "senha", todos do tipo VARCHAR, destinada ao armazenamento de credenciais de login e verificações de autenticação.

Os arquivos de conexão com o banco de dados e os parâmetros de sessão foram desenvolvidos com base no curso PHP Jedai: Métodos Mágicos, da Danki Code, e adaptados às necessidades do projeto. A funcionalidade de sessão, que utiliza um *array* temporário para armazenar informações do usuário extraídas do BD, foi implementada seguindo boas práticas de programação.

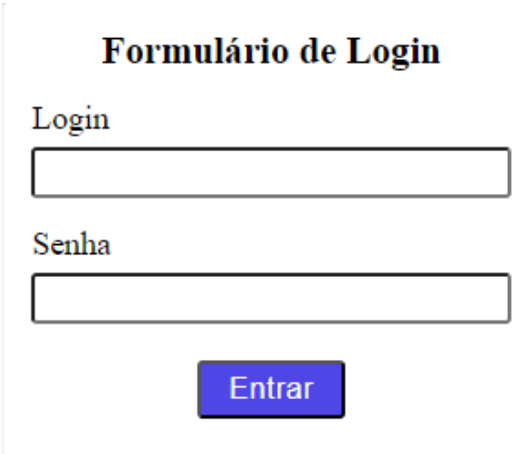
Os arquivos "config.php" e "BD.php" foram mantidos separados para modularidade e fácil manutenção. A classe "BD.php" estabelece a conexão com o banco via XAMPP, enquanto a "config.php" define diretórios e parâmetros necessários para o login.

O projeto foi desenvolvido com base em uma regra de negócio orientada à segurança, conforme IBM (2024), que estabelece diretrizes automatizadas como conformidades e aprovações, condicionando ações ao cumprimento de critérios específicos.

Na interface, ao acessar a tela de login (Figura 8), o sistema verifica a existência de uma sessão ativa. Caso positiva, o usuário é redirecionado à página inicial; caso contrário, deve preencher o formulário com e-mail e senha. Os dados são validados quanto ao formato do e-mail (@email.com) e preenchimento obrigatório.

Essa estrutura assegura alto nível de segurança no processo de autenticação e gerenciamento de acesso.

Figura 8 – Tela de login.



Fonte: Autor.

A validação de sessão nas páginas secundárias utiliza uma abordagem inversa, verificando a ausência de uma sessão ativa. Caso nenhuma sessão esteja em vigor, o sistema realiza o redirecionamento automático para a página de login. Essa lógica é implementada exclusivamente no back-end, reforçando a segurança ao restringir tentativas de exploração indevida do sistema.

Quando a validação é bem-sucedida, uma consulta ao banco de dados é executada para verificar se o e-mail e a senha fornecidos coincidem com os registros armazenados na tabela "usuario".

Caso os dados do formulário não correspondam a nenhuma entrada, a página é recarregada e o acesso é negado. Por outro lado, em caso de correspondência, uma sessão é inicializada, carregando as informações do usuário, e o mesmo é redirecionado para a página inicial. O funcionamento detalhado do processo de login está descrito no APÊNDICE H – Fluxograma de login do usuário.

A página inicial funciona como um ponto de acesso centralizado, oferecendo ao usuário opções como cadastro, edição ou exclusão de credenciais, acesso aos painéis de gráficos e logout. Informações como nome e e-mail do usuário são exibidas no topo da página, otimizando a navegação e identificação.

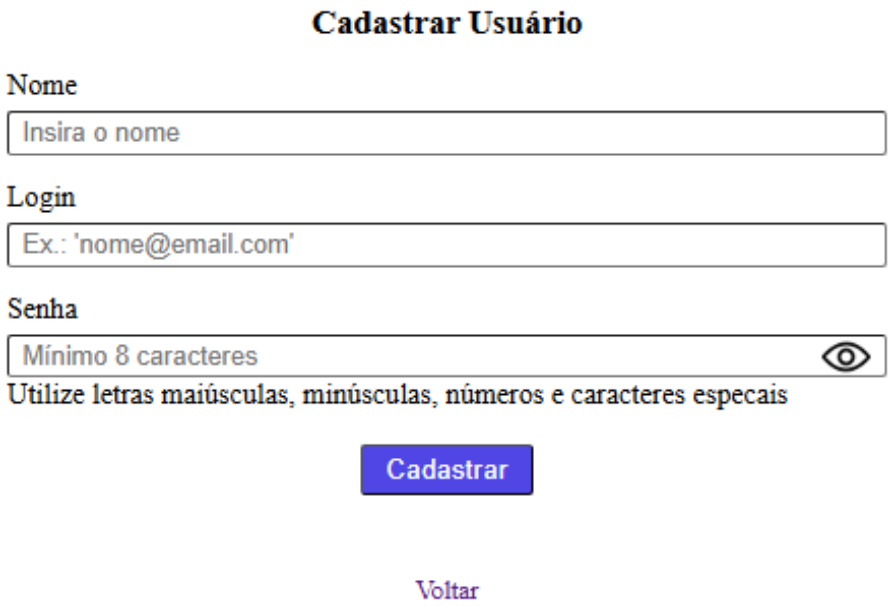
Todas essas funcionalidades estão organizadas em um menu lateral responsivo, baseado em Larissa Kich (2024). A Figura 9 ilustra a página inicial do projeto, onde cada item do menu conduz o usuário à página correspondente, assegurando uma navegação intuitiva e eficiente.

Figura 9 – Página inicial com barra lateral responsiva.

Fonte: Autor.

Para o cadastro de um novo usuário, é necessário que um usuário autorizado realize o processo dentro da plataforma. Após a validação da sessão ativa, o sistema verifica se todos os campos do formulário foram devidamente preenchidos (Figura 10). Caso contrário, mensagens de erro específicas são exibidas, como: "Nome vazio. Preencha-o!", "Senha não informada!", "E-mail inválido!" ou "E-mail não informado!".

Figura 10 – Tela de cadastro de novo usuário.



Fonte: Autor.

Em seguida, ocorre a validação da senha fornecida. Para ser aceita, a senha deve atender a critérios de complexidade, incluindo: no mínimo 8 caracteres, presença de letras maiúsculas e minúsculas, números, caracteres especiais e ausência de sequências alfabéticas ou numéricas simples.

Após atender todas as especificações, é realizada uma verificação no banco de dados para garantir que o e-mail informado no formulário seja único, já que o login é feito por meio deste dado.

Caso não exista outro usuário com o mesmo e-mail, a tentativa de cadastro é realizada. Se houver alguma falha no processo devido à conexão com o banco de dados ou outro erro imprevisto, uma mensagem de erro é exibida; caso contrário, uma mensagem de sucesso é apresentada com um link para retornar à página inicial.

A partir desse momento, o novo usuário está cadastrado e pode acessar a plataforma com os dados fornecidos no formulário. O APÊNDICE C – Fluxograma de cadastro de usuário mostra o fluxograma desse processo.

Se o usuário decidir alterar seus dados de acesso, ele pode acessar a seção de edição de dados (Figura 11). O formulário deve ser preenchido novamente com nome, e-mail e senha, atendendo às mesmas especificações do cadastro. Durante esse processo, é verificado se o ID do usuário é válido, utilizando um processo de verificação binária e comparativa.

Figura 11 – Tela de edição de dados do usuário.

Fonte: Autor.

Primeiramente, o sistema verifica se o usuário ainda existe, realizando uma consulta para contar o número de IDs na tabela de usuários que correspondem ao ID da sessão ativa. O esperado é que o resultado seja 1, representando o próprio usuário ativo, já que o ID é autoincrementado e, portanto, único para cada usuário.

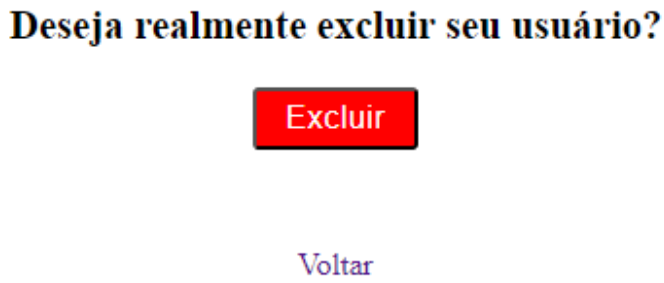
Em seguida, verifica-se se o e-mail fornecido para atualização já está em uso. A lógica é similar à anterior, onde se busca garantir que o ID e o e-mail sejam únicos. Caso o contador da consulta seja diferente de 0, a atualização não será permitida, e uma mensagem de erro será exibida.

Se o usuário existir no banco de dados, o e-mail não estiver em uso por outro usuário e os dados do formulário forem válidos, as informações são atualizadas tanto no banco de dados quanto no *array* de sessão para manter o usuário ativo, e uma mensagem de sucesso é exibida (ver APÊNDICE E – Fluxograma de edição de dados do usuário).

Por fim, para excluir o login do usuário, o sistema verifica se o ID no *array* da sessão está corretamente definido e se existe no banco de dados. A exclusão é

realizada removendo a linha correspondente ao ID da sessão ativa na tabela de usuários. A Figura 12 representa a tela de exclusão que pode ser acessada pelo usuário.

Figura 12 – Tela de exclusão de usuário logado.



Fonte: Autor

A *posteriori*, o *array* de sessão é limpo através do método *unset* e uma mensagem é exibida informando que a operação foi bem sucedida

Nesse momento, como os dado da sessão não existem mais, o usuário ao recarregar a página ou mesmo tentar voltar para a anterior, é redirecionado para a página de login (ver APÊNDICE F – Fluxograma de exclusão do login de acesso).

4.9. Versão 8

Até a versão anterior, os gráficos exibiam dados armazenados localmente, no banco de dados ou nos feeds do Adafruit IO. Com a integração das versões antigas, incluindo a versão 1 (única que exibia dados em tempo real), com as versões 6 e 7, o projeto passou a ser capaz de gerar gráficos em tempo real, recebendo dados diretamente do dispositivo ESP8266 via protocolo MQTT. Além disso, passou a exibir dados armazenados na nuvem nos feeds do Adafruit IO e implementou um sistema de login para cadastro, edição e exclusão de acesso.

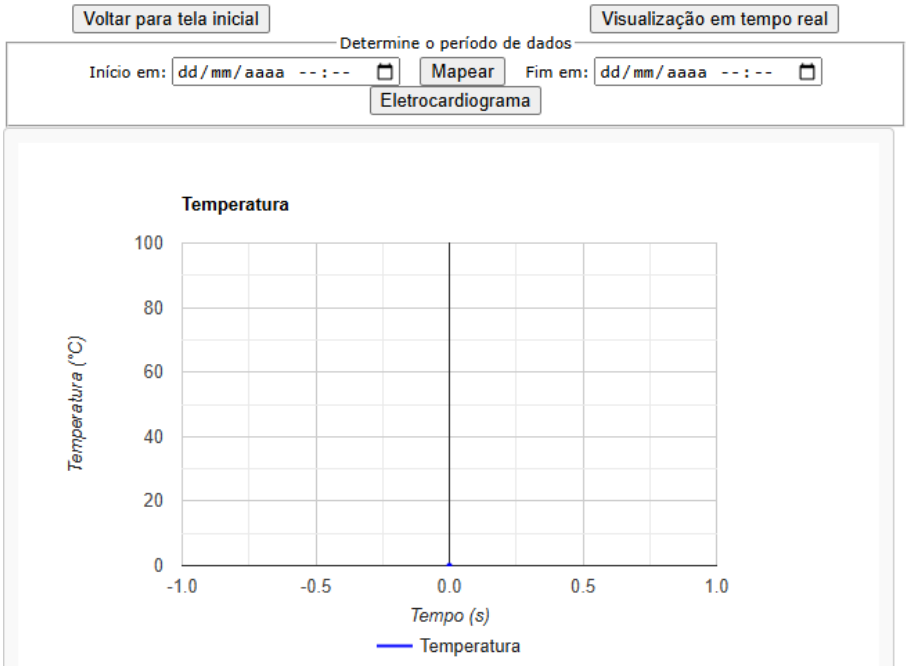
As modificações nesta versão incluem melhorias na organização da estrutura do código e na performance da exibição dos gráficos. Além disso, como mencionado na Versão 4, devido às limitações do Adafruit IO, foi necessário modificar o diretório para consulta de dados por período para uma tabela MySQL no mesmo banco de dados que contém as informações de login do usuário, descontinuando o uso do Adafruit IO. Essa mudança foi implementada para garantir que o sistema seja hospedado nos servidores da UFPA, dentro de seus domínios de rede.

A seguir, será detalhada a etapa final dos painéis gráficos que exibem os componentes de temperatura, ECG e BPM.

4.9.1. Consulta de dados armazenados no BD (Temperatura e ECG)

Os gráficos iniciam vazios até que um período de tempo seja selecionado nas funções “Início em:”, “Fim em:” e o botão “Mapear” seja pressionado (Figura 13). Como o gráfico de ECG contém um volume elevado de dados, ele está em uma aba separada, acessível pelo botão “Eletrocardiograma”, com outra opção para visualizar gráficos em tempo real, tanto na aba de temperatura quanto na de ECG.

Figura 13 – Tela de exibição do painel de gráfico de temperatura.



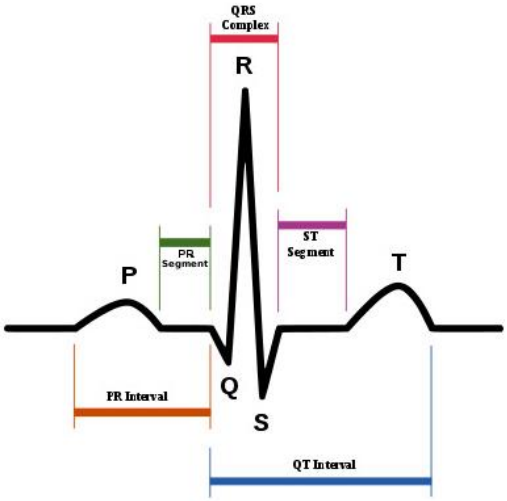
Fonte: Autor.

O fluxo de dados para visualização dos gráficos de temperatura e ECG compartilha similaridades na codificação, como a consulta por intervalo de tempo no BD e o processamento do JSON para exibição sequencial. A principal diferença é que o gráfico de temperatura é mais simples, pois não requer intervalos curtos, já que a temperatura corporal não varia abruptamente. Por outro lado, o gráfico de ECG inclui funções adicionais para identificar picos de batimento e calcular o intervalo entre o primeiro e o último dado.

A detecção de picos é realizada considerando a amplitude máxima das amostras de dados nos complexos QRS que representam a atividade elétrica dos ventrículos durante a ejeção de sangue. Para identificá-los, utilizam-se métodos baseados no desvio padrão, traçando uma "linha imaginária" que varia dinamicamente conforme os dados são amostrados. Para evitar a consideração de picos da onda T

ou picos com amplitudes menores que o padrão do complexo QRS (Figura 14), é aplicada uma margem de 220% sobre o valor do Desvio Padrão.

Figura 14 - Segmentos de ondas de ECG.



Fonte: Anuar Saleh.

O intervalo de tempo é definido em segundos com base na diferença de tempo entre os dois períodos (início e fim) declarados pelo usuário. Uma vez que se tem conhecimento de quantos picos do complexo QRS foram identificados e o intervalo de tempo, é possível estimar o BPM de acordo com a equação a seguir.

Equação para cálculo de BPM.

$$\text{BPM} = \left(\frac{\text{Quantidade de picos detectados}}{\text{Intervalo de tempo em segundos}}\right) \text{x } 60$$

A aplicação verifica a sessão ativa, permitindo a conexão com o BD apenas se o usuário estiver autenticado. Após a validação, tenta-se estabelecer a conexão com o BD, notificando o usuário em caso de falha, como problemas de rede ou alteração no diretório do BD.

Com a conexão estabelecida, o usuário define o intervalo de tempo a ser analisado. Os dados são extraídos da tabela, organizados e indexados. A função nativa “split” é usada para separar os dados com base na vírgula como delimitador. Após a organização, a aplicação calcula o intervalo entre o primeiro e o último dado, usando a coluna “timestamp” da tabela para registrar o momento de inserção dos dados. A conexão é então encerrada, e os dados são exibidos no gráfico, com o eixo X representando o tempo e o eixo Y mostrando a amplitude ao longo do tempo (ver

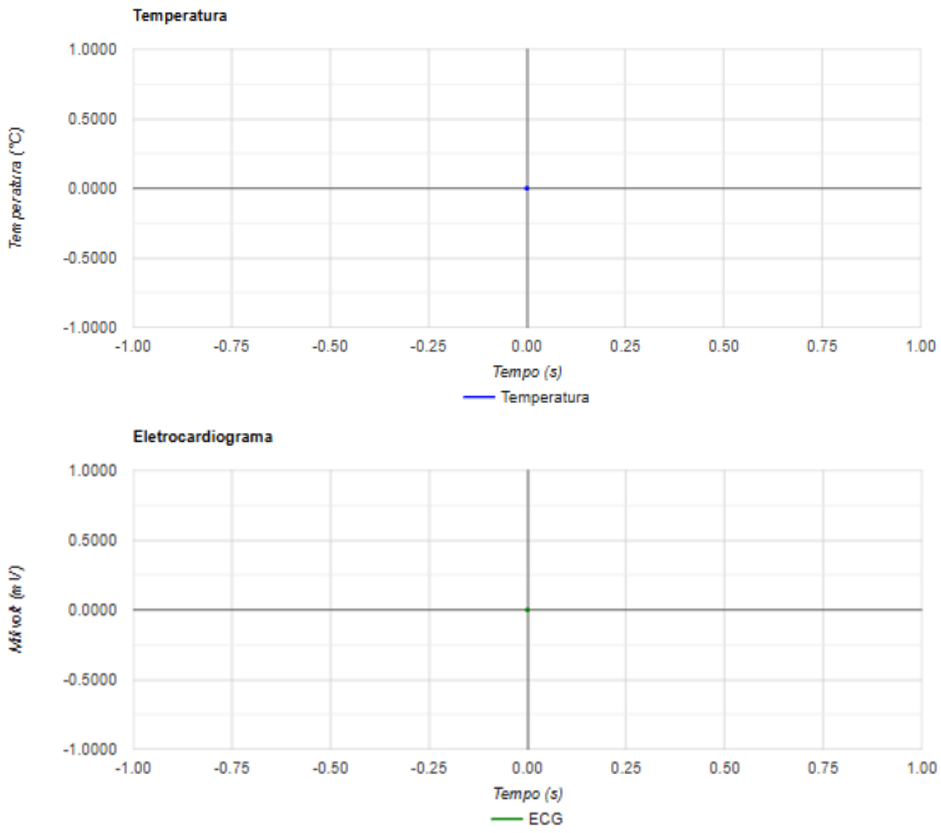
APÊNDICE D – Fluxograma de consultar dados de temperatura e ECG armazenados no BD).

Os gráficos são limitados para evitar sobrecarga visual: o gráfico de temperatura suporta até 100 dados, e o de ECG até 2 mil. Quando esses limites são excedidos, novos blocos de gráficos são gerados em sequência, com o próximo bloco continuando de onde o anterior parou. Se o último bloco de ECG não tiver dados suficientes para completar os 2 mil, ele é preenchido com zeros, mantendo a proporcionalidade com os demais blocos.

4.9.2. Dados em tempo real via MQTT

Para reforçar o conceito de IoT no programa, que demanda execução de tarefas em tempo real, a Versão 1 da aplicação foi reimplementada com algumas melhorias. Inicialmente, gráficos e informações são exibidos vazios (Figura 15), até que os primeiros dados sejam recebidos. Os gráficos operam de forma independente, cada um assinando tópicos específicos, e o ESP8266 envia dados de temperatura em um intervalo de tempo mais espaçado em comparação ao ECG, sendo atualizado somente ao receber novos dados.

Figura 15 – Página de exibição de temperatura, BPM e ECG em tempo real via MQTT.



Fonte: Autor.

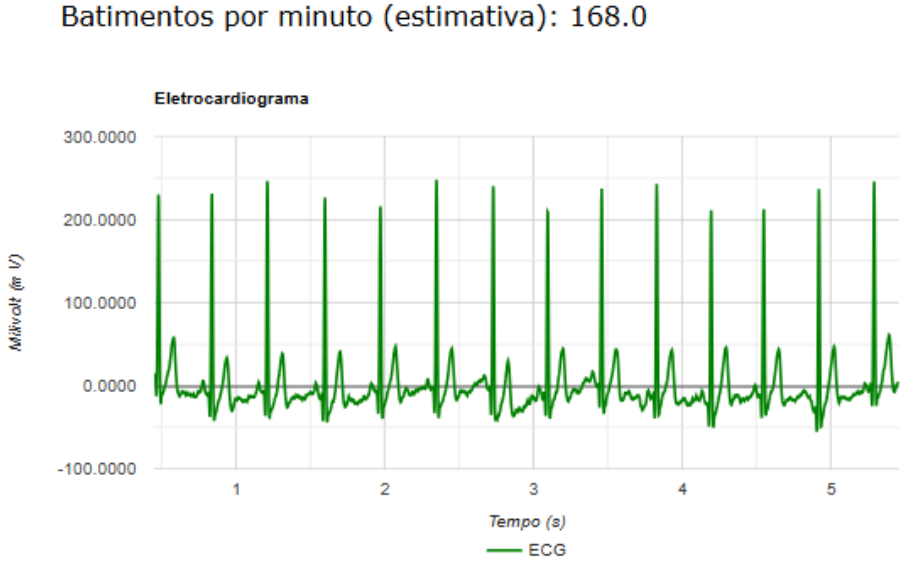
Ao acessar a página de monitoramento em tempo real, a aplicação verifica a sessão ativa e importa as bibliotecas necessárias, incluindo o modelo de gráfico do Google Charts (CHARTS, 2024) e a biblioteca Paho para MQTT. Inicialmente, os gráficos e informações são carregados vazios.

A função MQTTConnect() estabelece a conexão com o broker via WebSockets, configurando as variáveis globais de MQTT, como o host (test.mosquitto.org), a porta (1883) e os tópicos para ECG (eletrocardiograma1883) e temperatura (temperatura1883).

A exibição do ECG segue a mesma lógica da temperatura, com um limite de 1.000 amostras por gráfico. Ao atingir esse limite, os dados mais antigos são removidos para garantir uma visualização limpa, evitando sobrecarga de informações. Considerando que o ESP8266 coleta um dado a cada 2ms e envia pacotes de 25 dados a cada 50ms, a página recebe cerca de 500 dados de ECG por segundo.

A identificação dos picos de ECG é feita conforme o método descrito no tópico 4.9.1, mas com a consideração do intervalo de 2ms entre as amostras. Para estimar o BPM, utiliza-se um intervalo de 5 segundos (2.500 amostras). O número de picos identificados nesse intervalo, através do desvio padrão, é aplicado na equação de BPM (mencionada no tópico 4.9.1) para calcula-la, que é então atualizado no gráfico. A Figura 16 ilustra um exemplo transmitido via MQTT através do ESP8266.

Figura 16 - Exemplo de ECG em tempo real. Captura de 14 picos de ECG em um intervalo de 5 segundos resultando em uma estimativa de 168 BPM.



Fonte: Autor.

Sendo assim, com todos os cálculos concluídos e os vetores preenchidos com as amostras, os gráficos e as informações de BPM são atualizados continuamente até que as mensagens deixem de ser recebidas ou o usuário mude de página (ver APÊNDICE G – Fluxograma de leitura de temperatura e ECG em tempo real via MQTT).

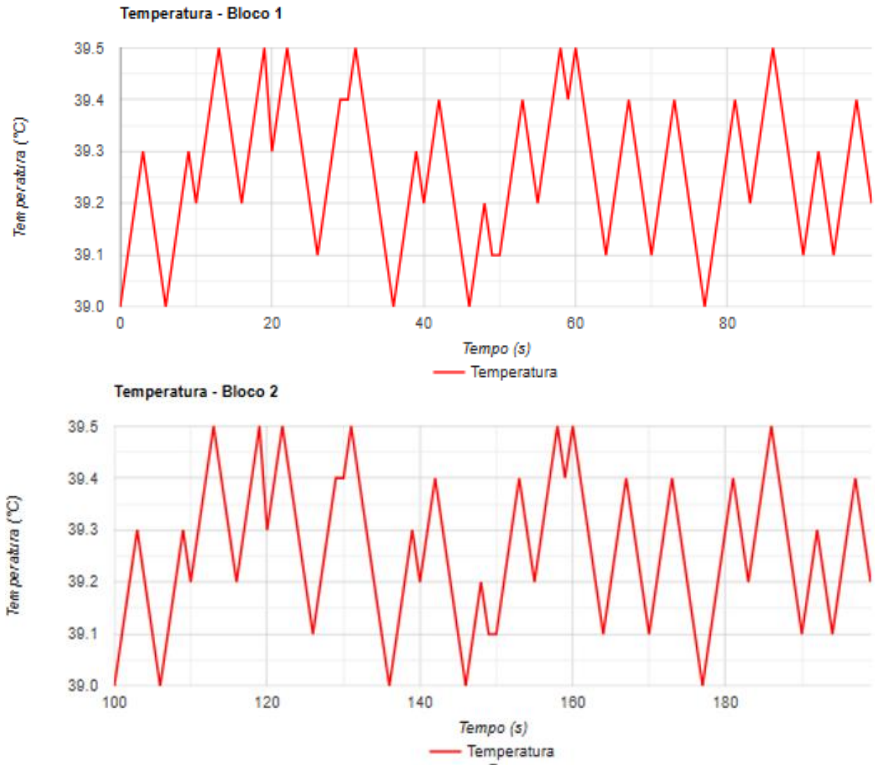
Toda a codificação do projeto está disponível no APÊNDICE A – Código do projeto.

5. RESULTADOS E DISCUSSÕES

Para demonstrar o funcionamento final do projeto, foi criado um ambiente de teste com um usuário fictício no banco de dados, autenticado na plataforma. Esse usuário pode gerenciar suas credenciais e visualizar gráficos de temperatura, ECG e BPM em tempo real, ou consultar dados históricos por meio de painéis interativos.

As tabelas de temperatura e ECG foram inicialmente populadas com dados de exemplo do Physio Zoo (2023), representando a temperatura corporal de um coelho. Esses dados foram organizados em planilhas Excel e carregados no banco de dados usando uma aplicação Python com a biblioteca Pandas, simulando consultas e apresentação de dados conforme o intervalo selecionado (Figura 17).

Figura 17 – Gráficos de temperatura carregados do BD simulando a temperatura corporal de um coelho.

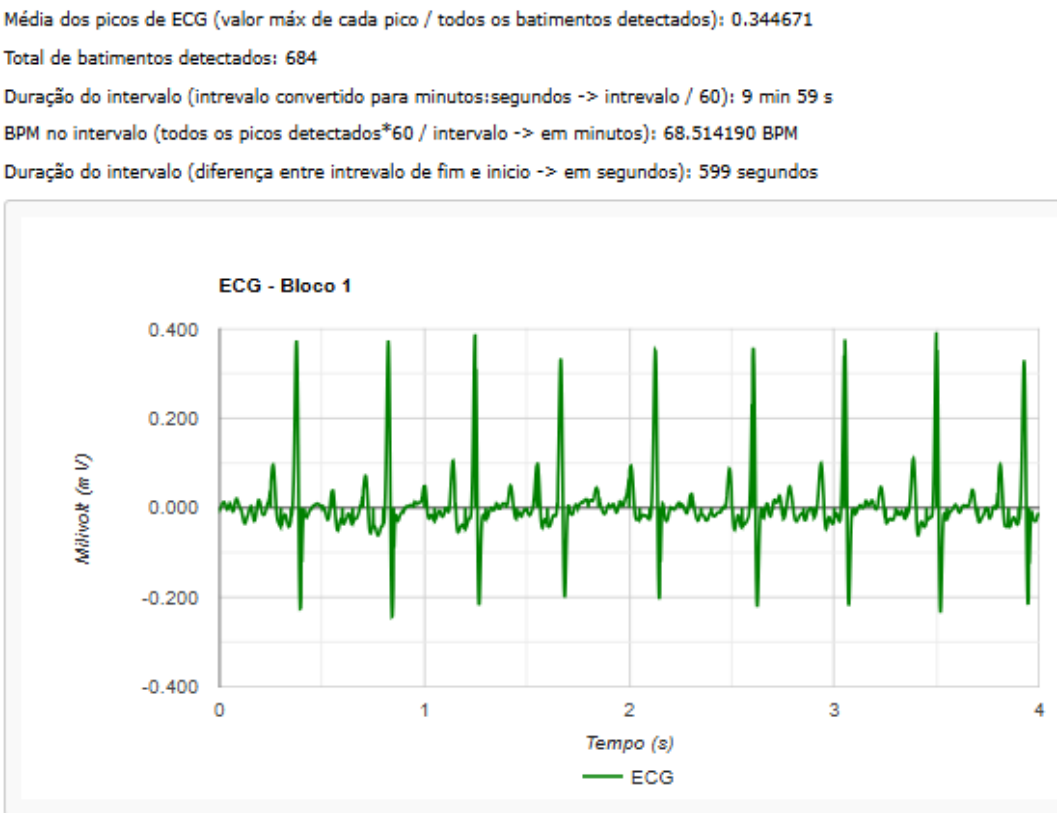


Fonte: Autor.

De acordo com Harkness & Wagner (1977), a temperatura corporal de um coelho em condições normais de conforto é aproximadamente 39°C. Os gráficos gerados refletem de forma precisa os dados registrados na tabela de temperatura, apresentando uma leve oscilação entre 39°C e 39,5°C, com cada ponto visualizável de maneira clara. Os dados são agrupados em blocos, com cada um contendo um limite de 100 amostras, garantindo uma exibição controlada e limpa.

Em relação aos dados de ECG, foram consultadas mais de 130 mil amostras, sendo que cada bloco gráfico exibe até 2 mil pontos (Figura 18). Informações adicionais sobre os dados de ECG são apresentadas acima dos gráficos, incluindo a média da amplitude dos picos, o número de batimentos identificados (599 batimentos no intervalo consultado), a duração do intervalo em segundos e minutos, e a frequência de batimentos por minuto (BPM).

Figura 18 – Gráficos de ECG carregados do BD, com informações calculadas de BPM.



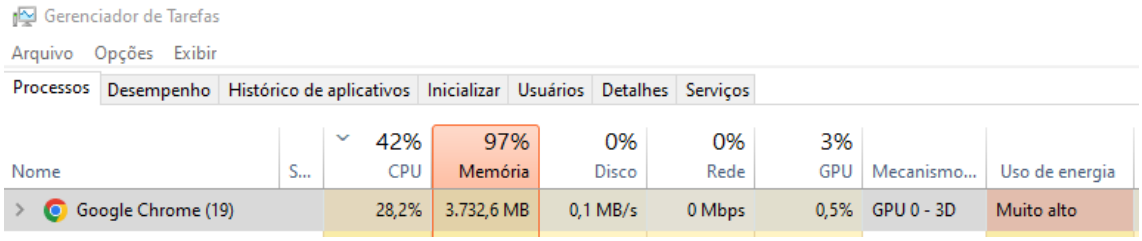
Fonte: Autor.

É importante destacar que os exemplos de ECG utilizados do Physio Zoo estão em sua forma bruta, ou seja, não foram normalizados ou suavizados, o que significa que os dados exibem variações naturais da amplitude e dos picos, sem qualquer tratamento para reduzir o ruído ou suavizar as flutuações.

Durante as consultas de gráficos de ECG no banco de dados, que envolvem grandes volumes de dados, foi observada uma lentidão no navegador. Para realizar uma análise comparativa, o comportamento do navegador foi monitorado ao carregar diferentes volumes de dados.

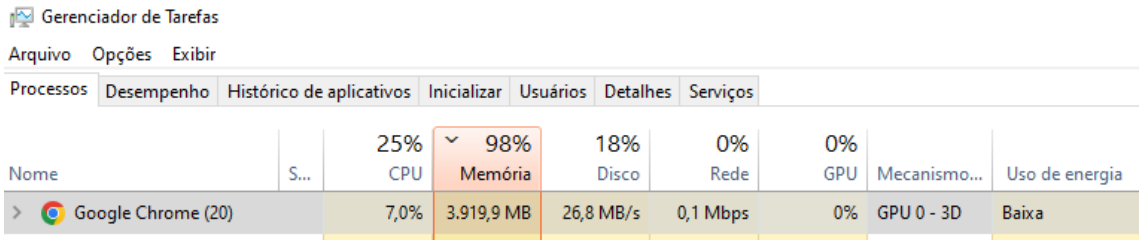
Na Figura 19, é possível observar que a carga computacional é significativamente mais intensa, sendo aproximadamente quatro vezes maior do que a observada na Figura 20, quando uma quantidade menor de dados é carregada.

Figura 19 – Comportamento do navegador no gereciador de tarefas enquanto carrega os dados de ECG.



Fonte: Autor.

Figura 20 – Comportamento do navegador após a leitura ser finalizada.



Fonte: Autor.

Um desafio significativo foi o processamento simultâneo de grandes volumes de dados para consultas históricas e visualização em tempo real, o que causou atrasos devido ao congestionamento de dados. Uma solução potencial seria otimizar o agrupamento e o envio dos dados, a fim de balancear a carga e melhorar a performance.

É possível visualizar a leitura em tempo real de ECG no vídeo anexado (ver APÊNDICE B – Vídeo de leitura de ECG em tempo real via MQTT). Os dados apresentados correspondem a leituras reais, processadas pelo ESP8266. Equipamentos mais avançados, usados em ambientes laboratoriais, identificam picos com base na morfologia do sinal, utilizando tendências de subida e descida.

No entanto, esse método não pôde ser implementado neste projeto, pois o sinal gerado pelo ESP8266 é altamente afetado por ruídos, o que compromete a suavização do sinal. Como resultado, mudanças bruscas no sinal, que não seguem

padrões claros, podem ser interpretadas erroneamente pela plataforma como segmentos de onda, dificultando a precisão na identificação de picos.

Para a visualização em tempo real via MQTT, a conexão foi bem-sucedida. Basta que o cliente assine os tópicos de conexão com o ESP8266 e envie os dados no formato JSON, como exemplificado:

{"v": "-0.006394,-0.003957,-0.001813,0.000000,0.002456,0.004581,0.007037,0.009162,0.010390,0.011910,0.012515,0.012827,0.011910,0.010390,0.008246,0.005809,0.003060,0.000000,-0.002125,-0.003353,-0.002125,-0.000292,0.002144,0.004581,0.006413"}

Embora haja um atraso na atualização gráfica devido ao processamento de dados na página, a experiência do usuário não é comprometida. Para um profissional veterinário, a capacidade de visualizar informações de ECG e temperatura em tempo real, com a possibilidade de compartilhá-las instantaneamente com outros pesquisadores, independentemente da localização geográfica, amplia significativamente a utilidade da ferramenta.

Os dados de BPM são processados com base em um intervalo de 5 segundos, que é necessário para aplicar as equações de cálculo apropriadas. Durante esse intervalo inicial, as informações de BPM podem não ser totalmente precisas devido à quantidade limitada de amostras, mas elas se ajustam rapidamente, proporcionando uma estimativa confiável.

Idealmente, o BPM deveria considerar a quantidade de picos de ECG dentro de um intervalo de 60 segundos, o que resultaria em um valor preciso, em vez de uma estimativa. No entanto, para analisar um intervalo de 1 minuto, o *array* que armazena as amostras seria sobrecarregado com aproximadamente 30 mil dados, o que causaria uma redução significativa na taxa de atualização da página em tempo real.

O fluxo geral do projeto segue uma abordagem semelhante à descrita por Yew et al. (2020) no artigo "IoT-based low-cost distant patient ECG monitoring system", com a adaptação para uma plataforma web.

Neste projeto, o módulo ESP8266 envia os dados coletados e processados para um broker MQTT, de onde são direcionados tanto para o banco de dados quanto para a plataforma web. Através dessa plataforma, é possível consultar o histórico dos dados armazenados, completando assim o ciclo de monitoramento em tempo real e consulta retrospectiva.

6. CONCLUSÃO

Este trabalho proporcionou um enriquecimento significativo para a formação profissional, ao integrar o aprendizado teórico com a aplicação prática de ferramentas de computação em nuvem e Internet das Coisas (IoT). Durante o desenvolvimento do sistema proposto, foi possível consolidar conhecimentos adquiridos ao longo da graduação, resultando em um projeto que exemplifica a convergência entre tecnologia e monitoramento de saúde animal.

A implementação nos servidores da rede da UFPA, inicialmente prevista, foi adiada devido a trâmites burocráticos. Como alternativa, o sistema de banco de dados MySQL foi configurado localmente, permitindo a simulação da execução do sistema.

Os testes realizados, que replicaram cenários reais, mostraram-se eficazes, especialmente na exibição gráfica de dados de temperatura e ECG, tanto em tempo real quanto em períodos específicos de consulta.

Os resultados obtidos confirmam a viabilidade do sistema em situações práticas, permitindo que sinais vitais de animais sejam coletados e analisados instantaneamente por médicos veterinários, seja de forma individual ou colaborativa.

A plataforma não apenas contribui para a otimização de práticas clínicas, mas também possibilita a integração entre saúde e tecnologia, oferecendo uma solução economicamente acessível. O uso de sensores e do módulo ESP8266, que apresentam custos significativamente mais baixos do que equipamentos laboratoriais sofisticados, garantiu medições com precisão adequada para os objetivos do estudo.

Para trabalhos futuros, sugere-se a expansão da base de dados por meio da implementação do sistema junto a um grupo de médicos veterinários do Instituto de Medicina Veterinária de Castanhal. Esses profissionais poderiam testar a usabilidade da plataforma ao longo dos próximos meses, coletando amostras de animais com o ESP8266 e gerenciando as informações na plataforma.

Até o momento da redação deste trabalho, os dados utilizados provinham do Physio Zoo e de testes com humanos. A próxima fase envolveria a implementação real do sistema com animais.

Outro aspecto relevante que pode ser explorado é a otimização da codificação do projeto. A adoção de métodos mais eficientes, modelos de consulta mais ágeis ou ferramentas mais sofisticadas poderia melhorar o desempenho do sistema, reduzindo

o custo computacional e garantindo uma experiência de usuário mais fluida, conforme os feedbacks obtidos nos testes de usabilidade.

A implementação de inteligência artificial ou aprendizado de máquina (*Machine Learning*) também se apresenta como uma possibilidade promissora para o sistema. A integração dessas tecnologias permitiria que o sistema previsse condições de saúde adversas com base nos dados vitais coletados, alertando os pesquisadores responsáveis.

Isso não apenas facilitaria a análise e diagnóstico pelos veterinários, mas também contribuiria para a melhoria da qualidade de vida dos animais, possibilitando a detecção precoce de patologias e a prevenção de zoonoses, com impactos significativos na saúde pública e no bem-estar animal.

REFERÊNCIAS

Adafruit IO. Disponível em: <https://io.adafruit.com/ceqsantos/feeds>. Acesso em: 14 out. 2024.

Adafruit IO API Reference. Disponível em: <https://io.adafruit.com/api/docs/#data>. Acesso em: 19 out. 2024.

Amazon WebServices. O queéaInternetdasCoisas(IoT)? 2023. Disponívelem: <https://aws.amazon.com/pt/what-is/iot/>. Acesso em: 10 de Outubro de 2024.

ARMBRUST, M. et al. *A View of Cloud Computing. University of California*, Berkeley, USA, 2010. Disponível em: <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf>. Acesso em: 08 de outubro de 2024.

Amazon Web Services, 2023. O que é MQTT? – Explicação sobre o protocolo MQTT – AWS. Disponível em: <https://aws.amazon.com/pt/what-is/mqtt/>.

BESCHOKOV, M. **What is WebSocket and How It Works?** Disponível em: <https://www.wallarm.com/what/a-simple-explanation-of-what-a-websocket-is>. Acesso em: 28 ago. 2024.

CIA, A. E, 2022. Automação residencial usando MQTT. Disponível em: <https://www.arduinoecia.com.br/automacao-residencial-usando-mqtt-wio-terminal-esp8266/>.

CHARTS. Google for Developers. Disponível em: <https://developers.google.com/chart>. Acesso em: 19 out. 2024.

CLEMENTE, Paulo. 2024. Rocketsat. **WebSocket: O que é e quando usar?**. Diponível em: <https://blog.rocketseat.com.br/websocket-o-que-e-e-quando-usar/>. Acesso em: 28 de agosto de 2024.

CRAGGS, I, 2017. Eclipse Paho | The Eclipse Foundation. Disponível em: <https://eclipse.dev/paho/index.php?page=clients/js/index.php>. Acesso em: 19 out. 2024.

DANKI CODE. Curso de PHP Jedai: Métodos mágicos. Disponível em: <https://cursos.dankicode.com/campus/php-jedai>. Acesso em: 13 out. 2024.

GARCIA, G, 2024. XAMPP: O que é, Como Funciona, Vantagens e Instalação da Ferramenta. Disponível em: <https://mercadoonlinedigital.com/blog/xampp/>.

Getting Started with MQTT Using Python (Esquema de protocolo MQTT). **EMBEDDED LABORATORY**. Disponível em: <http://embeddedlaboratory.blogspot.com/2018/01/getting-started-with-mqtt-using-python.html>. Acesso em: 27 de agosto de 2024.

HADIS, N. S. M. et al. Iot based patient monitoring system using sensors to detect, Analyse and monitor two primary vital signs. Journal of Physics: Conference Series, IOP Publishing, v. 1535, n.1, p. 012004, may 2020. Disponível em: <https://dx.doi.org/10.1088/1742-6596-/1535/1/012004>.

HARKNESS, J. E.; WAGNER, J. E. The biology and medicine of rabbits and rodents. Pennsylvania: Lea & Febger Philadelphia, 1977. 152 p.

How do WebSockets work? (Comparação de conexãoWebSockets vs HTTP). **Wallarm**. Disponível em: <https://www.wallarm.com/what/a-simple-explanation-of-what-a-websocket-is >. Acesso em: 28 de agosto de 2024.

HTML: Linguagem de Marcação de Hipertexto. **MDN WEB DOCS**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Glossary/HTML>. Acesso em: 28 de agosto de 2024.

HTML - Glossário do MDN Web Docs: Definições de termos relacionados à Web | MDN. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Glossary/HTML>. Acesso em: 19 out. 2024.

H. T. Yew, M. F. Ng, S. Z. Ping, S. K. Chung, A. Chekima and J. A. Dargham, "IoT Based Real-Time Remote Patient Monitoring System," 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), Langkawi, Malaysia, 2020, pp. 176-179.

IBM. O que são regras de negócios?. Disponível em: <https://www.ibm.com/br-pt/topics/business-rules>. Acesso em: 13 de outubro de 2024.

INTERNET E COISAS. Curso de MQTT. Disponível em: <https://www.youtube.com/watch?v=WqHIFSYMHy0&list=PLMmiQibT0iTblNNF_y6_xZfvid5LcWK6_>. Acesso em: 19 out. 2024.

LARISSA KICH. Como fazer um MENU LATERAL responsivo (SIDEBAR) | HTML, CSS e JavaScript. Disponível em: <https://www.youtube.com/watch?v=lZVQGjTEX-w>. Acesso em: 21 out. 2024.

LNCC, 2024. Os quatro pilares da segurança da informação – Confidencialidade, Disponibilidade, Integridade e Autenticidade. Disponível em: <https://www.gov.br/lncc/pt-br/centrais-de-conteudo/campanhas-de-conscientizacao/gestao-de-seguranca-da-informacao/os-quatro-pilares-da-seguranca-da-informacao-2013-confidencialidade-disponibilidade-integridade-e-autenticidade>. Acesso em: 14 out. 2024.

NERI, Renan; LOMBA, Matheus; BULHÕES, Gabriel. **Escola politécnica UFRJ**. 2019. <https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/>. Acesso em: 27 de agosto de 2024.

O que é CSS? - Aprendendo desenvolvimento web | MDN. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/CSS/First_steps/What_is_CSS>. Acesso em: 28 de agosto de 2024.

O que é JavaScript? - Aprendendo desenvolvimento web | MDN. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript>. Acesso em: 28 de agosto de 2024.

PAULA, O, M, 2017. IoT: os dispositivos são mesmo inteligentes? - Excelência em Pauta. Disponível em: <https://www.excelenciaempauta.com.br/iot-dispositivos-inteligentes/>. Acesso em: 13 out. 2024.

PFUETZENREITER, Márcia Regina; ZYLBERSZTAJN, Arden; AVILA-PIRES, Fernando Dias de. Evolução histórica da medicina veterinária preventiva e saúde pública. **Ciência Rural**, v. 34, p. 1661-1668, 2004.

PHP: HYPERTEXT PREPROCESSOR. PHP: Informações Gerais - Manual. Disponível em: <https://www.php.net/manual/pt_BR/faq.general.php#faq.general.what. Acesso em: 19 out. 2024.

PHYSIOZOO, 2023. GitHub - physiozoo/Examples. Disponível em: <https://github.com/physiozoo/Examples>. Acesso em: 21 out. 2024.

Programando o ESP8266 pelo Arduino IDE. **RoboCore**. Disponível em: <https://www.robocore.net/tutoriais/programando-o-esp8266-pela-arduinoide?srsltid=AfmBOooj7rC_BQSgBzkQxaE20FyoxqeLvx7NUHRmMVrXALy5Tbr-UdRp>. Acesso em: 28 de agosto de 2024.

P. Singh and A. Jasuja, "IoT based low-cost distant patient ECG monitoring system," 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 2017, pp. 1330-1334, doi: 10.1109/CCAA.2017.8230003.

ROBOCORE. Programando o ESP8266 pela Arduino IDE - RoboCore. Disponível em: <https://www.robocore.net/tutoriais/programando-o-esp8266-pela-arduino-ide?srsltid=AfmBOopdrdukL7Ng9HXN3aAz0Z9SRe75F40B6mHd5FahCiTNlxxB0O15>. Acesso em: 19 out. 2024.

SALEH, A. **Eletrocardiograma normal: a base para o entendimento de qualquer outra alteração**. Disponível em: <https://www.medway.com.br/conteudos/eletrocardiograma-normal-a-base-para-o-entendimento-de-qualquer-outra-alteracao/>.

T. H. Hafsiya and B. Rose, "An IoT-Cloud Based Health Monitoring Wearable Device For Covid Patients," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2021, pp. 266-269.

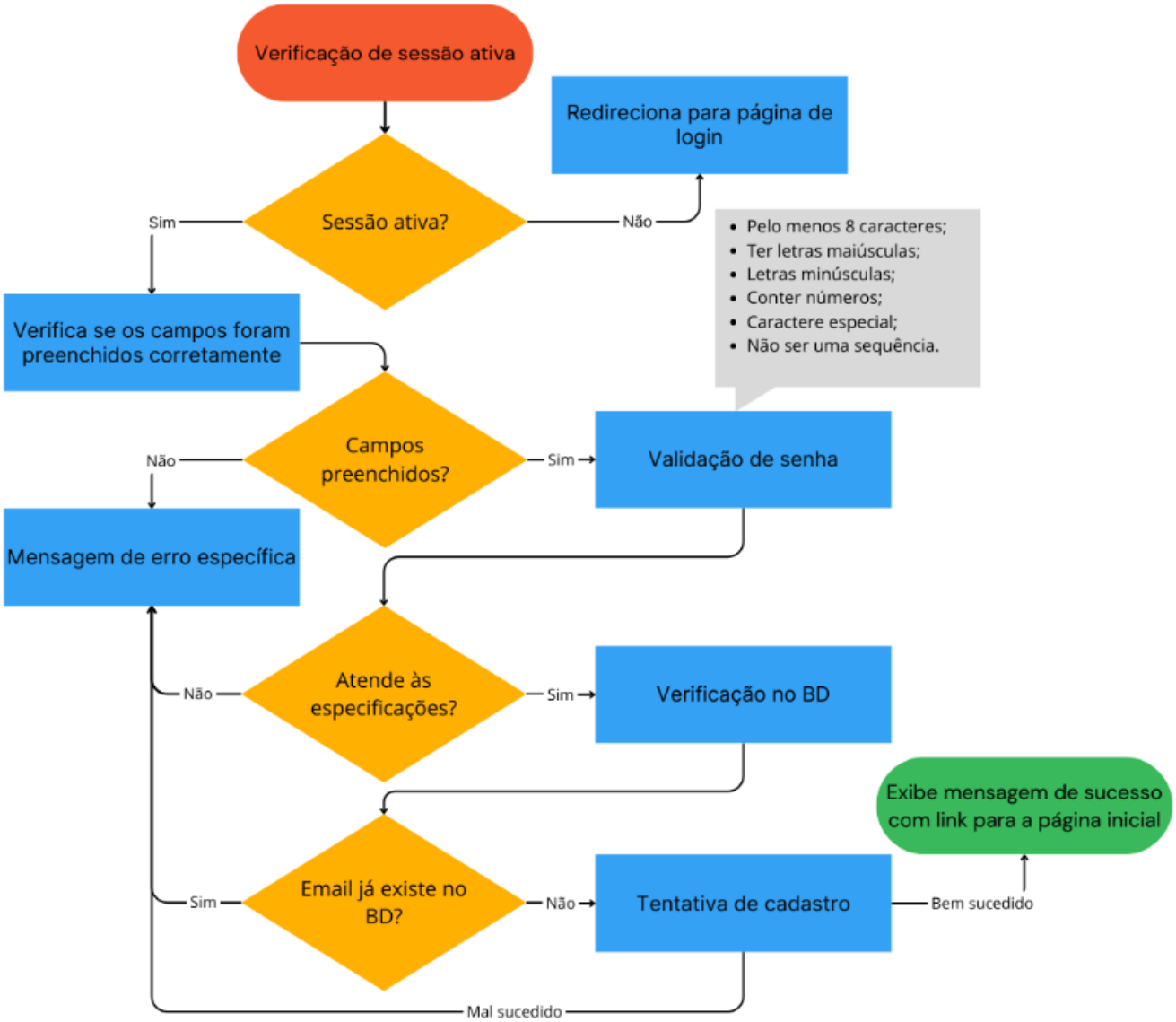
XAMPP Installers and Downloads for Apache Friends. Disponível em: <https://www.apachefriends.org/pt_br/index.html>. Acesso em 19 out. 2024.

APÊNDICES

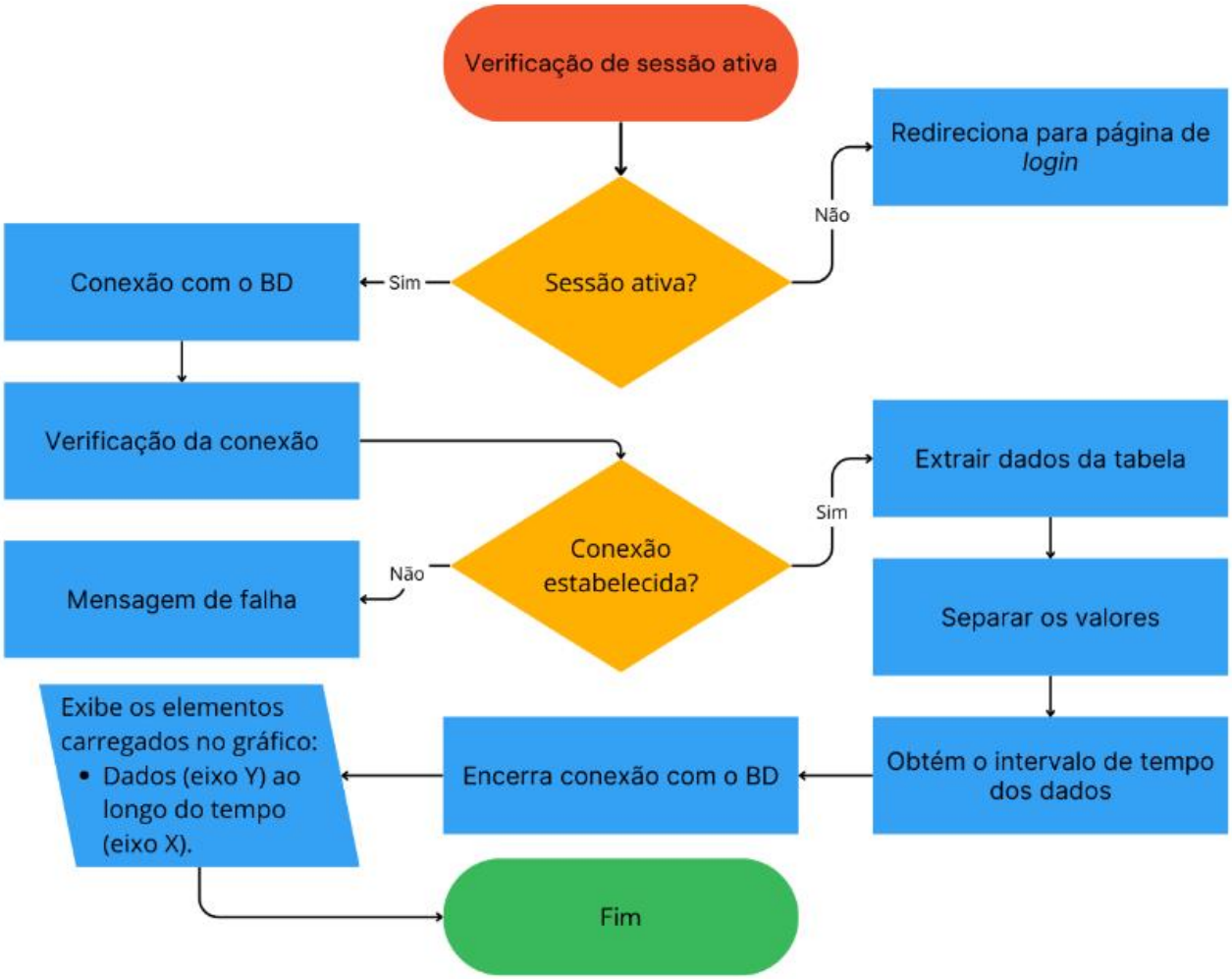
1. APÊNDICE A – Código do projeto
https://drive.google.com/drive/folders/15zXayvN2JmYcTvitqFosrHvwB2GCz67t?usp=drive_link

2. APÊNDICE B – Vídeo de leitura de ECG em tempo real via MQTT
https://drive.google.com/drive/folders/1QrFeFTmRDFKrhjDCWT9ktQFmX4l-U7oR?usp=drive_link

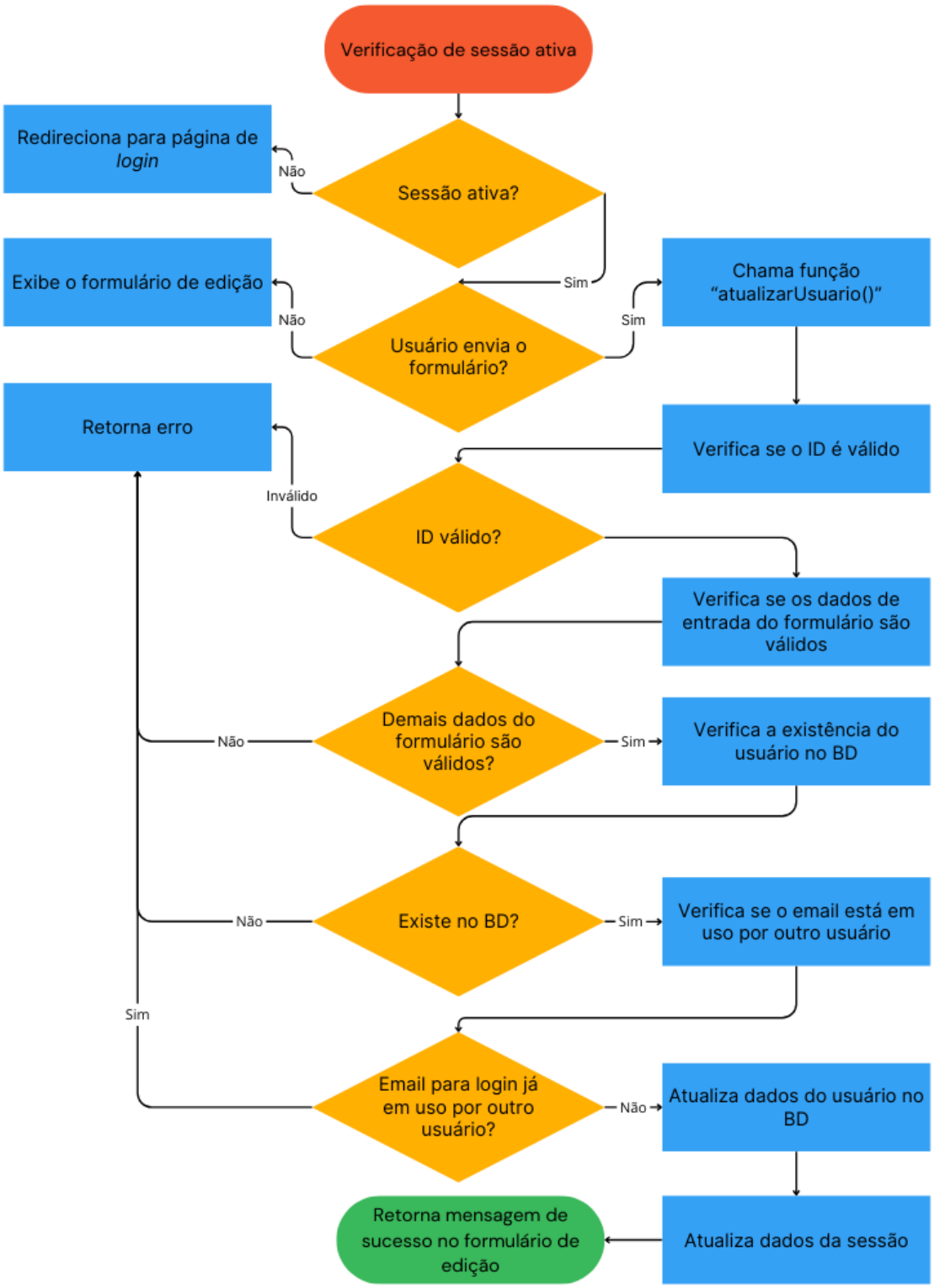
3. APÊNDICE C – Fluxograma de cadastro de usuário



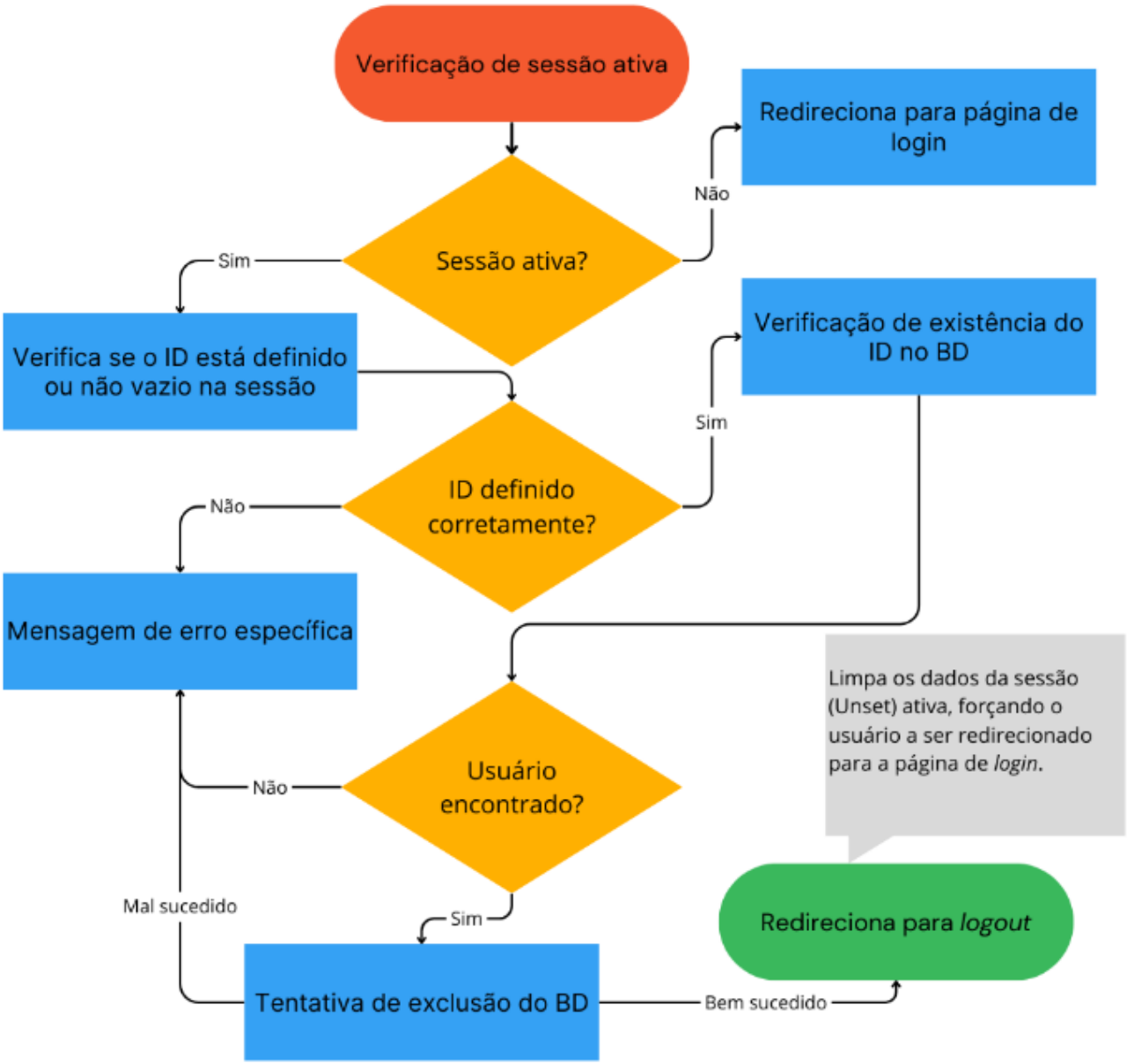
4. APÊNDICE D – Fluxograma de consultar dados de temperatura e ECG armazenados no BD



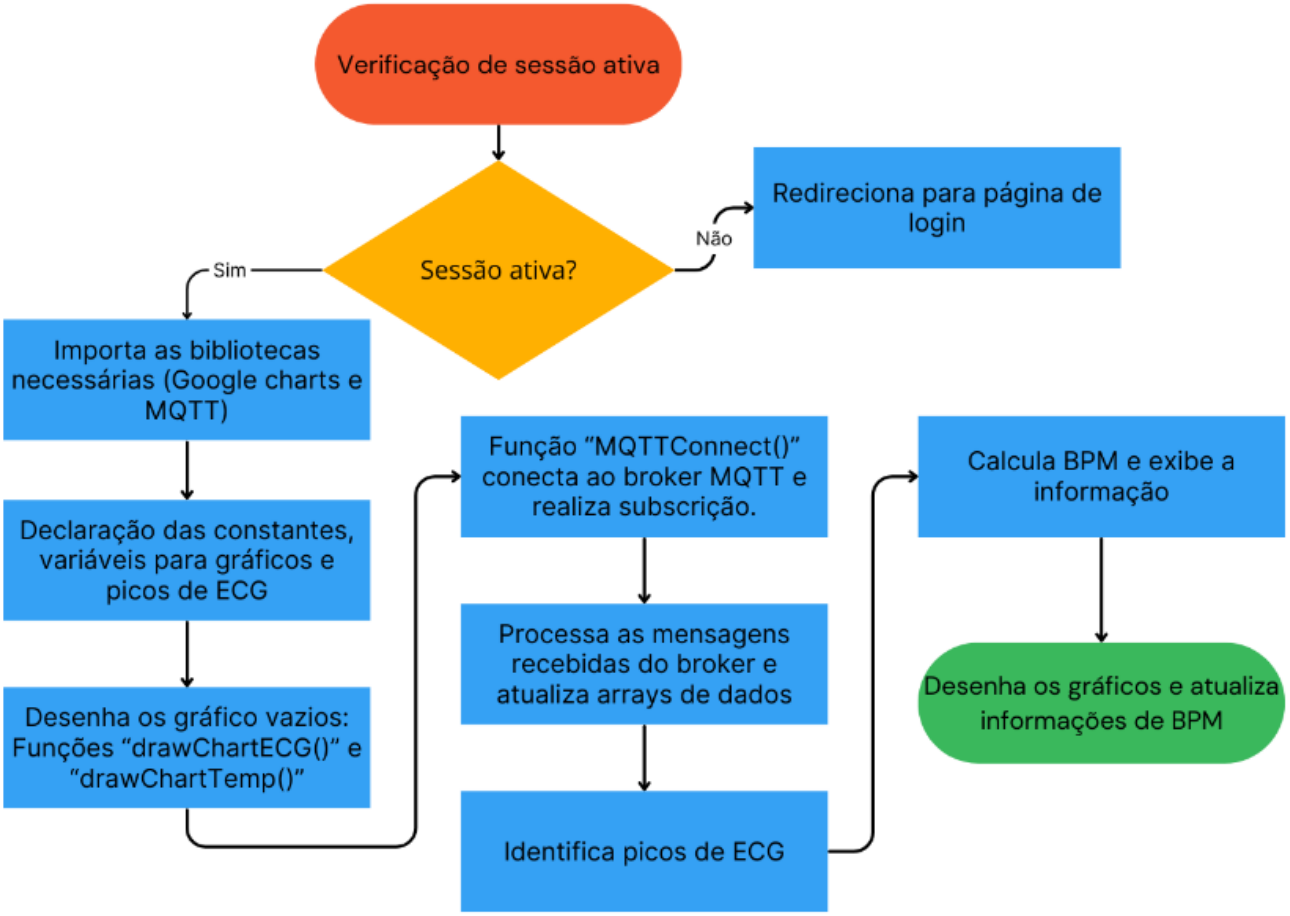
5. APÊNDICE E – Fluxograma de edição de dados do usuário



6. APÊNDICE F – Fluxograma de exclusão do login de acesso



7. APÊNDICE G – Fluxograma de leitura de temperatura e ECG em tempo real via MQTT



8. APÊNDICE H – Fluxograma de login do usuário