

Manual de Instruções

LOCUS
IPT

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
<17/11/2022>	<Vitor Zeferino>	<2>	<Preenchimento dos componentes externos utilizados>
<17/11/2022>	<Melyssa Rojas>	<2.2>	<Preenchimento dos componentes de hardware utilizados na placa>
<17/11/2022>	<Melyssa Rojas>	<2.3>	<Preenchimento das tecnologias utilizadas no backend>
<18/11/2022>	<Júlia Togni>	<2.4>	<Preenchimento do guia de montagem>
<18/11/2022>	<Ariel>	<2.5>	<Preenchimento da solução>
<18/11/2022>	<Ariel>	<2.6>	<Preenchimento da arquitetura>

<02/12/2022>	<Melyssa Rojas>	<2.7>	<Preenchimento do guia de instalação>
<02/12/2022>	<Melyssa Rojas>	<2.8>	<Preenchimento do guia de configuração>
<14/12/2022>	<Melyssa Rojas>	<2.9>	<Preenchimento da seção 8 e formatação do manual>
<15/12/2022>	<Melyssa Rojas>	<3>	<Preenchimento da seção 6, 7 e a atualização da arquitetura>

Índice

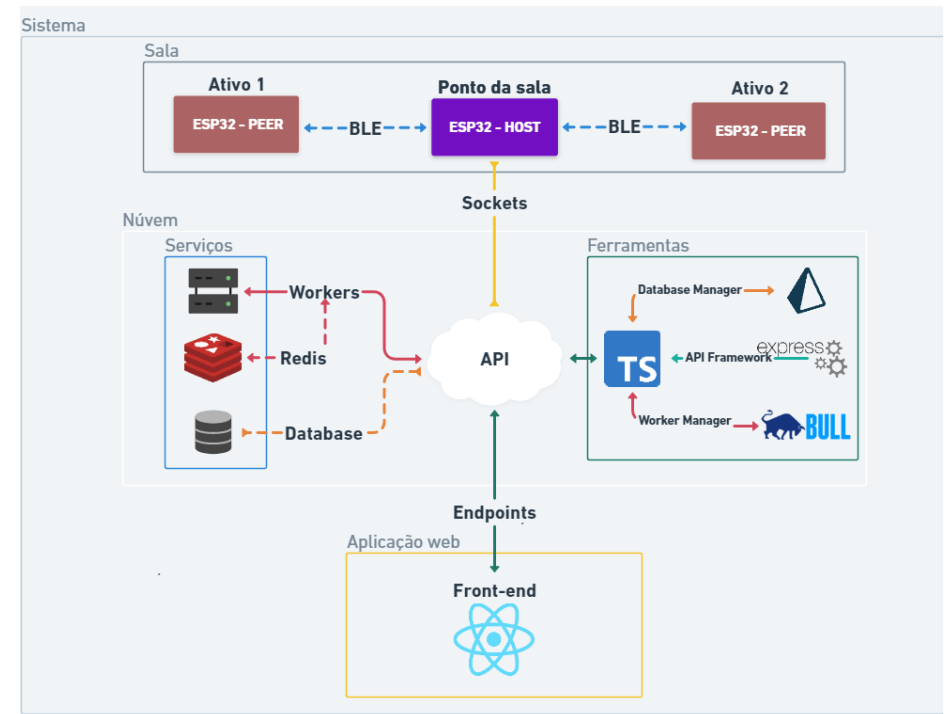
1. Introdução	3
1.1. Solução	3
1.2. Arquitetura da Solução	3
2. Componentes e Recursos	5
2.1. Componentes de hardware	5
2.2. Componentes externos	6
2.3. Requisitos de conectividade	7
3. Guia de Montagem	9
4. Guia de Instalação	10
5. Guia de Configuração	16
6. Guia de Operação	20
7. Troubleshooting	21
8. Créditos	22

1. Introdução

1.1. Solução

A nossa proposta de solução é implementar o sistema de rastreamento para que quando algum funcionário da controladoria for buscar por um ativo ele consiga, em tempo real, rastrear a sala em que o objeto está. Além disso, caso o aparelho saia do campus do IPT, esse funcionário é prontamente notificado para questionar ao responsável sobre a saída do equipamento sem aviso prévio ou qualquer outro tipo de problema que possa ter ocorrido.

1.2. Arquitetura da Solução



Informações gerais

Nosso fluxo de funcionamento principal:

Resgatar informações da posição de um equipamento -

Dentro da página de aplicação *web*, será possível solicitar diretamente a posição de um ativo em específico, quando o mesmo for solicitado, a requisição será feita para a *API*, que por sua vez, verifica no banco de dados o *ID* do dispositivo em que esse equipamento está "conectado", assim, pode-se procurar no banco de dados os históricos de posição do objeto e retornar ao *front-end* (usuário).

Equipamentos utilizados:

Neste exemplo foi utilizado três placas ESP32 para simular o sistema, nos quais, são independentes de qualquer outro dispositivo / sensor, pois nossa solução atualmente não requer outros componentes.

Descritivo:

ESP32-WROOM-32D: Placa principal utilizada para ser "marcadores" de posição, assim, comparando a posição do mesmo com outros dispositivos já conhecidos, é possível prever o local do dispositivo.

API e Aplicação Web:

Tanto a *API*, quanto a aplicação *web* devem ser hospedadas em algum outro servidor para serem acessados de qualquer local, além dos próprios dispositivos de rastreamento.

Descrição de posicionamento dos equipamentos:

Nessa solução, existem, como já explicado, duas alternativas de segmento para os microcontroladores, sendo o **ESP32-HOST** e o **ESP32-PEER**, o **HOST** deve ser posicionado em cada sala que deseja identificar a posição, criando uma área semelhante a uma *Rede Mesh*(rede de "teia", onde há vários dispositivos em um certo local, criando uma corrente), assim, pode-se comprar a posição do **HOST** em comparação com os outros microcontroladores, no caso, os **PEERS**, que por sua vez faz o trabalho de ser as *Tags* de identificação de posição, onde serão anexados aos equipamentos a serem monitorados. Lembrando que o dispositivo **HOST** deve ter alcance para alguma rede *wifi* para se conectar a *API*.

A proposta é que o **HOST** esteja conectado na alimentação constante(se possível), e os **PEERS** conectados na alimentação externa(baterias)

Sala

ESP32 - HOST:

Responsável por receber as solicitações por *sockets web* e retornar as informações de posições dos equipamentos de sua área.

O *Host* estará diretamente conectado a uma rede wifi para se comunicar com a *API* e sempre estar a espera de novos comandos / solicitações. Exemplo: Recebe solicitação de posição dos equipamentos mais próximos, assim, o mesmo retorna as informações.

ESP32 - PEER:

Subordinado que estará sempre 'dormindo' durante inatividade, ao menos que seja solicitado a identificação para que o *Host* também tenha informação da sua distância atual do mesmo.

O *Peer* tem como seu único objetivo ser uma *Tag* que armazena um *ID* único, que estará atrelado a um equipamento (pelo sistema de *API*).

ESP32 Host para Esp32 Peer (BLE)

O BLE (bluetooth low energy) que é citado como comunicação entre os microcontroladores, são nada mais que rotas de resposta do escaneamento via bluetooth, que faz o calculo de medida de distância entre os microcontroladores com base na intensidade da potência.

Núvem

API:

É o serviço a ser consumido pelos microcontroladores e *front-end*(aplicação web). Com a *API* será possível ter acesso a conexão por *sockets* e *endpoints* de rotas, sendo respectivamente, para a comunicação constante com os microcontroladores, pois os dados de posições / atualizações devem ser enviados em tempo real, assim o *sockets* teria um ótimo desempenho, reduzindo a necessidade de chamada repetitivas de rota em "*heath beat*", já para as rotas de *endpoints* web, seria para consumo da aplicação que o usuário final utilizará, ou seja, rotas que requisita informações de posições de algum equipamento, histórico do mesmo...

Serviços:

São os auxiliares da *API*, onde pode ser encontrado o banco de dados de longo prazo (como por exemplo PostgreSQL), o banco de dados de curto prazo (Redis) junto ao seu "sistema" de *workers*. O banco de dados de longo prazo é utilizado para armazenar dados como: O *ID* de tal dispositivo representa o equipamento X, gerenciamento de usuários e acessos...

Agora sobre os *workers*, são essencialmente tarefas realizadas em segundo plano, não atrapalhando os processos principais, como o serviço de rotas. (O *Redis* nessa situação é um armazém de dados temporários sobre as tarefas realizadas). Em prol do projeto, seria utilizado para monitoria / salvamento das posições dos equipamentos, assim gerando o histórico de posições do mesmo.

Ferramentas:

As ferramentas são as tecnologias a serem utilizadas na *API*, sendo citado, Typescript, que por sua vez, tem como pilares do projeto, as bibliotecas, Prisma(faz o gerenciamento do banco de dados por ORM), Express(Framework para criar os endpoints e sockets) e *JsBull*(Que gerencia de forma fácil as tarefas de segundo plano, os *workers*).

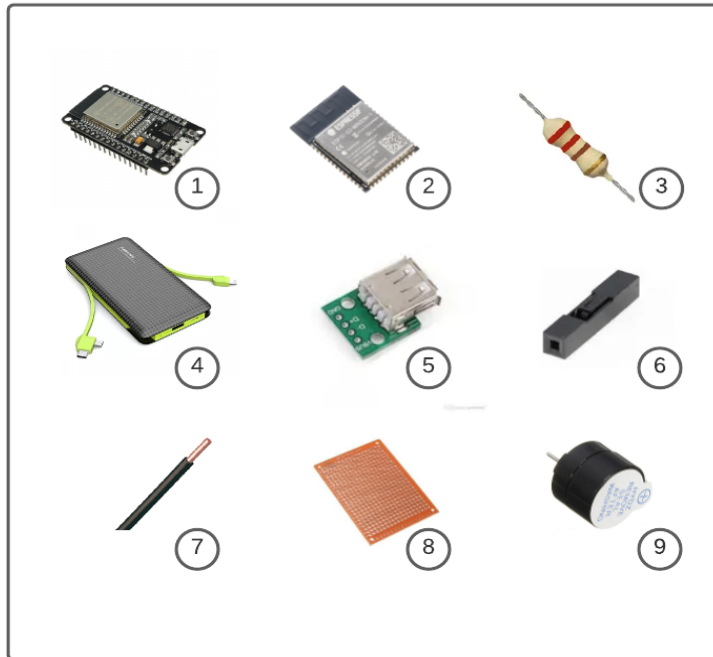
Aplicação Web

Front-End:

Para a criação do *Front-end* e consumo dos *endpoints* da *API*, utilizaremos em base o *React Native*(uma *framework* de desenvolvimento *web*, que facilita processos, sendo gerado a maioria do código de *front*, em próprio *JavaScript*). Com essa tecnologia, será possível entregar de maneira mais eficaz a solução.

2. Componentes e Recursos

2.1. Componentes de hardware



Nome (modelo e marca)	Especificação técnica	Funcionalidade na solução
devkit ESP-32-S3 (1);	Integração do software no hardware;	Funciona como receptor de posicionamento de outro esp, manda dados para o servidor via websockets, e ainda, dependendo

		de sua função, cria uma rede própria;
ESP32-S3-WROOM-1/1U (2);	Processa dados e executa funções;	Processa dados e executa funções da aplicação;
2 resistores (resistência 220 Ohms) (3);	Diminuidor da tensão;	Ele diminui a tensão para haver a leitura da bateria;
Carregador Portátil Power Bank Pineng 10000 Mah V8 (4);	Bateria;	Serve para energizar o dispositivo lot;
Módulo Adaptador USB Fêmea 2.0 (5);	Conector de entrada USB;	Serve para conectar o esp32 ;
Conector Jumper fêmea (6);	Auxilia na conexão dos cabos;	Ele diminui a tensão para haver a leitura da bateria;
Cabos sólidos (7);	São cabos que possuem apenas um fio de cobre;	São usados para a conexão dos componentes da placa ilhada;
Placa Ilhada (8);	Placa utilizada para auxiliar na conexão entre os componentes;	Essa placa permite a soldagem dos fios e dos componentes que vão ser usados;

Buzzer (9) (TMB12A05);	Componente que solta frequências de sons;	Permite que o usuário saiba onde está o dispositivo IoT, por meio do som;
---------------------------	---	---

2.2. Componentes externos

Nome	Especificação técnica	Utilidade na solução
API		É o serviço a ser consumido pelos microcontroladores. Acesso a conexão por sockets e endpoints de rotas para comunicação constante com os microcontroladores.
PostgreSQL		Banco de dados de longo prazo para guardar informações como

		ID.
Redis		Banco de dados de curto prazo para armazenar informações temporárias como localização do ESP.
NodeJS e suas bibliotecas		Gerenciamento do banco de dados, criar endpoints e gerenciar as tarefas de segundo plano.
React Native		Criação do Front-End e consumo dos endpoints da API.
Dispositivo	Conexão com a internet	Necessário para utilizar o dashboard da aplicação.

2.3. Requisitos de conectividade

Nome (tipo)	Especificação técnica
-------------	-----------------------

Node (ambiente de execução);	Plataforma utilizada para executar e criar códigos typescript fora do navegador;
Prisma (library);	Ferramenta que utiliza ORM (Object-relational mapping) para a criação das tabelas;
Express (framework);	Ferramenta utilizada para a criação de rotas, endpoints e da criação do servidor;
Socket.io (library);	Ferramenta utilizada para a criação da comunicação via websocket;
Typescript (linguagem de programação);	Linguagem utilizada na criação do backend;
npm (package manager)	Um gerenciador de pacotes que permite instalar, desinstalar e atualizar dependências do projeto;
WiFi (library)	Uma biblioteca que permite que os dispositivos inteligentes possam usufruir de funções da própria biblioteca, funções essas que permitem a conexão a uma rede local, criar

	uma própria rede como ponto de acesso;
WebSocketsClient (library)	Uma biblioteca que permite o cliente participar da troca de mensagens entre ele próprio e o servidor, por meio da inicialização da comunicação via ip e da porta do servidor;
Plataform.io	Editor de código para sistema embarcado com as bibliotecas necessárias instaladas;

3. Guia de Montagem

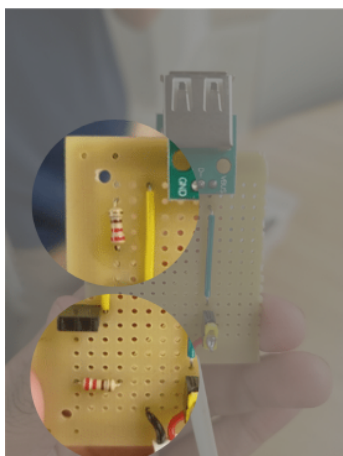
Separe todos os componentes listados na seção 2.1 e utilize também um ferro de solda para a montagem do medidor.

1. Para a montagem da placa de medidor de bateria do

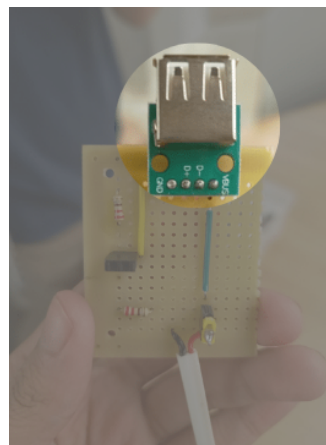


esp:

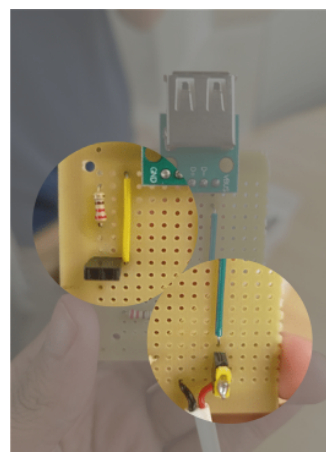
2. Na placa ilhada, adicione os dois resistores (componentes de número 3) como mostrado na figura abaixo



3. Adicione o componente 6 na placa



4. Conecte o componente 5 ao componente 6 e conecte ao componente 7 (o fio)



Para a montagem do buzzer com o ESP 32 S3, segue a montagem de acordo com a preferência do usuário, mas abaixo haverá um exemplo de montagem:

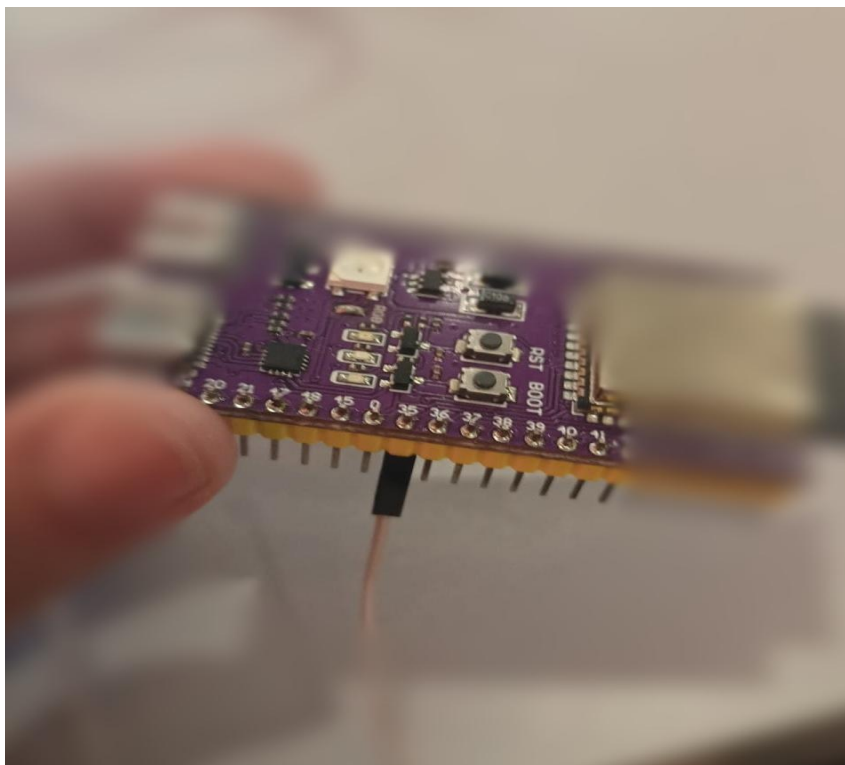
1. Use dois cabos, esses dois serão de um lado fêmea e do outro fêmea também deste tipo abaixo:



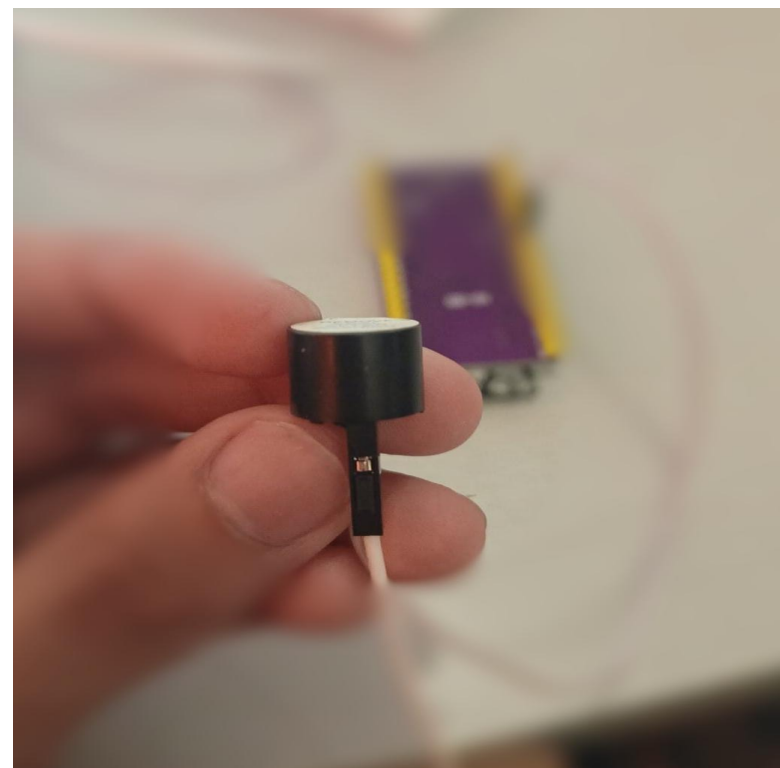
2. Conecte um lado fêmea com a parte positiva do buzzer



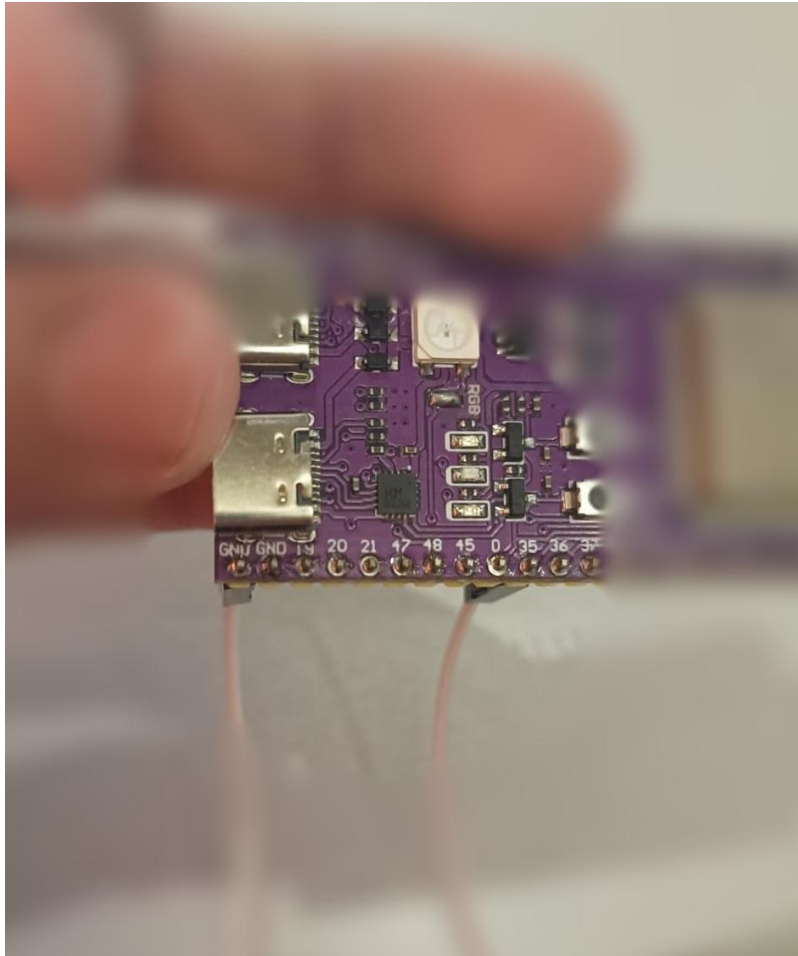
3. Com o outro lado do cabo use-o para se conectar com qualquer pino do ESP32-S3 que seja valor numérico



4. Pegue o outro cabo e coloque um lado deste no outro pino do buzzer



5. E agora coloque o outro lado desse cabo no pino terra(GND)



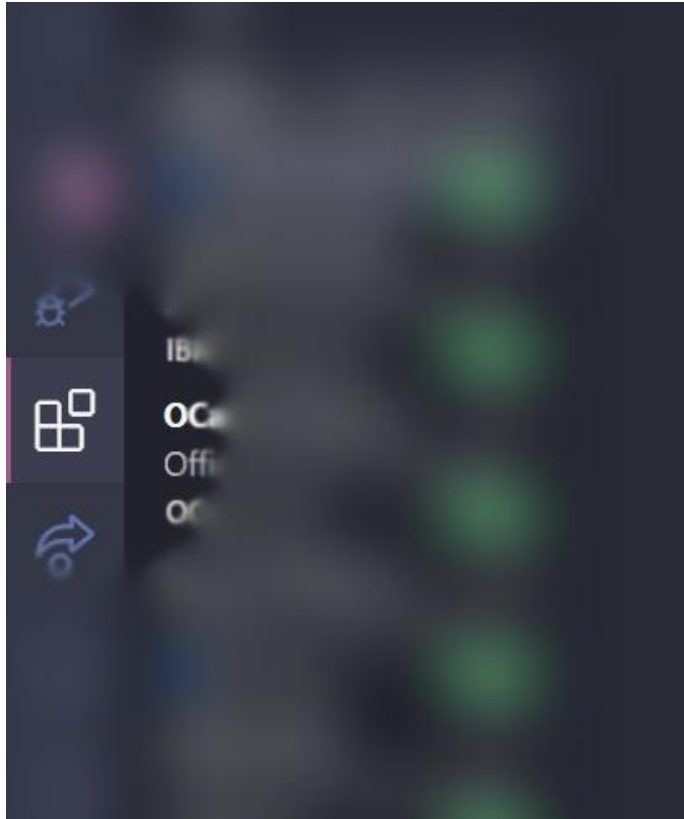
4. Guia de Instalação

Requisito de tecnologia:

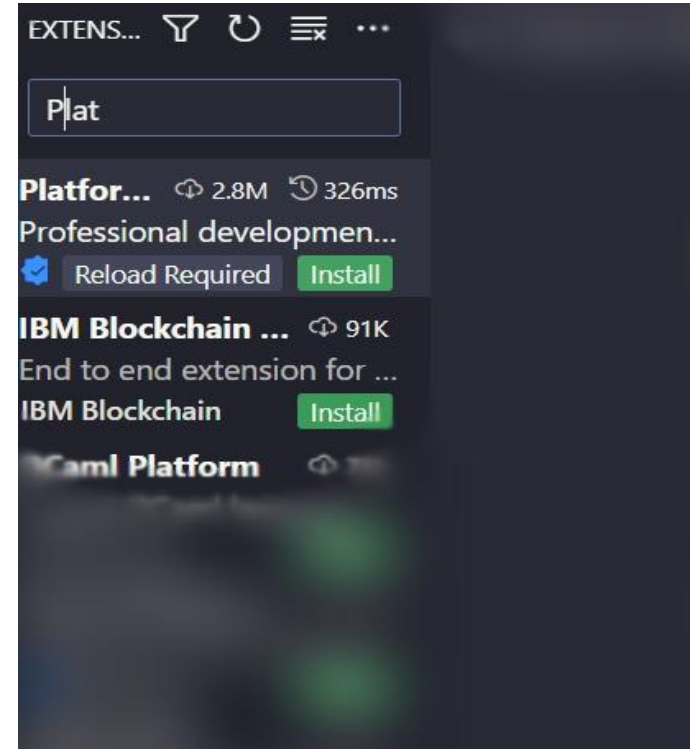
- Instale o vscode na máquina;
- Instale a extensão do Platform IDE no vscode, como no exemplo abaixo:

Passo a passo na instalação do Platform IDE:

- Vá até o vscode e caminhe até a parte de extensões na sidebar da esquerda do editor;



- Depois digite Platform IDE e instale o primeiro item que aparecer:

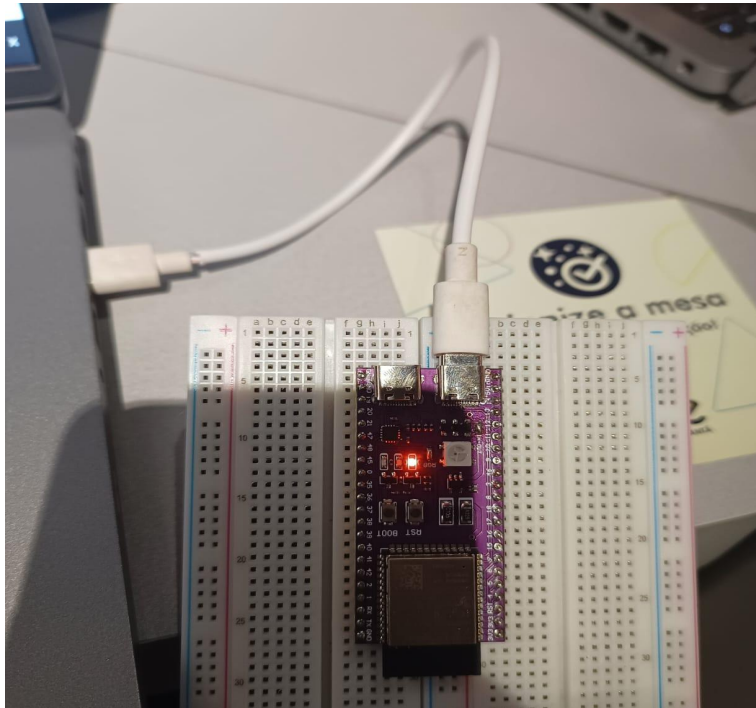


Passo a passo:

1. Siga a montagem do guia da seção 3 para todos os ESP 32-S3 que for utilizar;
2. Pegue todos os ESP 32-S3 que for ser o host (dispositivo que irá ficar de preferência no teto em vários cantos da sala que

não sejam nas bordas da sala) e coloque o código dentro deles, de acordo com o passo a passo abaixo:

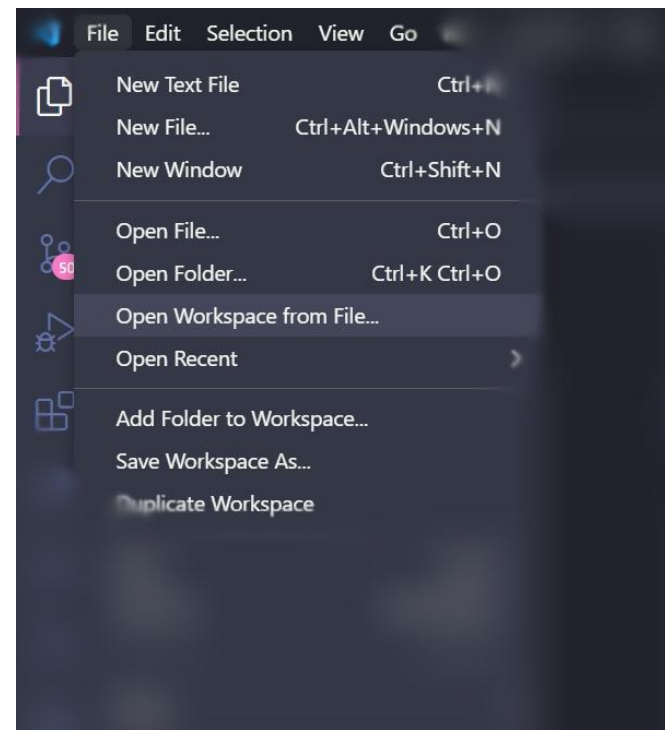
- Conecte com um cabo de comunicação serial USB:



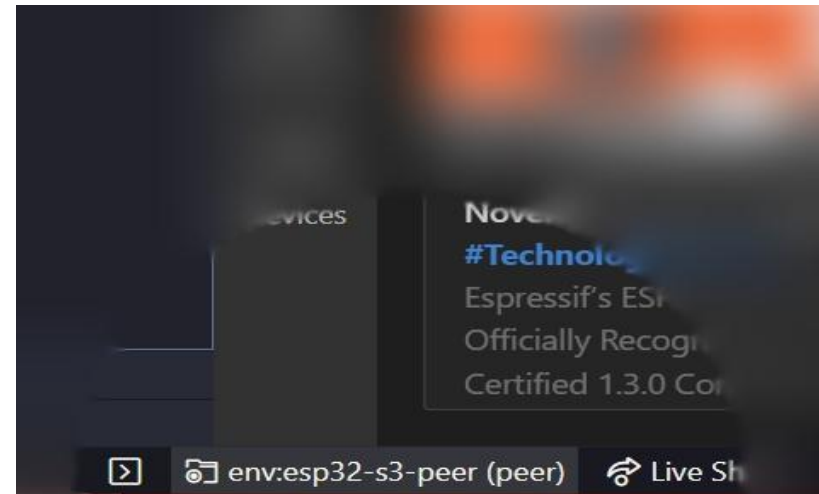
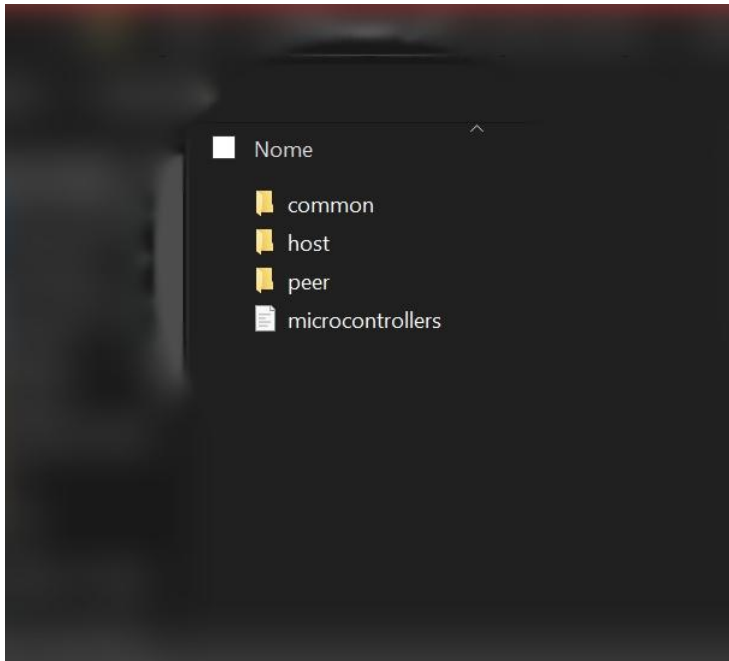
- Clone o código do github com esse link
<https://github.com/2022M4T1-Inteli/Projeto5.git>;

- Entre no vscode e abra o diretório que contém o projeto clonado do github;

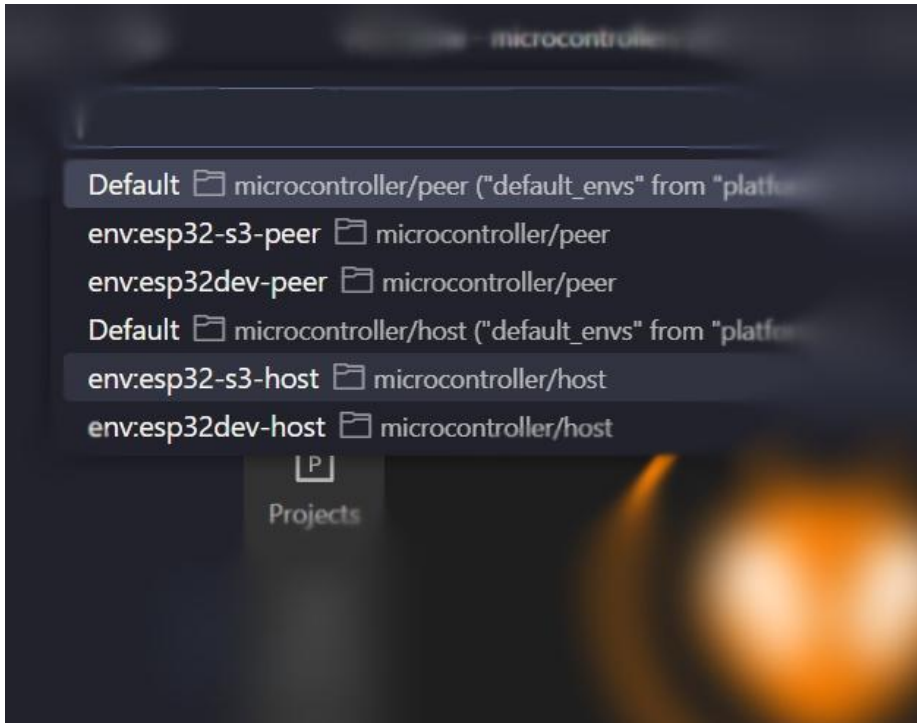
- Vá até file no vscode e clique no 'Open Workspace from file':



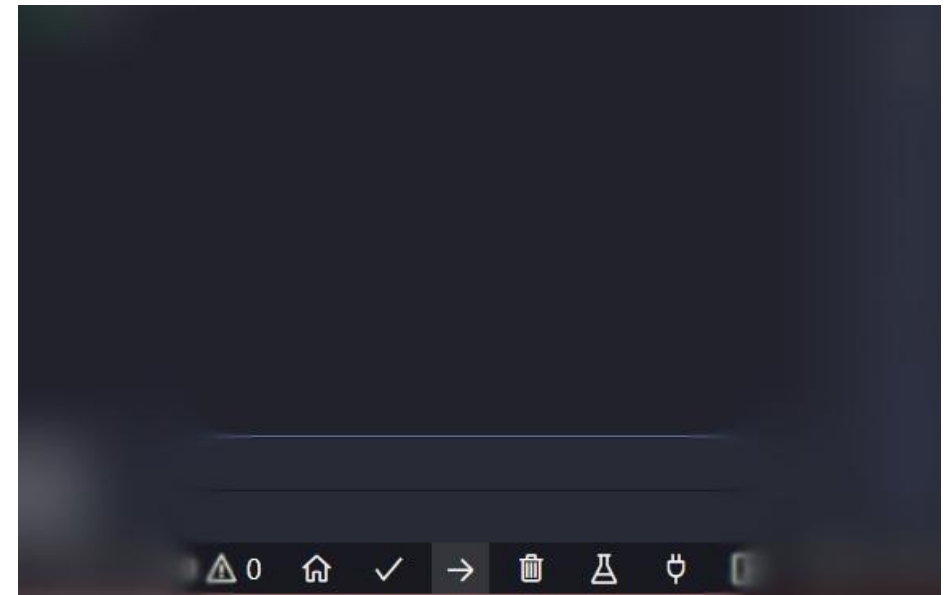
- Depois vá até a pasta microcontroller e clique no arquivo escrito microcontrollers que irá abrir um workspace no vscode:



- No canto inferior do vscode, com o workspace já aberto, clique na aba de switch para mudar o ambiente do projeto que está sendo executado para o ambiente de host, no caso, "env:esp32-s3-host":

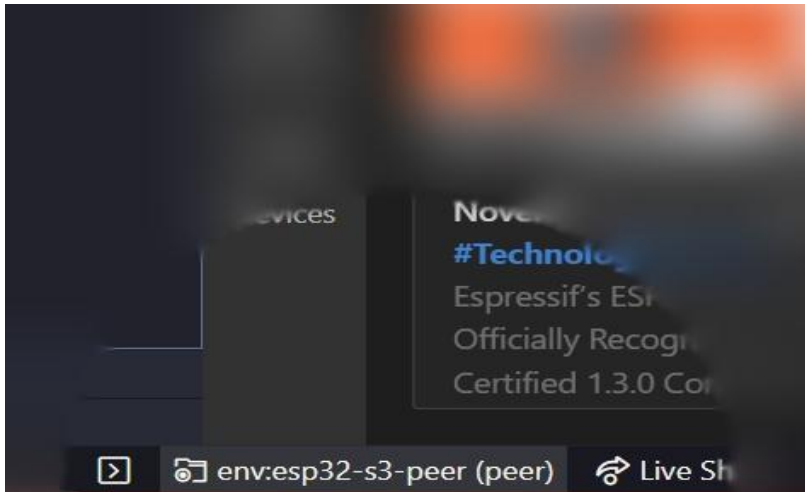


- Agora com o ambiente certo para a execução, clique no "Upload" para carregar o código no dispositivo:

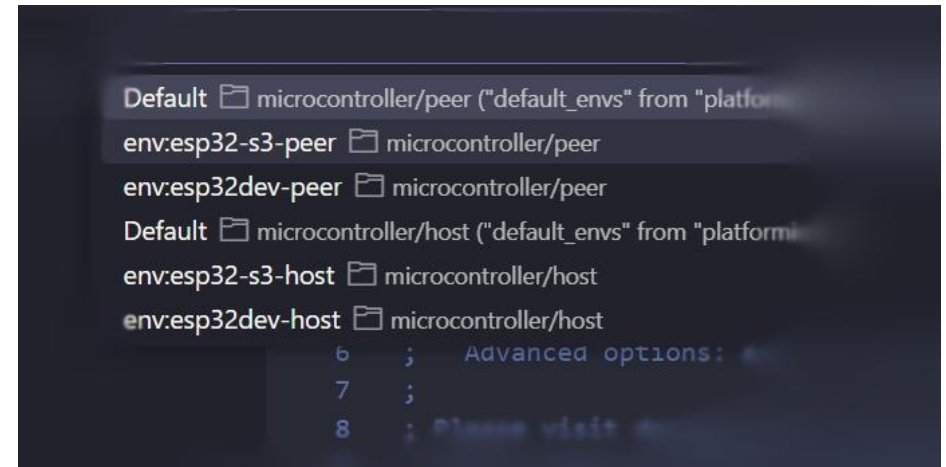


- Com o código carregado no host, coloque o esp no lugar ideal especificado no passo dois desta seção;
3. Pegue todos os ESP 32-S3 que for ser o peer (dispositivo que irá ficar no equipamento a ser rastreado) e coloque o código dentro deles, de acordo com o passo a passo abaixo:

- Repita todos os passos da etapa 2 do host até a parte que clica no “Open Workspace from file”, logo após isso, clique na aba switch abaixo:



- Após realizar o procedimento acima irá abrir um campo de escolha do ambiente de projeto automaticamente na parte superior da tela do vscode, nesse campo escolha o “env:esp32-s3-peer” e clique nele:



- Agora com o ambiente certo para a execução, clique no “Upload” para carregar o código no dispositivo:

- O esp32 - S3 contém alcance de 5 metros;

Limites:

- Como se trata do cálculo distância pelo escaneamento da potência, então há possibilidade de não haver precisão absoluta, devido a potência sofrer facilmente interferências;

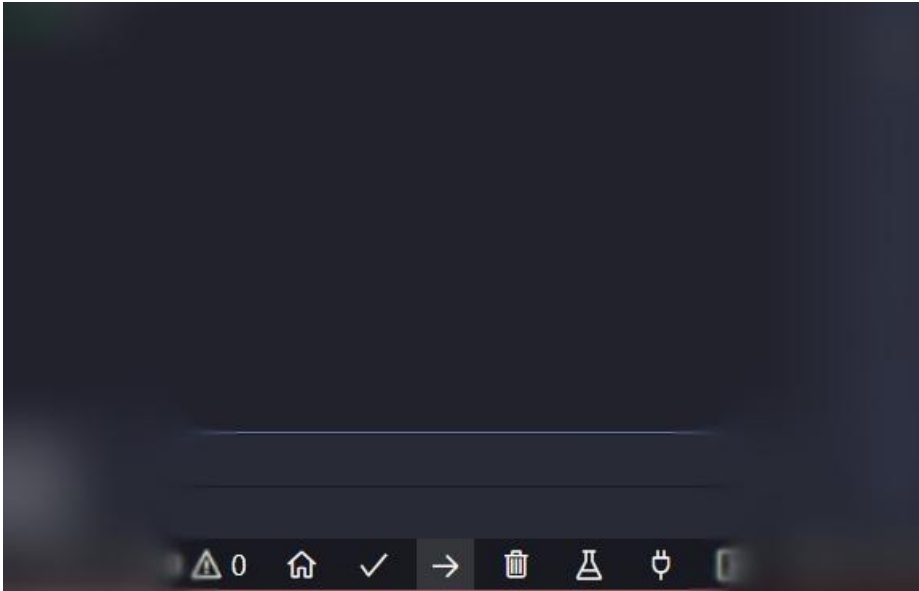
5. Guia de Configuração

Requisito de tecnologia:

- Instale PostgreSQL (banco de dados relacional) na máquina local, como mostrado no passo a passo abaixo:

Passo a passo na instalação do PostgreSQL:

- Clique neste link que irá encaminhar para a parte de downloads do PostgreSQL, <https://www.postgresql.org/download/>;
- Clique no ícone do seu sistema operacional:

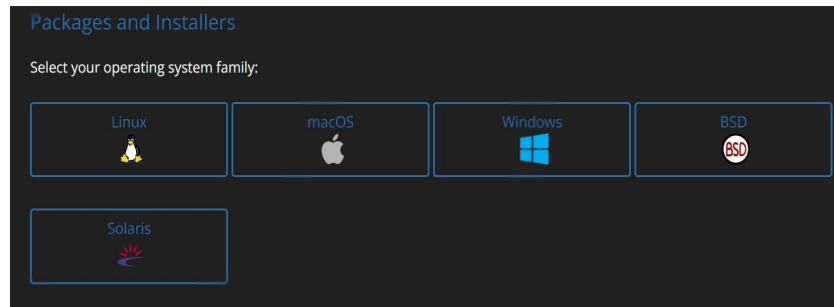


- Após o procedimento acima realizado, o peer já está pronto para ser colocado sobre o dispositivo a ser rastreado;

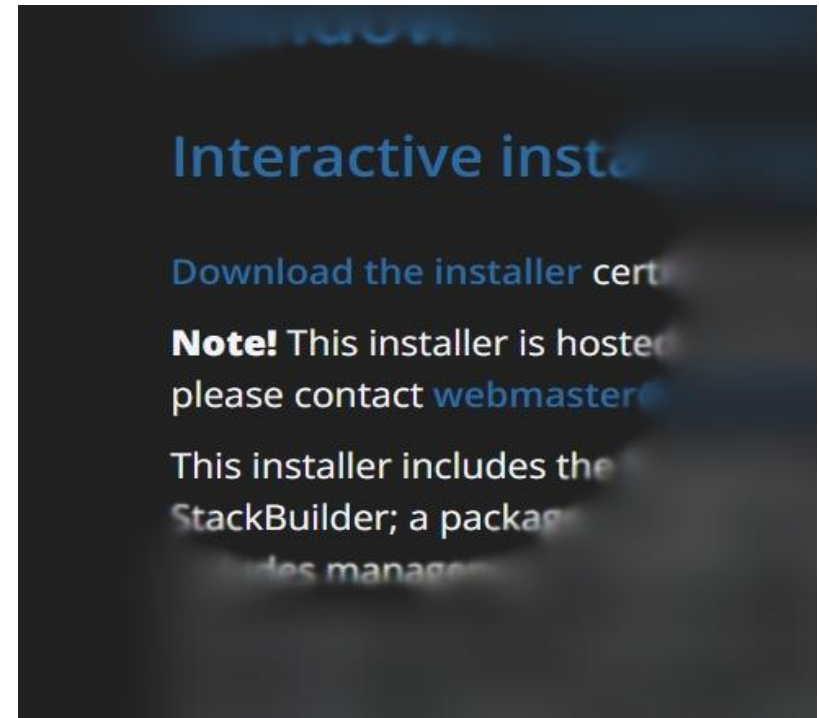
Propriedades:

- A conexão entre os dispositivos e o rastreamento da localização do dispositivo é feito por Bluetooth e por potência;

Alcance:



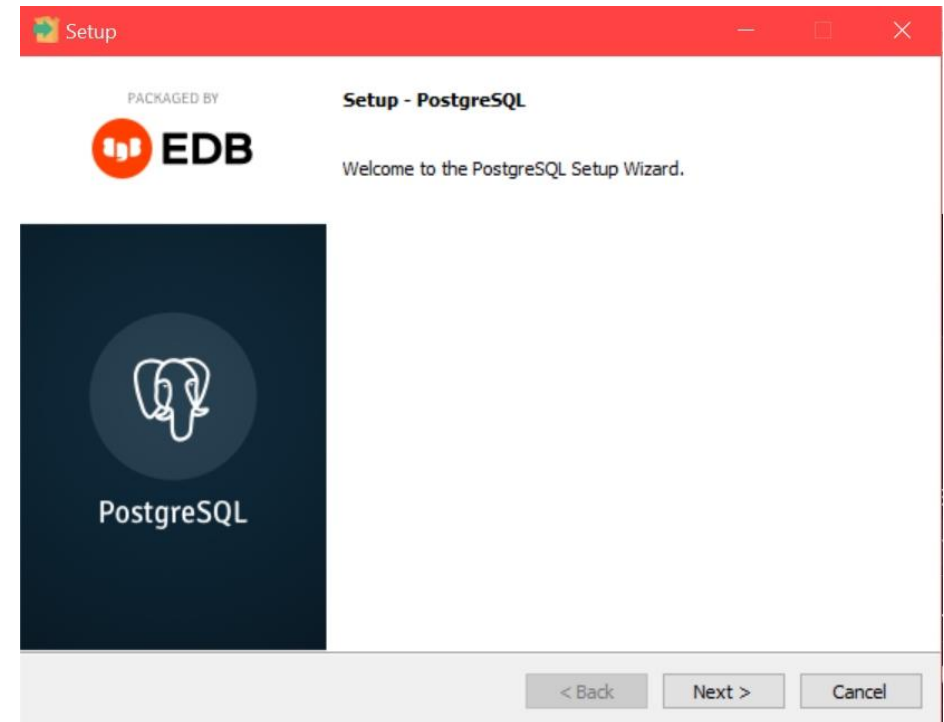
- Clique no "Download the installer":



- Clique nesse ícone na página que abriu pós etapa anterior dependendo do seu sistema operacional utilizado, irá ser baixado o arquivo executável do PostgreSQL:



- Abra o arquivo executável e irá aparecer uma tela assim na sua máquina:



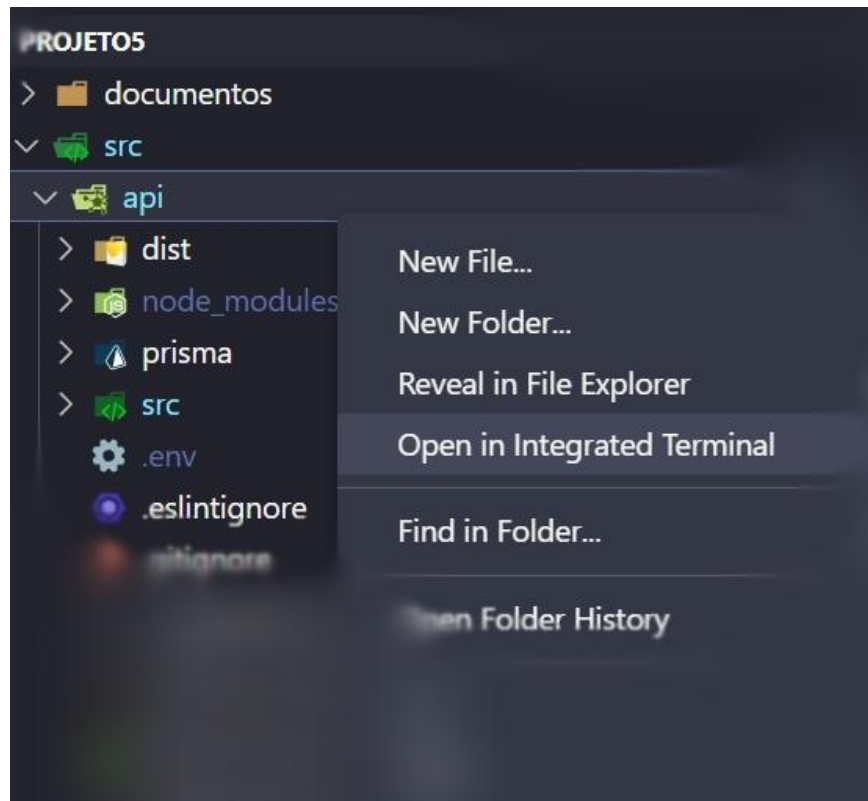
- A cada arquivo clique em “next”, na última página vai haver um campo “finish” clique sobre ele, e assim, o PostgreSQL vai estar instalado;

Logo abaixo, temos uma lista indicando os procedimentos necessários para a configuração das tecnologias usadas para executar o servidor (componente usado para servir informações para o usuário):

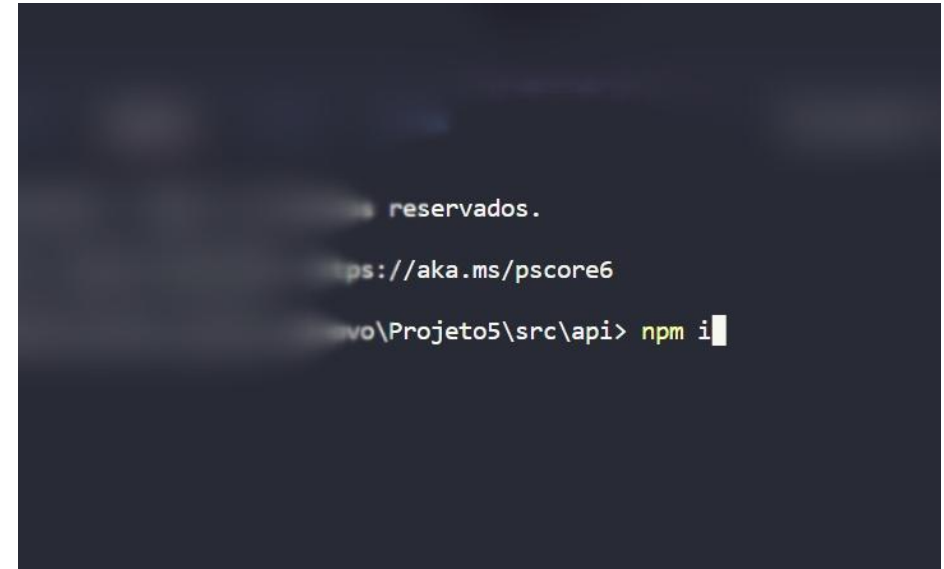
Passo a passo:

1. Com o projeto do github na máquina local, etapa mostrada na seção 4, abra este no vscode novamente, e vá até o diretório “api”:

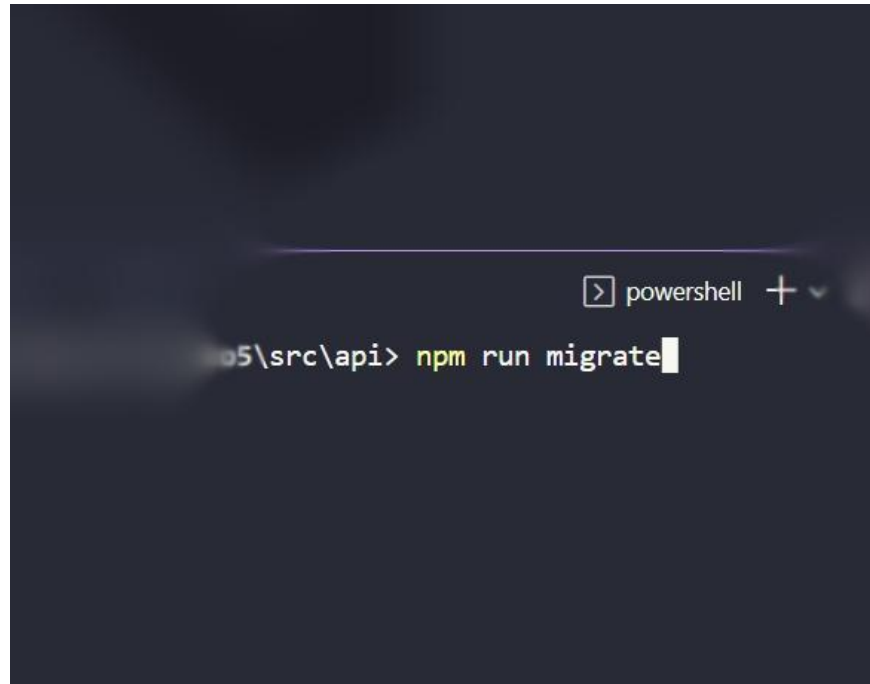
- Clique no lado direito do mouse sobre o diretório “api” e clique na parte de “Open in Integrated Terminal”;



- O terminal será aberto, digite neste o comando “npm i” de enter:



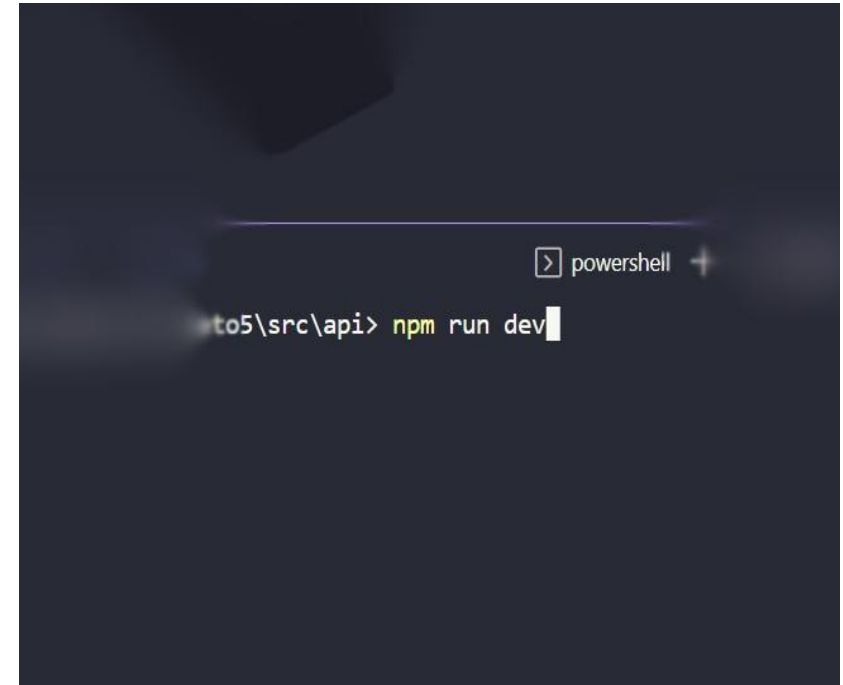
- Neste mesmo terminal escreva o comando "npm run migrate" e de enter:



```

> powershell +
to5\src\api> npm run migrate
  
```

- Após a execução do comando acima, digite "npm run dev" e de enter:



```

> powershell +
to5\src\api> npm run dev
  
```


6. Guia de Operação

Link para acessar o tutorial da aplicação web:

 Interface_Web.pdf

7. Troubleshooting

#	Problema	Possível solução
1	Se a bateria acabar.	Tem que haver a troca desta pela bateria de escolha do usuário.
2	Se o medidor de bateria falhar.	Refaça a seção 3 deste manual.
3	Se a rede WiFi falhar no dispositivo IoT usado para host.	Resetar o dispositivo ou checar a rede WiFi do IPT.

8. Créditos

Abaixo está a equipe e suas respectivas responsabilidades no projeto:

- **Júlia Togni** - Frontend e Documentação;
- **Henrique Marlon** - Frontend, Backend e Documentação;
- **Vitor Zeferino** - Frontend e Documentação;
- **Ariel Kisilevzky** - Documentação;
- **Vinícios Lugli** - Backend, Frontend, Microcontrolador e Documentação;
- **Melyssa Rojas** - Frontend, Microcontrolador e Documentação;