UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Luiz Gustavo Nunes Theodorico

DESENVOLVIMENTO DE UM RASTREADOR UTILIZANDO DISPOSITIVOS IOT E INTEGRAÇÃO COM MAPAS

Santa Maria, RS
2023

Luiz Gustavo Nunes Theodorico

DESENVOLVIMENTO DE UM RASTREADOR UTILIZANDO DISPOSITIVOS IOT E INTEGRAÇÃO COM MAPAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação, Área de Concentração em CNPq, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Sistemas de Informação**.

Orientador: Prof. Celio Trois

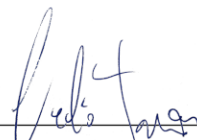
Santa Maria, RS
2023

Luiz Gustavo Nunes Theodorico

DESENVOLVIMENTO DE UM RASTREADOR UTILIZANDO DISPOSITIVOS IOT E INTEGRAÇÃO COM MAPAS

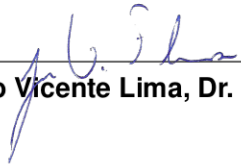
Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação, Área de Concentração em CNPq, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Sistemas de Informação**.

Aprovado em 1° de fevereiro de 2023:



Celio Trois, Dr. (UFSM)
(Presidente/Orientador)

Marcia Pasin, Dra. (UFSM)



João Vicente Lima, Dr. (UFSM)

Santa Maria, RS
2023

RESUMO

DESENVOLVIMENTO DE UM RASTREADOR UTILIZANDO DISPOSITIVOS IOT E INTEGRAÇÃO COM MAPAS

AUTOR: Luiz Gustavo Nunes Theodorico
Orientador: Celio Trois

Atualmente a Internet interliga a tudo e a todos, conectando não apenas computadores e *smartphones*, mas também com dispositivos do cotidiano, recebendo por isso a denominação de Internet das Coisas. Essa evolução possibilitou que os dispositivos estejam interligados com sistemas de rastreamento, seja para segurança, ou para obter a localização atual. Com base neste contexto, o presente trabalho apresenta a metodologia e desenvolvimento de um sistema de rastreamento de código aberto para ser implantado utilizando módulos IoT de baixo custo. Em conjunto com o sistema de rastreamento, também foi desenvolvimento um aplicativo para *smartphone*, que possui como finalidade desenhar a trajetória de pontos geográficos em um mapa, possibilitando uma visualização gráfica do rastreamento dos dispositivos. Para validar a proposta, foi criado um protótipo em bancada conectado à Internet através do módulo integrado Wi-Fi. Neste protótipo, o dispositivo IoT se comunica com o *smartphone* através de um microsserviço desenvolvido em python. A comunicação da placa controladora com este microsserviço ocorre através do protocolo de comunicação MQTT. Nos testes realizados, o protótipo obteve êxito em coletar as coordenadas geográficas e armazená-las em um banco de dados. Uma API, também desenvolvida nesse trabalho, possibilita acesso aos pontos armazenados, de forma que o aplicativo do *smartphone* renderize, mostrando o rastreamento dos dispositivos.

Palavras-chave: IoT, Rastreamento, Microsserviços, React Native, Python

ABSTRACT

DEVELOPMENT OF A TRACKER USING IOT DEVICES AND INTEGRATION WITH MAPS

AUTHOR: Luiz Gustavo Nunes Theodorico
ADVISOR: Celio Trois

Currently, the Internet interconnects everything and everyone, connecting not only computers and smartphones, but also with everyday devices, thus receiving the name Internet of Things. This evolution made it possible for devices to be interconnected to tracking systems, either for security, or to obtain the current location. Based on this context, the present work presents the methodology and development of an open source tracking system to be implemented using low-cost IoT modules. Along with the tracking system, a smartphone application was also developed, which aims to draw the trajectory of geographic points on a map, allowing a graphical visualization of the tracking of devices. To validate the proposal, a bench prototype was created and connected to the Internet through the Wi-Fi integrated module. In this prototype, the IoT device communicates with the smartphone through a microservice developed in python. The communication between the controller board and this microservice takes place via the MQTT communication protocol. In the tests carried out, the prototype was successful in collecting the geographic coordinates and storing them into a database. An API, also developed in this work, allows access to the stored points, so that the smartphone application renders, showing the tracking of the devices.

Keywords: IoT, Tracker, Microservices, React Native, Python

LISTA DE ABREVIATURAS E SIGLAS

API *Application Programming Interface*

APP Aplicativo Móvel

$ASGI$ *Asynchronous Server Gateway Interface*

BD Banco de Dados

GPS *Global Positioning System*

GSM *Global System for Mobile communication*

$HTTP$ *Hypertext Transfer Protocol*

ID Identificação

IoT *Internet of Things*

$JSON$ *JavaScript Object Notation*

$LoRaWAN$ *Long Range Wide Area Networking*

$M2M$ *Machine to Machine*

$MQTT$ *Message Queuing Telemetry Transport*

RF *Radio Frequency*

RX *Receive*

SDK *Software Development Kit*

$SGBD$ Sistema de gerenciamento de Banco de Dados

SMS *Short Message Service*

SQL *Structured Query Language*

TCP *Transmission Control Protocol*

TX *Transmit*

XML *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 7
2 TRABALHOS RELACIONADOS 9
3 DESENVOLVIMENTO E IMPLEMENTAÇÃO 11
3.1 ARQUITETURA DO SISTEMA DE RASTREAMENTO 11
3.2 DISPOSITIVO IOT E MÓDULO GPS 12
3.3 COMUNICAÇÃO IOT 13
3.4 RECEPÇÃO E CONSUMO DE DADOS 14
3.4.1 Módulo de Recepção de Dados 14
3.4.2 Repositório de Dados 14
3.4.3 API para Consumo dos Dados 15
3.4.4 Arquitetura em Microsserviços 15
3.4.5 Hospedagem 17
3.5 APLICATIVO MÓVEL 18
4 CONCLUSÃO 21
REFERÊNCIAS BIBLIOGRÁFICAS 22

1 INTRODUÇÃO

Atualmente, vivemos em um século marcado por um *boom* tecnológico, onde é possível fazer com que a tecnologia chegue rapidamente às pessoas (MCELROY, 2018). Em virtude disso, ampliaram-se as possibilidades de criar soluções para problemas e desafios encontrados no dia a dia, tais como ferramentas de trabalho, comunicação, entre outros. Processos estes que antes eram realizados de forma manual, ou até mesmo impossíveis de serem realizados manualmente, passam a ser computadorizados e entregarem resultados precisos de forma ágil (AL-SAQQA et al., 2014).

Há processos e equipamentos, que devido ao alto custo e falta de portabilidade, são acessíveis a apenas um nicho específico de público. Porém, pesquisadores estão conseguindo propor soluções mais baratas e portáveis, especialmente devido ao desenvolvimento de dispositivos e tecnologias IoT (CIUFFOLETTI, 2018). Este termo inicialmente apresentado em 1999 por Kevin Ashton, onde ele propunha a possibilidade de utilizar dados coletados sem intervenção humana para aprimorar atividades rotineiras das pessoas (ASHTON et al., 2009). O termo IoT então foi reconhecido pela União Europeia em 2008, tendo como definição a interligação de dispositivos via comunicação sem fio (WEBER; WEBER, 2010).

Os dispositivos IoT tem como finalidade coletar dados por meio de sensores e compartilhar os valores obtidos com outros dispositivos, para então resultar em uma solução mais completa e precisa. Atualmente os dispositivos IoT possuem como foco a automação de processos e dados que não são possíveis coletar manualmente. Como por exemplo, realizar a automação de uma casa por meio de sensores interligados que controlam a incidência de luz, umidade do ar, etc (PUJARIA et al., 2020).

Um outro exemplo de aplicação dos dispositivos IoT são para rastreamento, onde realizam coletas de coordenadas geográficas via GPS e compartilham estes dados com dispositivos remotos (COULBY et al., 2021). Permitindo assim que estes rastreadores sejam aplicados a qualquer objeto móvel, gerando uma maior segurança, visto que vivemos em um mundo onde há um alto índice de criminalidade em função de furtos e roubos de dispositivos eletrônicos e outros bens pessoais. Tendo em vista isto, há algumas soluções responsáveis por prover recursos de geolocalização remota (SINGH; KUMARI, 2019; SAMSON; DHAKSHYANI; ABDULLA, 2020; PATII; IYER, 2017). Essas soluções, por sua vez, possuem custo variado de implantação e recorrência mensal, assim como requerem de um conhecimento técnico elevado para realizar a instalação.

Este trabalho apresenta um sistema de rastreamento usando dispositivos IoT de baixo custo. Para execução do trabalho, primeiramente foi realizado uma pesquisa tecnológica a fim identificar quais tecnologias seriam necessárias para a implementação do mesmo. Em seguida, definiu-se que os dispositivos deveriam possuir um baixo custo, pos-

sibilidade de conexão com a Internet e fácil implantação.

A proposta implementa um módulo controlador que recebe coordenadas geográficas do módulo GPS e as envia para um servidor através de um serviço de mensagens. No lado do servidor, há uma aplicação *Backend* desenvolvida em *python* que é responsável por receber e tratar cada mensagem recebida, inserindo-as em um banco de dados. Por fim, uma API exporta os dados para um aplicativo móvel para ser utilizado no *smartphone*, que desenha a trajetória dos pontos geográficos, mostrando-os em um mapa. É importante ressaltar que toda a arquitetura foi desenvolvida usando o conceito de microsserviços, e implementadas através da plataforma Docker (ANDERSON, 2015).

Para validar a proposta, foi implementado um protótipo que utiliza um micro controlador em conjunto com módulo de localização GPS para obter as coordenadas geográficas. Utilizando o serviço de mensageria MQTT (HUNKELER; TRUONG; STANFORD-CLARK, 2008), os dados são enviados para um servidor e armazenados em um Banco de Dados PostgreSQL. Após isto, um aplicativo móvel implementado em React consume os dados do banco de dados através de uma API, desenvolvida com a biblioteca FastAPI do python, para então renderizar um mapa, usando o SDK do Google Maps para desenhar a trajetória dos pontos geográficos coletados.

O presente trabalho está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados, uma breve descrição de trabalhos que envolvem o mesmo tema escolhido para este trabalho. A Seção 3 mostra a arquitetura proposta, o desenvolvimento e a implementação do protótipo, descrevendo os algoritmos criados e tecnologias utilizadas. Por fim, a Seção 4 apresenta as conclusões e trabalhos futuros.

2 TRABALHOS RELACIONADOS

Esta seção apresenta trabalhos relacionados ao tema, nos quais microcontroladores IoT são usados para obter coordenadas geográficas, propondo algum tipo de sistema para rastreamento.

No trabalho de (SINGH; KUMARI, 2019) foi implementado um sistema anti-furto veícular utilizando Arduino e os módulos GSM e GPS. O sofware desenvolvido pelo autor tem como finalidade monitorar partidas indevidas no motor do veículo. Por meio de troca de mensagens utilizando o módulo GSM é possível controlar que ações o Arduino deve tomar, que seria bloquear ou permitir o funcionamento do veículo. As coordenadas obtidas pelo referido sistema são armazenadas no celular destino ao receber as mensagens, para então serem plotadas manualmente no Google Maps a fim de visualizar a rota exata do veículo.

Em (SAMSON; DHAKSHYANI; ABDULLA, 2020), os autores destacaram sobre a importância do conteúdo de uma carteira na vida de uma pessoa, e também apresentam pesquisas onde é demonstrado que é o bem mais perdido e/ou extraviado. Com isto, os autores implementaram um sistema de rastreio em tempo real utilizando dispositivos IoT. Para conexão e comunicação do dispositivo foi utilizado o protocolo LoRaWAN, na qual realiza conexão Wi-Fi com os dispositivos próximos. Ainda no mesmo trabalho, o autor realizou uma integração com o Google Maps para recebimento das coordenadas geográficas, utilizando o software Ubidot para exibição do mapa com os respectivos pontos geográficos.

No trabalho de (PATII; IYER, 2017), os autores identificaram que durante uma guerra muitos soldados vem a óbito e demonstraram que é possível reduzir este número com um acompanhamento em tempo real. Para isto, os autores implementaram um sistema que conta com placa controladora, módulo Wi-Fi, sensores IoT e módulo de localização GPS. Com isto se torna possível coletar informações de saúde, assim como localização geográfica de cada soldado, compartilhando assim para a sala de controle possuir mais informações para apoio.

Já no trabalho (RAY; DASH; DE, 2019), foi implementado um sistema que integra diversos dispositivos e sensores IoT para monitorar e apoiar pacientes com demência. O sistema consiste em sensores que identificam passos e localização geográfica, se comunicando via RF com o *smartphone*. Neste trabalho, um aplicativo no *smartphone* é responsável por realizar o envio dos dados coletados, não necessitando um módulo IoT de conectividade.

Trabalhos	Este Trabalho	(SINGH; KUMARI, 2019)	(SAMSON; DHAKSHYANI; ABDULLA, 2020)	(PATII; IYER, 2017)	(RAY; DASH; DE, 2019)
Tipo	rastreador	anti-furto	rastreamento em tempo real	monitoramento de saúde	rastreamento pessoas com demência
Dispositivo	ESP32	Arduino	Arduino ESP8266	Arduino UNO	Sensores IoT
Conexão	Wi-Fi	GSM	LoRaWAN	LoRaWAN	rede do *smartphone*
Módulo	GPS	GPS	GPS	GPS e sensores	integrado com outros gadgets
Troca de mensagem	MQTT	SMS	MQTT	SMS/E-mail	SMS/E-mail
Armazenamento	sim	caixa SMS	sim, integrador	não	sim, integrador
Integração com mapas	sim	coordenadas adicionadas manual-mente no google maps	sim, integrador	não	sim, integrador

Tabela 1 – Comparação de trabalhos

Conforme visualizado na Tabela 1, percebe-se que na literatura existem trabalhos que envolvem o tema de rastreamento de diversos tipos, tanto projetos com um simples envio de coordenadas via SMS, quanto a utilização de protocolos de rede para tratativas em outros sistemas. O presente trabalho, descrito em detalhes na Seção 3, possui um diferencial dos demais, na qual um software desenvolvido em python recebe as coordenadas geográficas via protocolo MQTT e as armazena em um servidor PostgreSQL. Tendo assim o histórico dos dados disponíveis a longo prazo para consumo de outros serviços, tal como o APP desenvolvido para desenhar a trajetória das coordenadas em um mapa.

3 DESENVOLVIMENTO E IMPLEMENTAÇÃO

Esta seção apresenta a solução proposta e a metodologia de desenvolvimento adotados na implementação do projeto. Primeiramente definiu-se uma arquitetura modular para o desenvolvimento do projeto. Em um segundo momento, foi realizado uma pesquisa para definir as bibliotecas e tecnologias utilizadas no desenvolvimento de um protótipo. Na terceira etapa, foi desenvolvido um aplicativo para *smartphone*, integrado com um mapa, que permite mostrar a trajetória dos pontos geográficos previamente obtidos. Por fim, realizou-se testes para comprovação do correto funcionamento da proposta.

3.1 ARQUITETURA DO SISTEMA DE RASTREAMENTO

O presente trabalho apresenta um sistema de rastreamento de baixo custo, que utiliza um microcontrolador com módulo de localização GPS para obter as coordenadas geográficas, um módulo de conectividade com Internet e um aplicativo móvel para exibição dos valores previamente obtidos, integrando esses dados com um sistema de mapas onde são desenhadas as trajetórias dos pontos geográficos coletados.

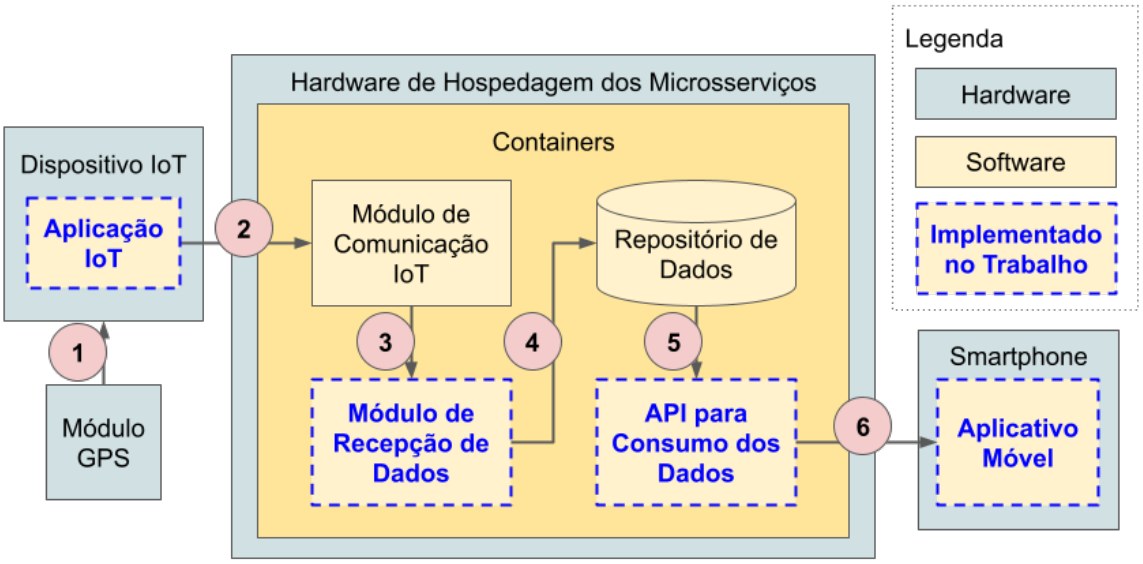


Figura 1 – Arquitetura do sistema proposto

Na Figura 1 é possível visualizar a arquitetura do sistema de rastreamento, composta por módulos operando de maneira isolada, visando assim a construção de um sistema modular. O funcionamento da arquitetura será explicado seguindo o fluxo de dados representado na figura com círculos enumerados:

1. Primeiramente o Dispositivo IoT obtém as coordenadas geográficas (latitude e longitude) do Módulo GPS;
2. O Dispositivo IoT envia as coordenadas juntamente com um identificador único para o Módulo de Comunicação IoT;
3. O Módulo de Comunicação IoT é responsável por oferecer mecanismos de comunicação entre o Dispositivo IoT e o Módulo de Recepção de dados;
4. O Módulo de Recepção de Dados trata as coordenadas recebidas e armazena as mesmas no Repositório de Dados;
5. A API para Consumo dos Dados possui conexão com o Repositório de Dados e expõe rotas para consumo dos dados;
6. Por fim, o Aplicativo Móvel realiza requisições HTTP na API para Consumo dos Dados para obter as coordenadas geográficas que estão armazenadas no Repositório de Dados.

Para o desenvolvimento dos módulos Comunicação IoT e Recepção de Dados, Repositório de Dados e API para Consumo dos Dados será criada uma arquitetura de microsserviços (NADAREISHVILI et al., 2016). As seções a seguir apresentam as etapas realizadas para o desenvolvimento da arquitetura proposta, contendo a definição das tecnologias, as bibliotecas e linguagens escolhidas e descrevendo a implementação e comunicação entre os módulos.

3.2 DISPOSITIVO IOT E MÓDULO GPS

O Dispositivo IoT é responsável por receber os dados de localização e encaminhar para o Módulo de Comunicação IoT. Para tal, devido ao poder de processamento *dual-core* e Wi-Fi integrado, definiu-se que seria utilizado o ESP32[1] como Dispositivo IoT e o NEO-6M[2] como Módulo GPS, ambas escolhas também se deram por conta do baixo custo de aquisição e baixo consumo de energia. Para obter as coordenadas geográficas, o Dispositivo IoT necessita se comunicar com o módulo GPS pelas portas TX e RX, posteriormente é realizado um processo de validação de coordenadas, caso sejam válidas, é chamado a função para envio ao Módulo de Comunicação IoT.

A função dos receptores GPS é descobrir a que distância eles estão de vários satélites. Os satélites transmitem informações sobre sua posição e a hora atual na forma de sinais de rádio em direção à Terra. Esses sinais identificam os satélites e informam

[1] Mais informações em: <https://www.espressif.com/en/products/socs/esp32>
[2] Mais informações em: <https://www.u-blox.com/en/product/neo-6-series>

ao receptor onde eles estão localizados. O receptor então calcula a distância de cada satélite, levando em consideração quanto tempo levou para os sinais chegarem. Uma vez que tenha informações sobre a distância de pelo menos três satélites e onde eles estão no espaço, o receptor pode identificar sua localização na Terra.

Para o desenvolvimento da `Aplicação Iot`, foi utilizado o módulo Wi-Fi integrado ao ESP32, módulo este é gerenciado pela biblioteca WifiManager[3] para realizar conexão à internet. No entanto, para comunicação entre o dispositivo e o `Módulo de Comunicação IoT` poderá ser usado, por exemplo, um módulo GSM com um chip de dados, pois há a necessidade do dispositivo estar conectado à uma rede móvel de internet em lugares remotos.

Com isto, foi implementado um código para o ESP32 onde primeiramente é inicializada a biblioteca WifiManager, na qual verifica se há arquivo de configuração Wi-Fi salvo no dispositivo contendo credenciais da rede Wi-Fi. Na ausência deste arquivo é propagado uma rede Wi-Fi para realizar a primeira configuração do dispositivo e, após conclusão, é armazenado um arquivo JSON contendo as informações da rede. Com o dispositivo conectado na rede, é iniciado a busca por satélites GPS e realizado uma conexão com o `Módulo de Comunicação IoT`, detalhado na seção a seguir.

3.3 COMUNICAÇÃO IOT

A comunicação entre o `Dispositivo IoT` e o `Módulo de Comunicação IoT`, na qual foi implementado utilizando o protocolo MQTT (HUNKELER; TRUONG; STANFORD-CLARK, 2008). Este protocolo de mensagens foi desenvolvido para telemetria M2M, utilizando pouca largura de banda da rede (HILLAR, 2017).

O MQTT trabalha com sistema de publicação e assinatura, sendo composto de um *Broker* e vários clientes. Serviço este denominado *Broker* é responsável por receber todas as mensagens, reconhecer qual cliente as enviou e enviá-las para os clientes inscritos. Os clientes são quaisquer dispositivos que comuniquem o protocolo MQTT e estejam conectados ao *Broker*. O cliente pode estar subscrito em um ou mais tópicos, recebendo as mensagens do *Broker* correspondentes àqueles tópicos. O cliente também pode ser um publicador, enviando mensagens aos tópicos do *Broker*. Em um último caso, o cliente pode ser ambos, recebendo e enviando mensagens.

As mensagens MQTT são publicadas em tópicos, que são áreas de interesse, os clientes assinam para receber mensagens de uma dada subscrição. As subscrições podem ser explícitas, ou organizadas hierarquicamente, com o cliente usando *wildcards* (# ou +) para receber mensagens de uma variedade de tópicos[4]. Neste trabalho, foi utilizado

[3]Mais informações em: <https://github.com/tzapu/WiFiManager>
[4]Mais informações em: <https://mosquitto.org/man/mqtt-7.html>

o tópico /tracker para o recebimento de mensagens.

3.4 RECEPÇÃO E CONSUMO DE DADOS

Em seguida, houve a definição da linguagem de programação Python para o desenvolvimento do (`Módulo de Recepção de Dados`) e da `API para Consumo dos Dados`. Python é uma linguagem de programação de alto nível orientada a objetos, também possui diversas bibliotecas disponíveis para auxiliar no desenvolvimento de uma aplicação.

3.4.1 Módulo de Recepção de Dados

O `Módulo de Recepção de Dados` foi desenvolvido como o Backend (MARYANSKI, 1980) deste trabalho[5]. O Backend registra-se como cliente no Broker MQTT através da biblioteca Paho[6] para monitorar e tratar as mensagens recebidas e enviadas em determinado tópico. Cada mensagem recebida no *Broker* conta com as seguintes informações: ID do dispositivo, tópico, comando, latitude e longitude. Desta maneira é possível receber e separar dados de vários dispositivos, classificando-os pela ID única de cada dispositivo.

Para manter o histórico de registros e também possibilitar que os dados sejam consumidos por outros serviços externos, após a mensagem ser recebida e tratada, o Backend confecciona uma *query* SQL de inserção e a executa no banco de dados PostgreSQL através do conector psycopg2[7].

3.4.2 Repositório de Dados

Para armazenamento dos dados coletados, foi realizado a viabilidade de utilizar um SGBD não relacional por se tratar de uma estrutura simples de dados. Porém, visando a escalabilidade do sistema e integração com todos os serviços, foi escolhido utilizar o SGBD relacional PostgreSQL.

PostgreSQL é um sistema de banco de dados *open-source* estável que fornece suporte a diferentes funções de SQL, como chaves estrangeiras, subconsultas, *triggers*, e diferentes tipos e funções definidas pelo usuário.

[5]Backend é o termo utilizado para códigos que que precisam se conectar com o banco de dados, gerenciar as conexões dos usuários e alimentar a aplicação com os dados previamente tratados.

[6]Disponível em: <https://pypi.org/project/paho-mqtt/>

[7]Disponível em: <https://pypi.org/project/psycopg2/>

3.4.3 API para Consumo dos Dados

O termo API significa *Application Programming Interface* ou, em português, Interface de Programação de Aplicativos. Uma API consiste em um conjunto de interações que utilizam o protocolo HTTP para se comunicar, geralmente utiliza JSON ou XML como formato de dados na qual é retornado.

Visando a necessidade de resgatar os dados previamente armazenados no banco de dados, foi realizado o desenvolvimento de uma API utilizando a biblioteca FastAPI do Python. Esta API tem como finalidade disponibilizar um servidor ASGI[8] e estar apta à receber requisições HTTP na rota configurada, esta rota possui como ação realizar conexão ao banco de dados e retornar as coordenadas geográficas armazenadas. Para uma melhor manipulação no retorno, a rota implementada na API possui um retorno do tipo JSON contendo latitude e longitude.

Na Figura 2 é apresentado o retorno da API, na qual mostra um conjunto de latitudes e longitudes coletados com um intervalo de 30 segundos.

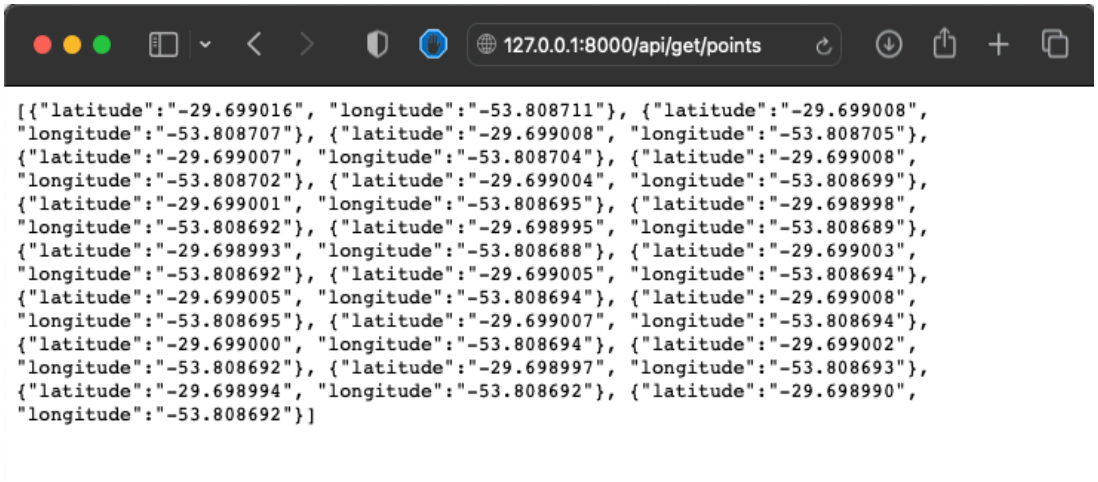


Figura 2 – Retorno da API

3.4.4 Arquitetura em Microsserviços

O termo microsserviço consiste em um tipo de arquitetura de software (NADAREISHVILI et al., 2016). Hoje ao desenvolver um software, o mesmo pode ser do tipo monolítico ou microsserviços. Na arquitetura monolítica, todos os serviços necessários estão configurados de forma integral e compartilhando recursos entre si, pois, a arquitetura monolítica possui como premissa trabalhar apenas com um servidor e centralizar serviços para operarem juntos.

[8]Mais informações em: <https://asgi.readthedocs.io/>

No entanto, com o passar dos anos a arquitetura monolítica passou a tornar os sistemas não escalonáveis, e suscetíveis a problemas de desempenho, visto que os serviços não operam de forma isolada. Com isto, surgiu a arquitetura denominada microsserviços. A arquitetura em microsserviços possui cada serviço operando de maneira individual e isolada, possibilitando assim otimização de desempenho e escalabilidade.

Como este projeto possui como objetivo a implantação dos serviços de forma independente, foi arquitetado e desenvolvido utilizando microsserviços, para isto foi utilizada a ferramenta Docker (ANDERSON, 2015). Com Docker é possível desenvolver cada microsserviço para operar de forma isolada, tendo a comunicação entre eles via protocolo de rede TCP. Visando um projeto modular e de fácil implementação, este projeto consiste em quatro microsserviços, sendo eles:

- Módulo de Comunicação IoT (Broker MQTT)
- Módulo de Recepção de Dados (Backend)
- Repositório de Dados (PostgreSQL)
- API para Consumo dos Dados (API)

Para a criação e configuração do microsserviço Broker MQTT, foi utilizado a imagem base “eclipse-mosquitto”, tendo como modificação o arquivo de configuração, na qual foi necessário adicionar o atributo “allow_anonymous true” para permitir requisições anônimas e exposta a porta TCP 1883 para permitir requisições externas.

Já na criação do microsserviço Backend, foi desenvolvido uma aplicação em Python e criado um arquivo Dockerfile, na qual é responsável para gerar uma imagem customizada. Esta imagem customizada possui como base a imagem “python” na versão 3, e instalado os pacotes necessários para o pleno funcionamento da aplicação, sendo eles: psycopg2 e paho-mqtt. Esta imagem customizada possui o comando “python ./main.py” para ser executado na sua inicialização.

O microsserviço API possui o mesmo código em Python do microsserviço Backend, porém, no arquivo Dockerfile é instalado as bibliotecas psycopg2, fastapi e uvicorn. Esta imagem possui o comando “uvicorn api:app –reload –host 0.0.0.0 –port 8000” para ser executado na inicialização, este comando é responsável por chamar a classe referente à API e inicializar o servidor ASGI.

Já o microsserviço PostgreSQL utiliza a imagem “postgres” na versão 13, informando variáveis de ambiente com o nome de usuário e senha do banco de dados. Também, foi criado um arquivo SQL de nome “create-tables.sql” e adicionado como *Entrypoint* do container, ou seja, é executado na primeira inicialização do microsserviço e tem como finalidade criar a estrutura necessária de tabelas. Na Figura 3 é possível visualizar a execução dos microsserviços no painel da ferramenta Docker.

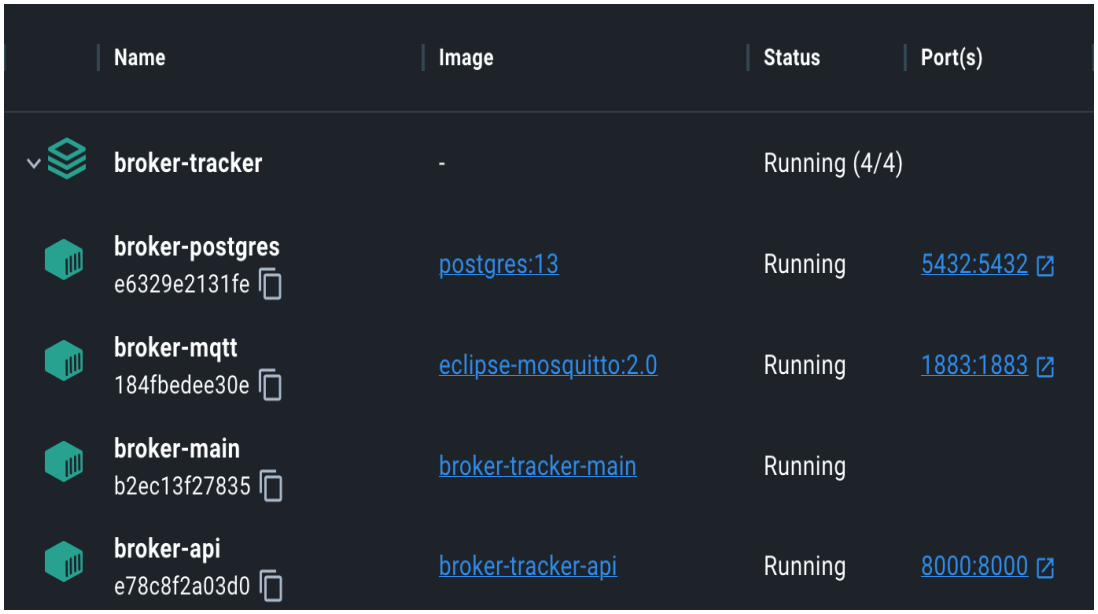


Figura 3 – Lista dos serviços

Para a execução e controle destes microsserviços foi utilizado o serviço Docker Compose na versão 3.7, responsável por criar cada container e gerir de maneira apropriada suas dependências e execução. No arquivo de configuração do Docker Compose, foi definido uma dependência do serviço “Broker MQTT” na inicilização do serviço “Backend”, ou seja, o serviço “Backend” somente vai inicializar caso o serviço “Broker MQTT” esteja em execução, a mesma dependência ocorre no serviço “API”, pois o serviço “Backend” precisa ser inicializado antes.

O Docker Compose possui o arquivo “.env” para a definição de variáveis de ambiente, neste arquivo foram definidas algumas variáveis para uso interno no serviço “Backend”, na qual utiliza estas informações para preencher o arquivo de nome “parameters.ini”, utilizado pelos serviços “Backend” e “API”. Esta configuração é feita no momento em que o serviço “Backend” é inicializado, pois neste momento é chamado a função “set_environments()” que realiza a leitura do arquivo “parameters.ini.sample”, preenche as informações necessárias e grava o arquivo “parameters.ini”.

Desta maneira, se torna possível que utilize um Broker MQTT ou um banco de dados externo e/ou de terceiros, basta remover estes serviços do arquivo “docker-compose.yaml” e ajustar as configurações no arquivo “./env”.

3.4.5 Hospedagem

Para hospedar e gerir todos os serviços necessários foi escolhido provisionar um ambiente Linux operando com o Sistema Operacional Debian na versão 11. A Debian é uma distribuição Linux onde cada versão é somente lançada após rigorosos testes de

segurança e correção de falhas, fazendo desta a mais segura e confiável dentre todas as outras distribuições Linux. É reconhecida como a mais segura, maior e atualizada mais frequentemente entre as outras distribuições Linux, além de ser a única sem fins comerciais.

Neste ambiente foram instalados os serviços GIT, Docker e Docker Compose para provisionar os serviços, foi realizado um clone do repositório contendo o projeto, parametrizado e executado o comando “docker-compose up -d” na raiz do projeto para subir os serviços. Mais detalhes sobre a arquitetura dos serviços está presente na Seção 3.4.4.

3.5 APLICATIVO MÓVEL

O React Native é um *framework* desenvolvido pela Facebook é baseado no React, que possibilita o desenvolvimento de aplicações móveis tanto para Android, como para iOS, utilizando apenas Javascript. Por meio dele, é possível desenvolver somente um código-fonte e reutilizá-lo para alimentar o aplicativo em ambas as plataformas.

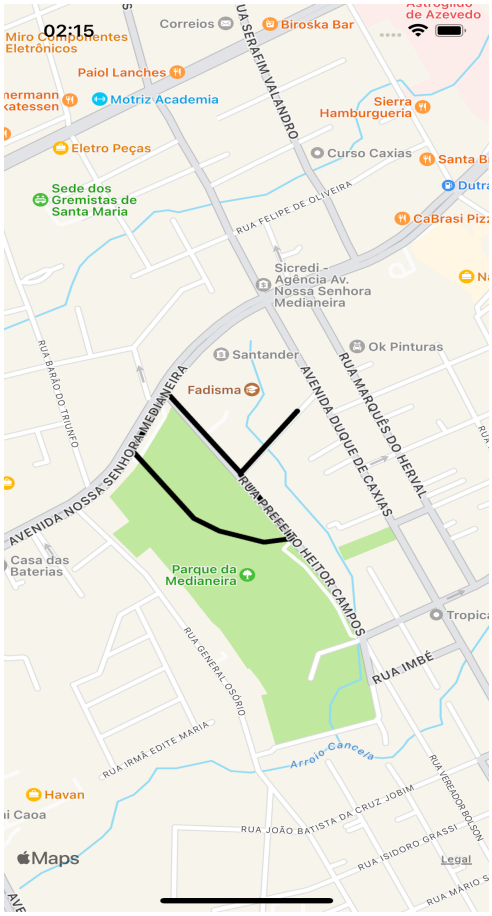


Figura 4 – Trajetória dos pontos geográficos

Com base nisto, foi escolhido este *framework* para o desenvolvimento do `Aplicativo`

Móvel, que usa o elemento MapView[9] para exibição do mapa. Na plataforma iOS é utilizado o mapa nativo do iOS, já no Android é utilizado o SDK do Google Maps e com isto se torna necessário obter uma chave de uso do SDK junto à Google.

Para obter os pontos geográficos, o aplicativo chama a API desenvolvida realizando uma requisição HTTP na rota "/api/get/points", recebendo assim um retorno no formato JSON. Esta chamada se dá por conta de uma função Assíncrona que é executada quando a variável "coordinatesPoints" não está definida e/ou está modificada, este evento é controlado pelo método "useEffect", nativo do *framework* React Native. Após receber os pontos geográficos via API, é utilizado o elemento "Polyline" para desenhar a trajetória dos pontos geográficos, possibilitando assim a exibição do histórico de localização do dispositivo. A Figura 4 representa a execução do aplicativo móvel após obter os pontos geográficos. A linha preta no mapa representa a trajetória dos pontos geográficos obtidos via API.

A Figura 5 mostra o cenário de teste para validação da proposta, onde percebe-se a utilização do ESP32 como Dispositivo IoT, o Neo6M como Módulo GPS, o Notebook Pessoal para Hospedagem dos Microsserviços e um Smartphone Apple para a execução do Aplicativo Móvel. A Figura 5 também apresenta uma visão geral dos módulos desenvolvidos (apresentados na Figura 1). Como Módulo de Comunicação escolheu-se o protocolo MQTT (Seção 3.4.1), enquanto que o Módulo de Recepção de Dados e a API para Consumo dos Dados foram implementados na linguagem Python. Para manter os dados, optou-se pelo PostgreSQL como Repositório de Dados. Por fim, o Aplicativo Móvel que mostra o rastreamento em um mapa, foi implementado em React Native (Seção 3.5).

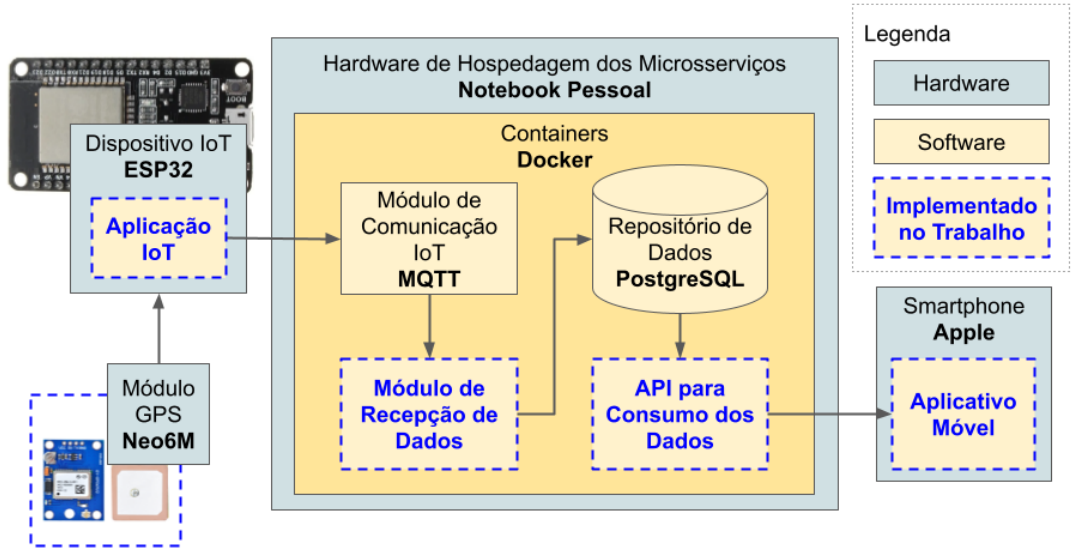


Figura 5 – Implementação da arquitetura dos serviços

[9]Disponível em: <https://docs.expo.dev/versions/latest/sdk/map-view/>

O sistema de rastreamento implementado neste trabalho possui código aberto (*open-source*) disponível no GitHub e dispõe de três repositórios, sendo eles: Código-fonte do ESP32[10], projetos das aplicações Backend e API[11], e projeto do aplicativo móvel[12].

[10] Disponível em: <https://github.com/luiznunes/client-tracker>
[11] Disponível em: <https://github.com/luiznunes/broker-tracker>
[12] Disponível em: <https://github.com/luiznunes/app-tracker>

4 CONCLUSÃO

Atualmente, dispositivos IoT são utilizados em diversas áreas, sejam elas automação residencial, monitoramento de saúde, rastreamento de objetos, dentre outras. Dispositivos IoT estão em constante evolução e podem ser controlados remotamente e integrar com serviços externos, como sistemas de rastreamento integrado com um *smartphone*.

Neste trabalho, foi apresentada uma arquitetura e as fases executadas para o desenvolvimento de um sistema de rastreamento utilizando dispositivo e módulos IoT integrado com mapas, este sistema possui componentes de baixo custo e foi desenvolvido com a premissa de fácil implantação. A validação deste trabalho ocorreu em bancada de testes e em ambiente fechado, com isto verificou-se que o módulo de GPS precisa estar em ambiente aberto para um resultado mais preciso. Para a comunicação com a internet durante a etapa de validação, foi utilizado o módulo integrado de Wi-Fi do ESP32. Em um ambiente real necessita ser utilizado um módulo GSM para comunicação móvel.

Este trabalho teve como objetivo explorar a integração de dispositivo IoT com os demais serviços demonstrados ao longo do trabalho,

Este trabalho não focou aspectos como a segurança e/ou otimizações de recursos. Recomenda-se então, como trabalhos futuros, o estudo de mecanismos de segurança utilizando autenticação no Broker MQTT para a troca de mensagens. Outra melhoria na questão da segurança é a implementação de um método de login no APP em conjunto com a API desenvolvida. Ainda na implementação da API, tem-se como trabalho futuro o recebimento de argumentos tais como faixa de horário e ID do dispositivo, visando filtrar e otimizar a consulta retornada. A API foi desenvolvida de forma que seja possível implementar o envio de comandos ao dispositivo via requisição HTTP em uma rota específica, implementando esta rota se torna possível enviar comandos ao dispositivo partindo do APP, recomenda-se a implementação de tempo de atualização do mapa, que por padrão é 30 segundos. Esta rota ao receber a requisição pode enviar um comando ao Backend e realizar uma publicação de mensagem no Broker MQTT, e o dispositivo por sua vez estará monitorando o tópico em específico.

Este projeto realiza consultas SQL em uma tabela de banco de dados que, grava em média de 80mil registros por mês para cada dispositivo, tendendo assim a aumentar o tempo de resposta no retorno da API. Devido à isto, recomenda-se a otimização de tabela utilizando índices para melhorar o tempo de resposta nas consultas, assim como a configuração do SGBD para utilizar o *plugin* TimescaleDB[1], na qual possui diversas otimizações para registros de histórico de dados.

[1]Mais informações em: <https://github.com/timescale/timescaledb>

REFERÊNCIAS

AL-SAQQA, S. et al. **How technology affects our life: The case of mobile free minutes in Jordan**. 2014. 1111–1127 p.

ANDERSON, C. Docker [software engineering]. **Ieee Software**, IEEE, v. 32, n. 3, p. 102–c3, 2015.

ASHTON, K. et al. That internet of things thing. **RFID journal**, Hauppauge, New York, v. 22, n. 7, p. 97–114, 2009.

CIUFFOLETTI, A. Low-cost iot: A holistic approach. **Journal of Sensor and Actuator Networks**, MDPI, v. 7, n. 2, p. 19, 2018.

COULBY, G. et al. Low-cost, multimodal environmental monitoring based on the internet of things. **Building and Environment**, Elsevier, v. 203, p. 108014, 2021.

HILLAR, G. C. **MQTT Essentials - A Lightweight IoT Protocol**. [S.l.]: Packt Publishing, 2017.

HUNKELER, U.; TRUONG, H. L.; STANFORD-CLARK, A. Mqtt-sa publish/subscribe protocol for wireless sensor networks. In: IEEE. **2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)**. [S.l.], 2008. p. 791–798.

MARYANSKI, F. J. Backend database systems. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 12, n. 1, p. 3–25, 1980.

MCELROY, E. Postsocialism and the tech boom 2.0: Techno-utopics of racial/spatial dispossession. **Social Identities**, Taylor & Francis, v. 24, n. 2, p. 206–221, 2018.

NADAREISHVILI, I. et al. **Microservice architecture: aligning principles, practices, and culture**. [S.l.]: "O'Reilly Media, Inc.", 2016.

PATII, N.; IYER, B. Health monitoring and tracking system for soldiers using internet of things(iot). In: **2017 International Conference on Computing, Communication and Automation (ICCCA)**. [S.l.: s.n.], 2017. p. 1347–1352.

PUJARIA, U. et al. Internet of things based integrated smart home automation system. **SSRN Electronic Journal**, 01 2020.

RAY, P.; DASH, D.; DE, D. A systematic review and implementation of iot-based pervasive sensor-enabled tracking system for dementia patients. **Journal of Medical Systems**, v. 43, 07 2019.

SAMSON, A. M.; DHAKSHYANI, R.; ABDULLA, R. Iot based sustainable wallet tracking system. **International Journal of Advanced Science and Technology**, v. 29, n. 1, p. 1301–1310, 2020.

SINGH, S.; KUMARI, P. Automatic car theft detection system based on gps and gsm technology. **International Journal of Trend in Scientific Research and Development (IJTSRD)**, 2019.

WEBER, R. H.; WEBER, R. **Internet of things**. [S.l.]: Springer, 2010. v. 12.