

**Controle do Documento**

**Histórico de revisões**

| **Data** | **Autor** | **Versão** | **Resumo da atividade** |
| --- | --- | --- | --- |
| 08/12/2023 | Kaylane Vasconcelos | 1.0 | Criação do documento e Preenchimento das seções 1 e 2. |
| 08/12/2023 | Victor Marques | 1.0 | Preenchimento das seções 3, 4, 5 e 6. |
| 21/12/2023 | Kaylane Vasconcelos | 2.0 | Ajustes finais. |
| 21/12/2023 | Victor Marques | 2.0 | Correção da seção 3. |

**Índice**

[**1. Componentes e Recursos 3**](#_heading=h.1t3h5sf)

[**1.1. Componentes externos 3**](#_heading=h.17dp8vu)

[**1.2. Requisitos de conectividade 4**](#_heading=h.3rdcrjn)

[**2. Guia de Montagem 7**](#_heading=h.26in1rg)

[**2.1 Equipamento e ferramentas Necessárias 7**](#_heading=h.26in1rg)

[**2.2 Componentes 7**](#_heading=h.26in1rg)

[**2.3 Passo a passo 9**](#_heading=h.26in1rg)

[**2.3.1 Módulo IoT 9**](#_heading=h.fel1p8bmfcgb)

[**2.3.2 Identificador RFID 9**](#_heading=h.j4zos3rkkvc6)

[**3. Guia de Instalação 11**](#_heading=h.35nkun2)

[**3.1 Instalação do Arduino IDE 11**](#_heading=h.pucyg2z6uloo)

[**3.2 Instalação do Código-Fonte 13**](#_heading=h.okqbkmddxyzv)

[**3.3 Abrir Código na IDE 14**](#_heading=h.cx61hhw9zqvj)

[**3.4 Ambiente Físico 16**](#_heading=h.990v3abhhky1)

[**4. Guia de Configuração 17**](#_heading=h.44sinio)

[**4.1 Conexão com Broker MQTT 17**](#_heading=h.cax4dis4f22)

[**4.1.1 Broker MQTT 17**](#_heading=h.r76yzh45rdse)

[**4.1.2 Configuração com AWS Core IoT 17**](#_heading=h.38vu0z3519hy)

[**4.1.3 Conexão com Cliente MQTT em Python 17**](#_heading=h.y2q7odxhu5t)

[**4.1.4 Cliente MQTT para Módulo Físico 18**](#_heading=h.h9hcfy5r95zm)

[**4.2 Aplicação Web 18**](#_heading=h.cji2lhs4pd6v)

[**4.2.1 Configuração do Front-end 18**](#_heading=h.rw331fh2pvw)

[**4.2.2 Configuração do Back-end 19**](#_heading=h.jla5df1gi4l8)

[**4.3 Banco de Dados 20**](#_heading=h.oqjsktxw74q7)

[**4.3.1 Criação de Conta no ElephantSQL 21**](#_heading=h.5lw4s4uhtv4q)

[**4.3.2 Criação de uma Instância no ElephantSQL 21**](#_heading=h.mx2k19k3m4kk)

[**4.6.3 Configuração do Banco de Dados PostgreSQL 21**](#_heading=h.exwysnalaoo6)

[**5. Guia de Operação 24**](#_heading=h.z337ya)

[**5.1 Cadastro de usuários 24**](#_heading=h.rfmk8c7jjgf0)

[**5.1.1 Processos internos 27**](#_heading=h.fn7pxbr1iiva)

[**5.2 Cadastro de peças 28**](#_heading=h.vsu0jbg8v17t)

[**5.2.1 Processos internos 29**](#_heading=h.5dmxwkjbsi6l)

[**5.3 Registro de Peças 30**](#_heading=h.e9tqdwifrhi1)

[**5.3.1 Processos internos 30**](#_heading=h.x9g5wqojtwg)

[**6. Troubleshooting 31**](#_heading=h.1y810tw)

# 1. Componentes e Recursos

## 1.1. Componentes externos

O grupo BIOT Solutions visa transformar a gestão operacional da Atvos por meio da implementação de uma solução inovadora. Integrando tecnologias como IoT, AWS Core IoT, PostgreSQL, ReactJS e NodeJS, a empresa poderá monitorar em tempo real e otimizar seu ciclo operacional. A plataforma em nuvem proposta processará dados intuitivamente, fornecendo *insights* para aprimorar a segurança, reduzir prejuízos e permitir decisões ágeis. Este projeto é um marco na modernização da Atvos, alinhado com sua visão de gestão eficiente e competitiva no setor agrícola.

Para o uso, implementação e funcionamento da solução, examinaremos agora os componentes externos essenciais que necessários para implementação desta solução.

1. **Dispositivos de Desenvolvimento e Acesso à Plataforma:**

Computadores e dispositivos móveis podem ser utilizados pelos usuários finais para acessar a plataforma, visualizar dados e interagir com a solução, como nossa plataforma é responsiva, o usuário terá a liberdade de escolher qual o melhor dispositivo para acessar a plataforma.

1. **Servidores na Nuvem:**

A AWS (Amazon Web Services) será utilizada na hospedagem da plataforma em nuvem, garantindo escalabilidade e disponibilidade. Serviços como EC2 para máquinas virtuais e S3 para armazenamento são alternativas que podem ser utilizadas.

1. **IDE (Ambiente de Desenvolvimento Integrado):**

O VSCode será o software de desenvolvimento usado no projeto, sendo a ferramenta principal para desenvolvimento de código-fonte, integração com repositórios, e depuração.

1. **Banco de Dados:**

Utilizamos o PostgreSQL para armazenamento de dados estruturados relacionados às operações, insumos, equipamentos e outros aspectos relevantes do ciclo operacional.

1. **Servidor de Banco de Dados Gerenciado:**

O ElephantSQL será empregado para gerenciar nosso banco de dados PostgreSQL, proporcionando facilidades como backup, escalabilidade e monitoramento.

1. **Front-end da Aplicação:**

O desenvolvimento do front-end será realizado utilizando ReactJS, HTML e CSS, com a adição da biblioteca ApexChartJS para a visualização de dados.

1. **Back-end da Aplicação:**

A lógica de negócios, manipulação de dados e a comunicação entre o front-end e o banco de dados serão gerenciadas pelo back-end, desenvolvido em NodeJS e AdonisJS.

1. **Cliente MQTT:**

Implementamos um cliente MQTT em Python para interação com o protocolo MQTT, responsável por coletar dados dos dispositivos IoT.

1. **Broker MQTT:**

O serviço AWS Core IoT será o broker MQTT responsável por gerenciar a comunicação entre os dispositivos IoT (sensores) e a plataforma.

## 1.2. Requisitos de conectividade

A eficácia de qualquer solução IoT reside não apenas na inovação dos dispositivos, mas também na robustez da infraestrutura que sustenta a comunicação e o processamento de dados. Neste contexto, a criação de uma rede eficiente e segura, juntamente com especificações adequadas de backend, é crucial para garantir o funcionamento fluido dos dispositivos IoT propostos pela Atvos. Vamos explorar os requisitos de rede, protocolos e as especificações de backend necessárias para criar uma base sólida que permitirá a coleta, processamento e análise eficientes de dados, contribuindo assim para o sucesso da solução inovadora da Atvos.

1. **Rede Local (LAN):**

A solução requer uma rede local para a comunicação eficiente entre os dispositivos IoT, o servidor de backend e outros componentes da infraestrutura. A rede local será fundamental para essa comunicação, pois permitirá uma troca rápida e confiável de dados no ambiente controlado da Atvos.

1. **Protocolos:**

* Protocolo MQTT (Message Queuing Telemetry Transport): O protocolo MQTT será fundamental para a comunicação assíncrona entre os dispositivos IoT e o broker MQTT, permitindo a troca eficiente de mensagens.
* Protocolo HTTPS (Hypertext Transfer Protocol Secure): Para garantir a segurança na transmissão de dados pela internet, a solução utilizará HTTPS, criptografando as comunicações entre o cliente e o servidor.
* Protocolo TCP/IP: O Protocolo de Controle de Transmissão/Protocolo de Internet (TCP/IP) é o conjunto de protocolos que define a comunicação na internet e em muitas redes privadas, estabelecendo conexões e garantindo a comunicação eficiente entre os dispositivos IoT, o front-end e o servidor de backend.
* Protocolo IPv4:O Protocolo da Internet versão 4 (IPv4) é a versão mais amplamente utilizada do protocolo IP. Ele fornece identificação e localização para dispositivos em uma rede, atribuindo endereços IP únicos a cada dispositivo conectado.

1. **Axios:**

Axios é uma biblioteca JavaScript baseada em Promise que facilita a realização de solicitações HTTP em navegadores e no Node.js. É amplamente utilizado para fazer chamadas a APIs, enviar dados ao servidor e receber respostas, tornando-se uma escolha comum na construção de aplicações cliente-servidor.

1. **Bibliotecas externas do protótipo físico:**

* WiFiClientSecure.h e WiFi.h: Conecta o dispositivo a uma rede Wi-Fi e a um servidor MQTT seguro. Isso permite que os dispositivos IoT se comuniquem com a plataforma em nuvem de forma segura.
* MQTTClient.h: Implementa a comunicação MQTT entre os dispositivos e a plataforma em nuvem. Enviando dados coletados, recebendo comandos e atualizações, e garantindo a comunicação eficiente entre os dispositivos e a nuvem.
* ArduinoJson.h: Estrutura e manipula os dados coletados antes de enviá-los para a plataforma em nuvem. Isso é útil para garantir que as informações sejam formatadas corretamente antes do envio.
* time.h: Obtém as informações de data e hora em tempo real. Marcando eventos, criando registros temporais e garantindo que as informações coletadas estejam devidamente carimbadas no tempo.
* SPI.h: Ao usar periféricos que se comunicam via SPI, como sensores ou dispositivos de armazenamento, pode-se utilizar esta biblioteca para configurar e realizar a comunicação.
* MFRC522.h: Esta biblioteca será fundamental para interagir com o leitor MFRC522 ao implementar controle de acesso ou rastreamento de ativos usando cartões RFID.
* Wire.h: Estabelece a comunicação I2C entre os dispositivos.
* LiquidCrystal\_I2C.h: Esta biblioteca será útil para controlar o display de forma eficiente.

1. **Bibliotecas externas do Cliente em python:**

* google.auth.transport.requests e google.oauth2.credentials.Credentials: São usadas para autenticar e obter credenciais para acessar serviços do Google, como o Google Cloud.
* google\_auth\_oauthlib.flow.InstalledAppFlow e googleapiclient.discovery.build: São usadas para autorizar o acesso a APIs do Google.
* paho.mqtt.client: Utilizada para implementar um cliente MQTT em Python.
* ssl: Usada para fornecer suporte a conexões seguras, o que pode ser relevante ao trabalhar com comunicação segura MQTT (por exemplo, MQTT sobre TLS/SSL).
* json: Usada para manipular dados no formato JSON.
* psycopg2: Utilizada para se conectar a bancos de dados PostgreSQL.

1. **Dependências do Back-End:**

* node v18.15.0: Esta é a versão do Node.js que o projeto utiliza.
* @adonisjs/core, @adonisjs/lucid, @adonisjs/repl: Bibliotecas do AdonisJS, um framework web para Node.js.
* luxon: Utilizada para manipulação de datas e horas em JavaScript.
* pg: Biblioteca Node.js para interagir com o banco de dados PostgreSQL.
* proxy-addr: Descobre endereços IP de clientes quando o servidor está atrás de um proxy reverso.
* reflect-metadata: Usada para habilitar a reflexão de metadados em TypeScript.
* source-map-support: Ferramenta que melhora o suporte a source maps em Node.js.

# 2. Guia de Montagem

## 2.1 Equipamento e ferramentas Necessárias

1. Ferro de solda e solda
2. Chave de fenda
3. Alicate de corte
4. Impressora 3D
5. Parafusos
6. Computador para programação do microcontrolador

Figura 1: Equipamentos e ferramentas Necessárias

## 

## 

## 

## 2.2 Componentes Necessários

Todos os componentes necessários para a montagem completa do protótipo físico, incluindo a quantidade para cada um deles e suas especificações estão presentes na tabela abaixo (Tabela 1).

Tabela 1: Componentes necessários

| N° | Quant. | Componentes | Especificação |
| --- | --- | --- | --- |
| 1 | 2 | Display LCD | 16x2 com módulo I2C |
| 2 | 2 | Bateria | Lítio ICR18650 3,7V 2700mAH |
| 3 | 2 | Suporte para a bateria | N/A |
| 4 | 2 | Chave Alavanca | 2 posições - MTS 102 |
| 5 | 2 | Led RGB | Catodo comum |
| 6 | 1 | Suporte para o Led | N/A |
| 7 | 4 | Parafuso | M2 |
| 8 | 1 | Conector Barra de Pinos | 1x40, fêmea, 180° |
| 9 | 21 | Fios Jumper | Macho-Macho e Macho-Fêmea |
| 10 | 2 | Sensor RFID | N/A |
| 11 | 2 | Microcontrolador ESP32 | N/A |
| 12 | 1 | Antena WiFi | N/A |
| 13 | 1 | Placa de Circuito | PCB |
| 14 | 1 | Protoboard | N/A |
| 15 | 1 | Case | Impressão 3D |

Figura 2: Componentes Necessários

\*Os números da figura acima (Figura 2) possuem correspondência com o N° que está na primeira coluna da tabela de componentes.

## 2.3 Passo a passo

### 2.3.1 Módulo IoT

1. Montagem do Case:

* Imprima o case em 3D segundo o modelo presente [neste link](https://drive.google.com/drive/folders/1MNjZIgHnQrbCnm6ZMYnhS3chFTVlYcTW?usp=sharing)(os arquivos .stl são os que a impressora 3d lê e os arquivos .iam e .ipt são usados para criar o modelo 3d).
* Instale o suporte e tampa para a bateria no interior do case, na figura 3, podemos observar como fica o resultado final desse processo.

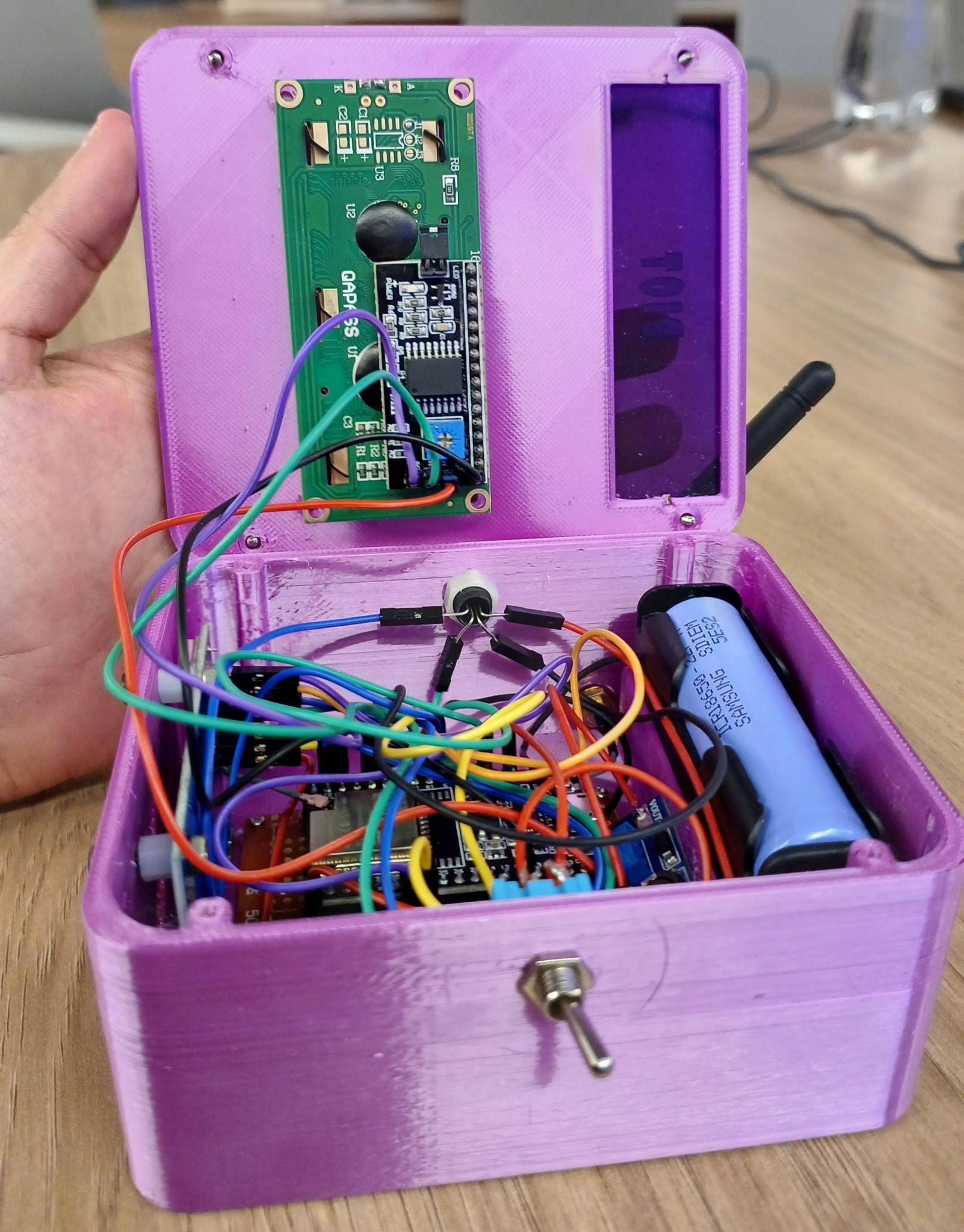
Figura 3: Case BIOT



1. Montagem do Display LCD:

* Conecte os Displays LCD aos Microcontroladores ESP32 usando os conectores barra de pinos.
* Fixe o Display no case utilizando parafusos M2, de acordo com a figura 4.
* Conecte os fios jumper Macho-Fêmea para estabelecer as conexões necessárias.

Figura 4: Display LCD



1. Integração da Bateria:

* Posicione a bateria no suporte, assim como mostra a figura 5.
* Conecte os terminais positivo e negativo da bateria aos pinos de alimentação correspondentes nos Microcontroladores ESP32. Geralmente, esses pinos são rotulados como VCC (positivo) e GND (negativo).

Figura 5: Bateria



1. Integração do LED:

* Para o LED, conecte o pino positivo (ânodo) a um pino de saída digital do Microcontrolador ESP32. Conecte o pino negativo (catodo) ao GND do microcontrolador.

Figura 6: LED



1. Integração da Chave Alavanca:

* Para a Chave Alavanca presente na figura 7, conecte um dos terminais ao GND e o outro ao pino de entrada digital do Microcontrolador ESP32.

Figura 7: Chave Alavanca



1. Integração do Sensor RFID:

* Conecte os pinos de comunicação do sensor RFID, presente na figura 8 (por exemplo, RX e TX) aos pinos de comunicação correspondentes nos Microcontroladores ESP32.
* Conecte os pinos de alimentação (VCC e GND) do sensor RFID aos pinos correspondentes no Microcontrolador ESP32.

Figura 8: Sensor RFID



1. Integração da Antena WiFi

* Conecte os fios da antena WiFi aos pinos de antena no Microcontroladores ESP32.
* Fixe o Microcontrolador ESP32 no case utilizando parafusos M2, assim como mostra a figura abaixo (Figura 9)

Figura 9: Antena WiFi



1. Conexões e Soldagem:

* Utilize o ferro de solda para realizar as soldagens necessárias nas placas de circuito impresso (PCBs), assim como demonstra a figura 10.
* Use o multímetro para verificar a continuidade e a correta conexão de todos os componentes.

Figura 10: Visão Interna do Protótipo



### 2.3.2 Identificador RFID

1. Organização dos Componentes:

* Disponha todos os componentes em uma superfície limpa.
* Certifique-se de ter todos os fios jumper necessários.

1. Conexão do Sensor RFID:

* Conecte os pinos do Sensor RFID à protoboard, seguindo o datasheet do sensor.
* Geralmente, os pinos incluem SDA, SCK, MOSI, MISO, IRQ, GND, e RST.
* Conecte o Sensor RFID ao ESP32 usando fios jumper.

1. Conexão do Display LCD:

* Conecte o display LCD à protoboard utilizando fios jumper.
* Verifique o datasheet do display para identificar os pinos necessários.
* Conecte o display LCD ao ESP32 usando o protocolo I2C.

1. Conexão do LED RGB:

* Conecte os pinos do LED RGB à protoboard.
* Normalmente, os LEDs RGB têm pinos para R (vermelho), G (verde), B (azul), e GND.
* Conecte os pinos do LED RGB ao ESP32 usando fios jumper.

1. Conexão do ESP32:

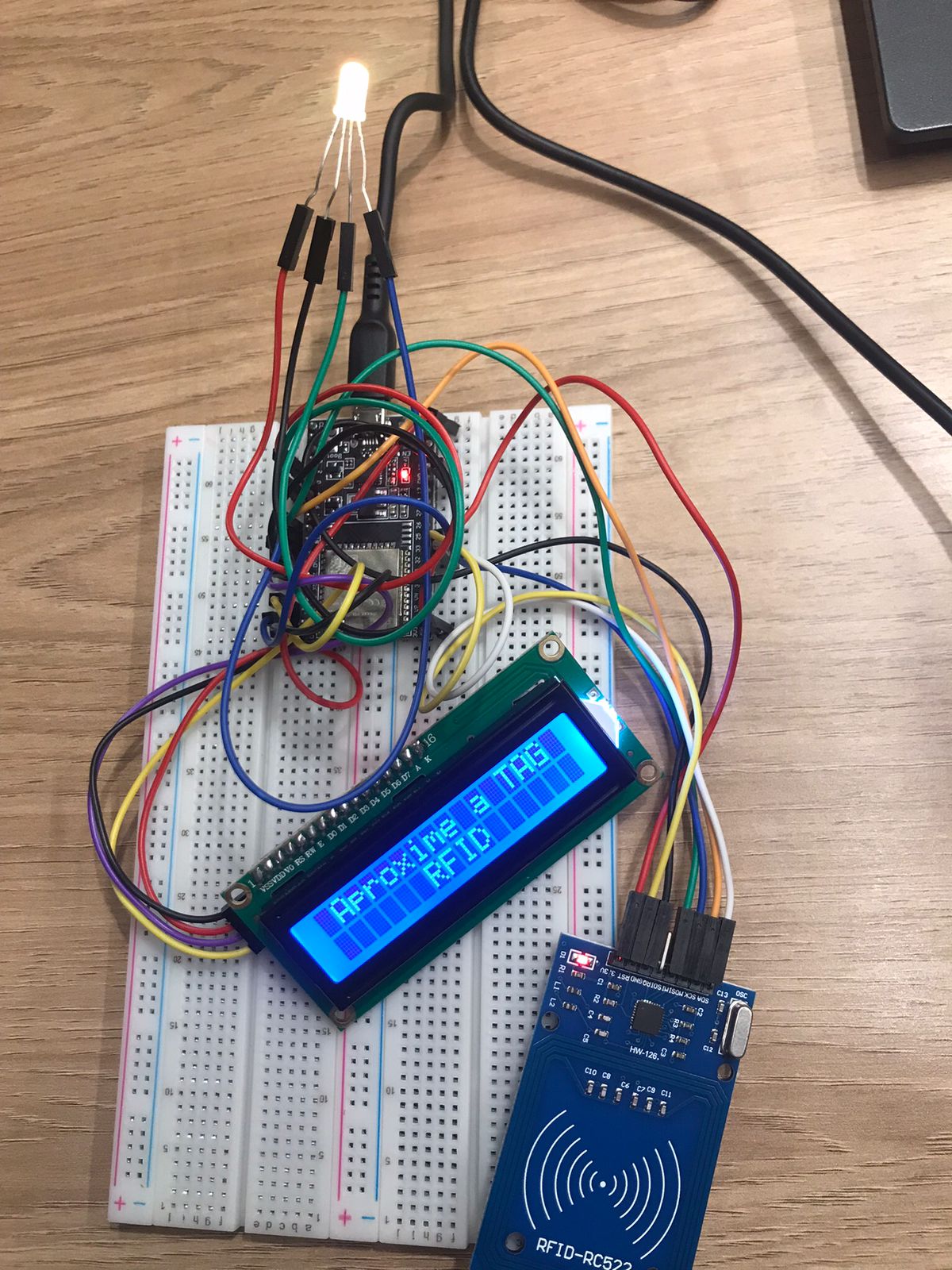
* Conecte o ESP32 à protoboard.
* Conecte os pinos necessários do ESP32 aos componentes (Sensor RFID, Display LCD, LED RGB) usando fios jumper.
* Conecte o ESP32 ao computador para programação, se necessário.

1. Conexões de Energia:

* Alimente o ESP32 e o display LCD usando a protoboard.
* Alimente o Sensor RFID e o LED RGB conforme necessário.
* Certifique-se de que todas as polaridades estão corretas.

1. Montagem Física:

* Posicione os componentes de forma organizada na protoboard, seguindo o exemplo abaixo (Figura 11)
* Fixe o ESP32, o display LCD e o Sensor RFID com cuidado.

Figura 11: Identificador RFID

# 3. Guia de Instalação

Precedentemente, este guia de instalação evidencia o procedimento para instalar a solução IoT nos ambientes necessários (almoxarifado e caminhões oficinas). Para esta etapa, são necessários os seguintes requisitos:

* Guia de montagem foi realizado;
* Internet local é acessível;
* Há uma bateria como fonte de disponível;
* Há um computador/notebook disponível.

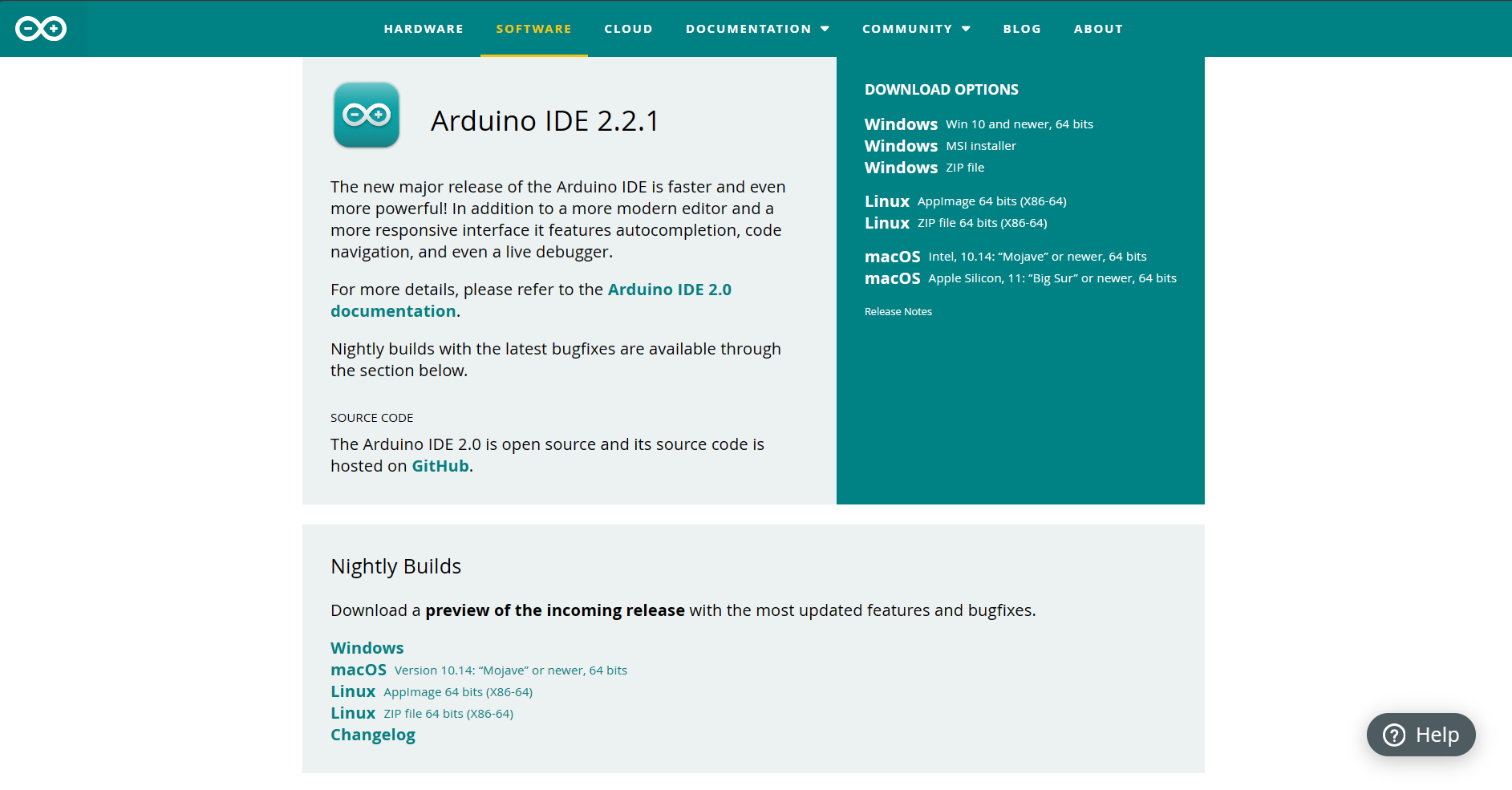
Destarte, considerando que a montagem da solução física foi realizada, consoante ao guia de montagem citado no tópico anterior, é necessário instalar o código-fonte da solução e, em seguida, carregá-lo no microcontrolador Esp32

Portanto, para realizar a instalação do código do sistema, faz-se necessário o uso de um computador/notebook com acesso à internet, bem como uma bateria carregada para a alimentação do sistema. Dessa forma, as etapas a seguir elucidam todo o processo de instalação inicial da solução IoT.

## 3.1 Instalação do Arduino IDE

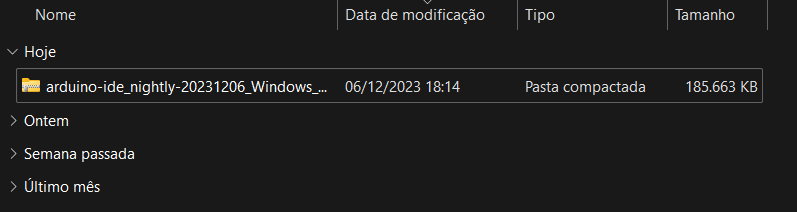
De antemão, é necessário, para o carregamento do código ao microcontrolador, uma interface intermediária que auxilie nesse carregamento. Por conseguinte, recomenda-se o uso de uma IDE (Ambiente de Desenvolvimento Integrado), neste caso, do Arduino IDE, cuja página de instalação está evidenciada na figura 12 a seguir.

Figura 12 - Página de download da ferramenta Arduino IDE



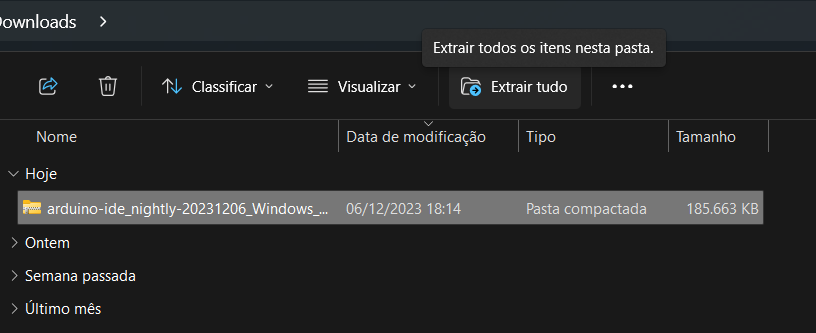
Sendo assim, basta realizar o download de acordo com o sistema operacional do computador/notebook que está sendo utilizado. Quando finalizado, a ferramenta de instalação estará disponível na pasta de downloads do dispositivo, como demonstra a figura abaixo (Figura 13).

Figura 13 - Ferramenta instalada e localizada na pasta de downloads do computador



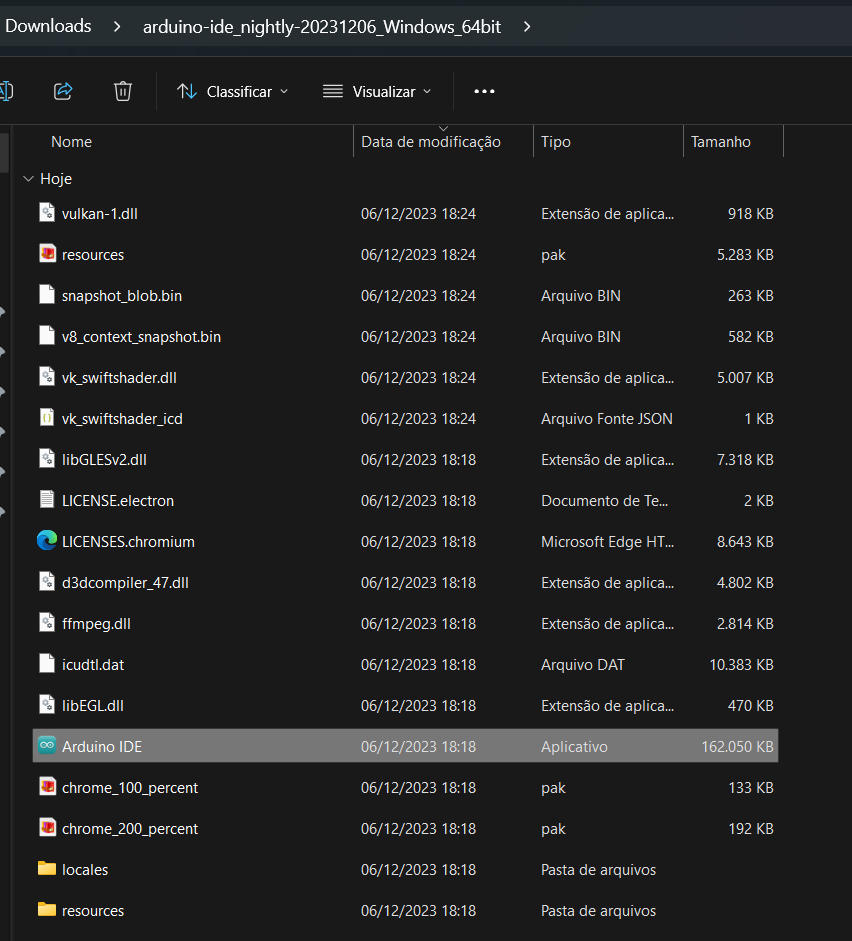
Com isso, faz-se necessário a extração da ferramenta, haja vista que, por padrão, está como pasta compactada. No windows 11, é possível extrair com facilidade, de modo que é preciso somente clicar sobre a pasta e, logo em seguida, clicar em “Extrair tudo” na barra superior. A título de exemplo, a figura 14 mostra esse procedimento.

Figura 14 - Extração da pasta compactada no windows 11



Desse modo, após a extração ser finalizada (o que leva, a depender do dispositivo, cerca de 5 minutos), basta clicar duas vezes sobre a pasta criada e, em seguida, iniciar o aplicativo, como demonstrado na figura abaixo (Figura 15).

Figura 15 - Aplicativo Arduino IDE disponível na pasta descompactada

****

Ao ser executado, o aplicativo abrirá uma janela semelhante a esta demonstrada na figura abaixo (Figura 16).

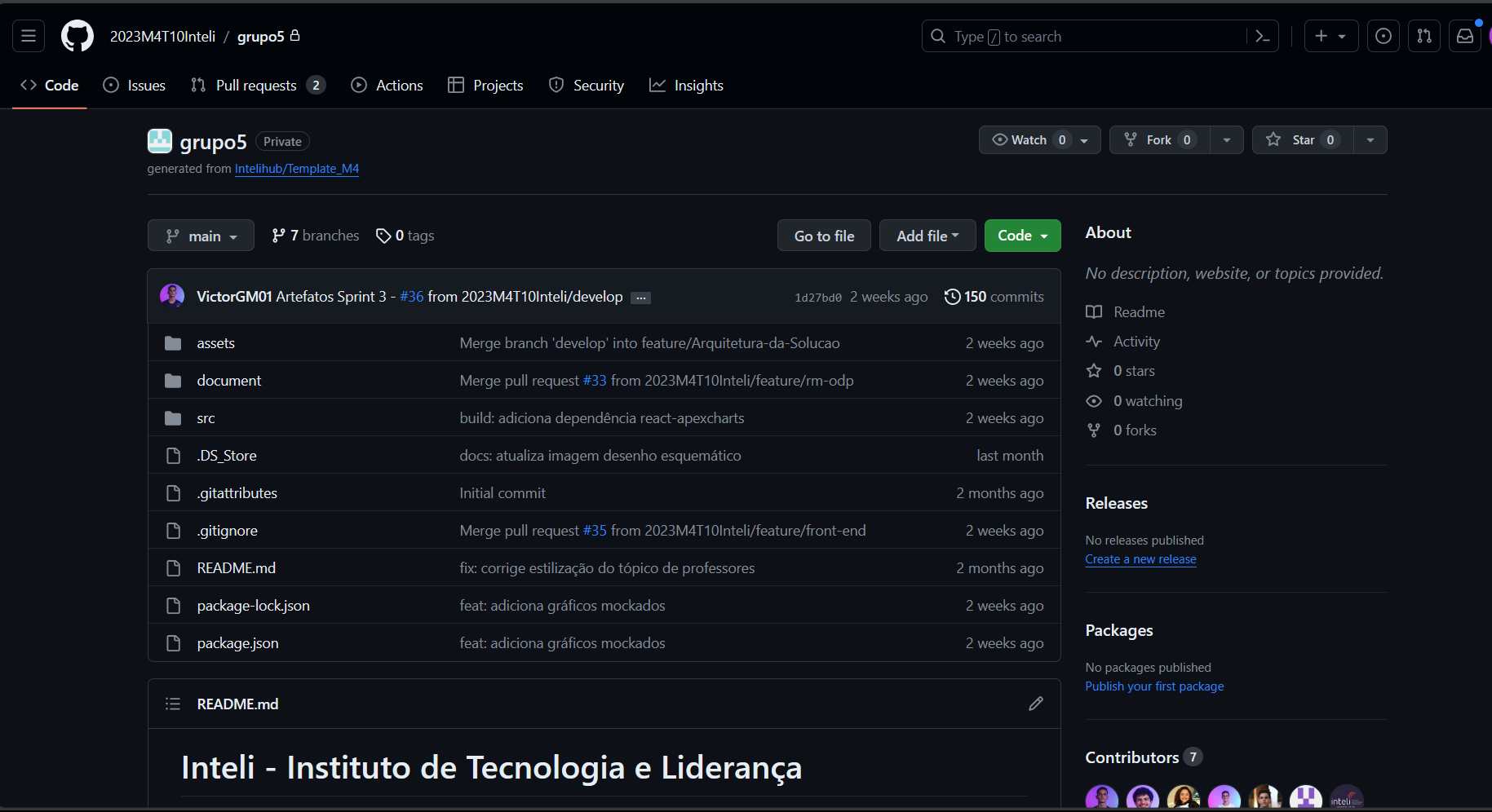
Figura 16 - Arduino IDE iniciado e realizando download dos drivers básicos de uso

****

## 3.2 Instalação do Código-Fonte

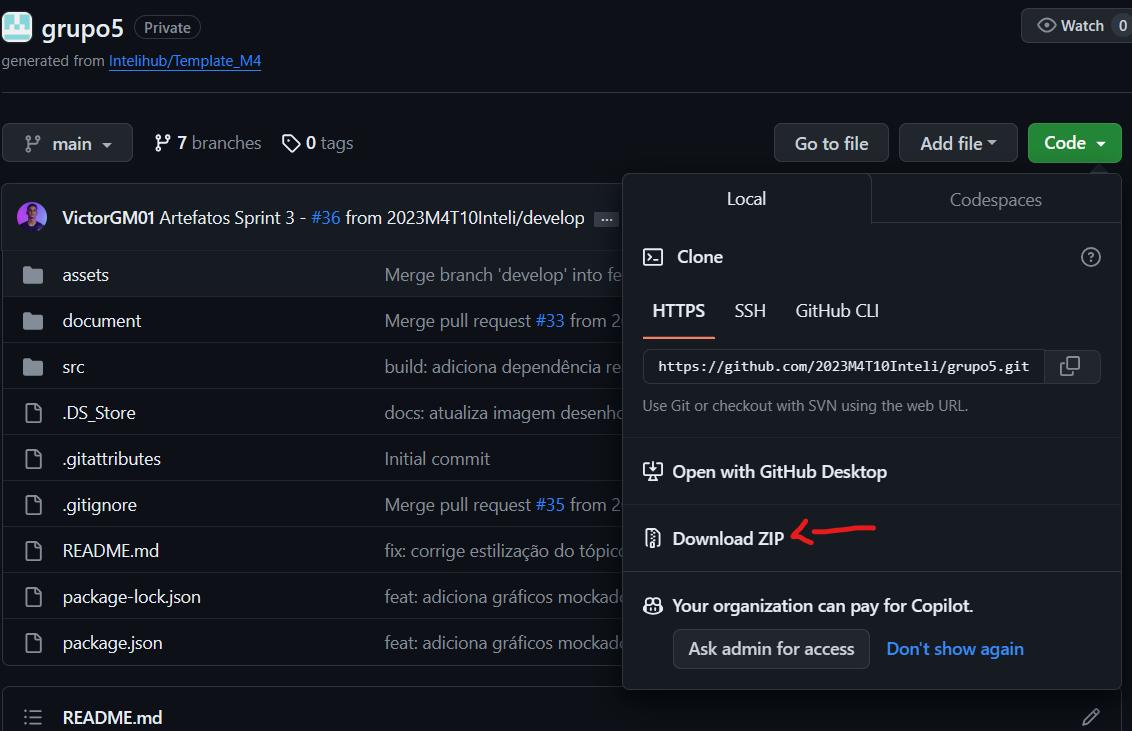
O próximo passo é baixar o código-fonte do projeto para o computador utilizado durante todas as etapas anteriores. Para isso, o código pode ser encontrado através do seguinte link: [2023M4T10Inteli/grupo5 (github.com)](https://github.com/2023M4T10Inteli/grupo5)**.** Dessa forma, ao acessar o link a página do GitHub (onde ficam armazenados os arquivos de código e documentação) do projeto será aberta, a qual terá a seguinte aparência (figura 17).

Figura 17 - Respositório no GitHub do projeto



Com o repositório aberto, basta clicar em Code (botão verde) e, em seguida, em “Download ZIP”, como exposto na figura a seguir. Este processo iniciará algo semelhante à instalação do Arduino IDE, de modo que resultará numa pasta compactada.

Figura 18 - Local de instalação do código-fonte do projeto via GitHub

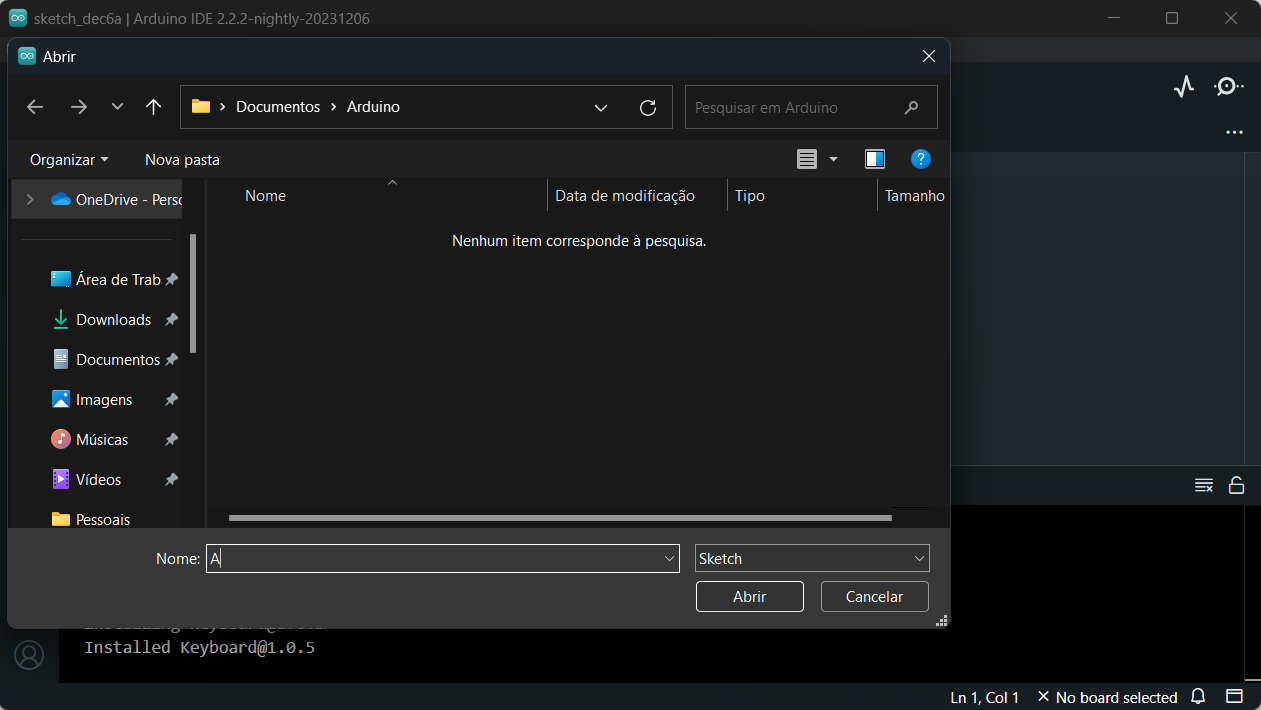


Com o download realizado, recomenda-se a descompactação da pasta gerada (semelhante ao processo de extração da pasta instalada na etapa do Arduino IDE) e, após isso, é possível voltar à IDE para abrir o código.

## 3.3 Abrir Código na IDE

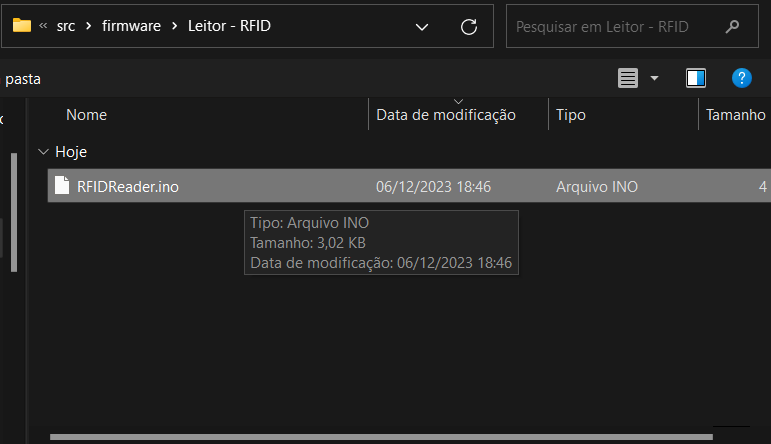
Destarte, com a IDE aberta, basta clicar em “File” (ou “Arquivo”) e, em seguida, em “Open” (ou “Abrir”). Um atalho para tal procedimento é clicar nas teclas “Ctrl” e “O” ao mesmo tempo, de modo que o resultado final é o mesmo: será aberta uma janela de localização de pastas no gerenciador de arquivos do dispositivo. No caso deste guia, a janela aberta está demonstrada na figura seguinte (Figura 19). Vale ressaltar que o sistema operacional usado neste guia é o Windows 11.

Figura 19 - Janela para abrir arquivos/pastas através da IDE



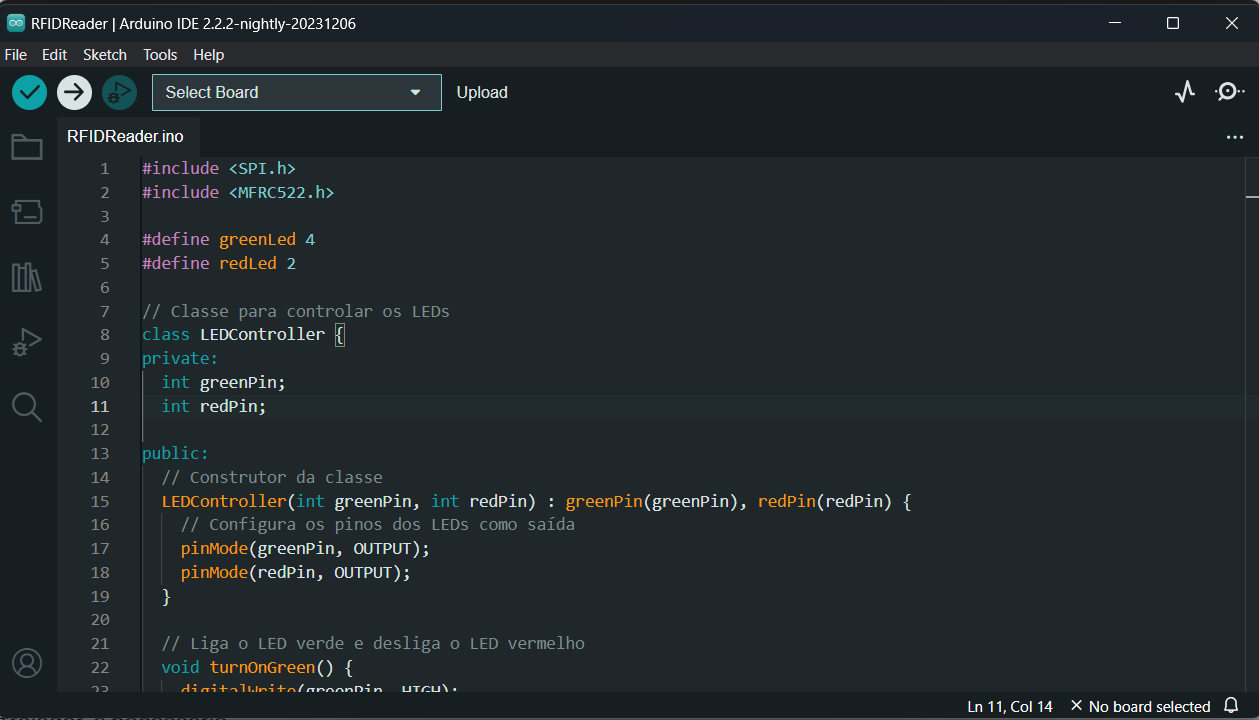
Nesse ponto, é necessário localizar a pasta descompactada na etapa anterior, clicar sobre a subpasta “src”, depois na subpasta “firmware”, na subpasta “Leitor - RFID” e, então, clicar duas vezes sobre o arquivo **RFIDReader.ino**, conforme exposto na figura 20.

Figura 20 - Arquivo RFIDReader.ino localizado em src>firmware>Leitor - RFID>



Nessa etapa, o código estará visível no computador/notebook utilizado e, agora, é necessário conectar o Esp32 ao computador por meio de um cabo USB e, em seguida, quando a IDE reconhecer o microcontrolador, é necessário clicar sobre o botão de “Upload”, na barra superior da IDE, como demonstra a figura a seguir (Figura 21).

Figura 21 - Botão de upload (ícone de seta para a direita circulado) clicado



Tal ação carregará o código para o microcontrolador e, em seguida, será necessário repetir o procedimento para o código responsável pelo envio de requisições via protocolo MQTT. Este, por sua vez, está acessível através do seguinte caminho: src > firmware > MQTT\_AWS\_RFID. O nome do arquivo é: **MQTT\_AWS\_RFID.ino**. Portanto, é necessário, novamente, clicar em File, Open e localizar tal arquivo, de modo a fazer o upload do código ao devido microcontrolador. Por fim, com o código na memória do Esp32, já é possível instalar o módulo completo, com a caixa protetora, nos estoques sobre rodas (caminhões oficinas e nos almoxarifados).

## 3.4 Ambiente Físico

Com o módulo montado e com os devidos códigos já na memória do Esp32, é possível levá-lo até o local de instalação. O módulo de registro de responsabilidades (principal solução do projeto) deve ser acoplado na parede do almoxarifado ou na parede do caminhão oficina, de modo que as peças possam ser aproximadas por baixo da caixa protetora, com um contato facilitado com o sensor RFID. Sendo assim, a figura a seguir (Figura 22) simula a instalação do módulo em uma parede, exemplificando sua utilização final.

No exemplo da figura, o módulo está solicitando que o usuário conecte a sua credencial de acesso (cartão com tag RFID de identificação do usuário no sistema). Tal módulo ficará, preferencialmente, como a imagem: acoplado numa parede, quer seja uma parede de caminhão quer seja uma parede de um almoxarifado. O sensor RFID está localizado na região inferior da caixa, visando facilitar a aproximação dos cartões de identificação, bem como as peças grandes (pneus, baterias e motores).

Figura 22 - Módulo IoT acoplado em uma parede de exemplo para simular sua instalação



Por fim, é válido destacar as limitações dos componentes, sobretudo no que tange ao Sensor RFID, haja vista que, tanto no almoxarifado quanto no caminhão oficina, a distãncia máxima permitida entre a TAG e o sensor é de 10 cm. Nesse sentido, recomenda-se, para um futura implementação final, a substituição do sensor atual por outro com um alcance maior, facilitando a aproximação das peças grandes, como pneus e motores, por exemplo.

# 4. Guia de Configuração

## 4.1 Conexão com Broker MQTT

A integração eficiente entre dispositivos em um sistema IoT depende, em grande parte, de uma arquitetura de comunicação robusta. Nesse contexto, o Broker MQTT (Message Queuing Telemetry Transport) desempenha um papel crucial, atuando como o intermediário que possibilita a troca assíncrona de mensagens entre dispositivos conectados. Esta seção explora a configuração do Broker MQTT no âmbito do projeto BIOT Solutions, abordando desde a definição do que é um Broker MQTT até a sua conexão com Clientes MQTT específicos.

### 4.1.1 Broker MQTT

Um Broker MQTT, na arquitetura de Internet das Coisas (IoT), é um componente vital que facilita a comunicação entre dispositivos por meio do paradigma de publicação e subscrição de tópicos. Sua função principal é gerenciar a troca de mensagens assíncronas entre os diversos nós da rede, promovendo uma comunicação eficiente e escalável.

### 4.1.2 Configuração com AWS Core IoT

A escolha do AWS Core IoT para configurar o Broker MQTT oferece uma infraestrutura sólida, escalável e segura. Esta plataforma, fornecida pela Amazon Web Services, é projetada especificamente para atender às demandas da IoT. A configuração no AWS Core IoT inclui a definição de tópicos, políticas de segurança e outras configurações essenciais.

### 4.1.3 Conexão com Cliente MQTT em Python

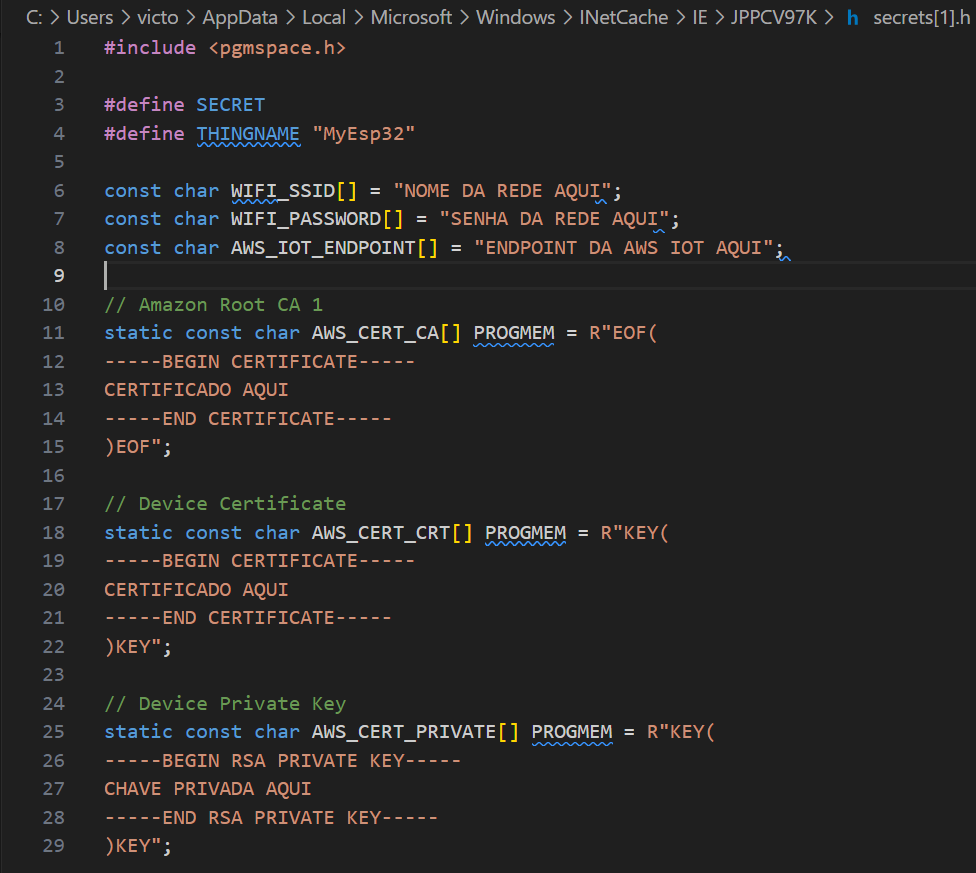
A integração do Broker com um Cliente MQTT em Python é crucial para a comunicação eficiente no ecossistema IoT. O Cliente MQTT atua como um ouvinte dedicado, processando cada requisição proveniente dos dispositivos conectados. Essa abordagem possibilita uma comunicação fluida e confiável entre os dispositivos e o sistema central.

Para garantir uma conexão segura, o Cliente MQTT em Python requer informações essenciais, como o endpoint do Broker e os certificados de segurança. A correta configuração desses parâmetros é vital para assegurar a autenticidade e integridade da comunicação. Certificados, chaves e outros detalhes são cuidadosamente configurados para garantir a segurança da transmissão de dados. Portanto, para realizar tais configurações recomenda-se a abertura do arquivo localizado em **src > api > AlmoxarifadoAPI.ipynb,** nos locais onde estão comentados (#CONFIGURAR AQUI), cada comentário desse antecede a explicação de como realizar tal configuração.

### 4.1.4 Cliente MQTT para Módulo Físico

Além do Cliente MQTT em Python, um Cliente MQTT dedicado ao módulo físico do sistema é responsável por adquirir dados do sensor RFID. Tal cliente está localizado em **src > firmware > MQTT\_AWS\_RFID.**. Este arquivo contém informações sensíveis, como credenciais de rede e certificados de segurança, essenciais para a integridade do fluxo de dados. A figura abaixo demonstra como o arquivo secrets.h deve ser criado:

Figura 23 - Exemplo de arquivo Secrets.h para configuração do cliente MQTT



## 4.2 Aplicação Web

No que tange à aplicação web, são necessários alguns procedimentos para que tanto o front-end quanto o back-end da aplicação rodem em perfeito estado de funcionamento. Dito isso, para ambos os casos, recomenda-se o uso da IDE VSCode para a edição do código.

A aplicação web desempenha um papel crucial no sistema IoT BIOT Solutions, proporcionando uma interface amigável para os usuários interagirem com os dados coletados e gerenciarem as operações de maneira eficiente. A seguir, estão delineados os passos para configurar tanto o front-end quanto o back-end da aplicação.

### 4.2.1 Configuração do Front-end

O front-end da aplicação é desenvolvido utilizando ReactJS, HTML e CSS, com a adição da biblioteca ApexChartJS para a visualização de dados. Siga os passos abaixo para configurar o front-end:

1. **Instalação do Node.js:**

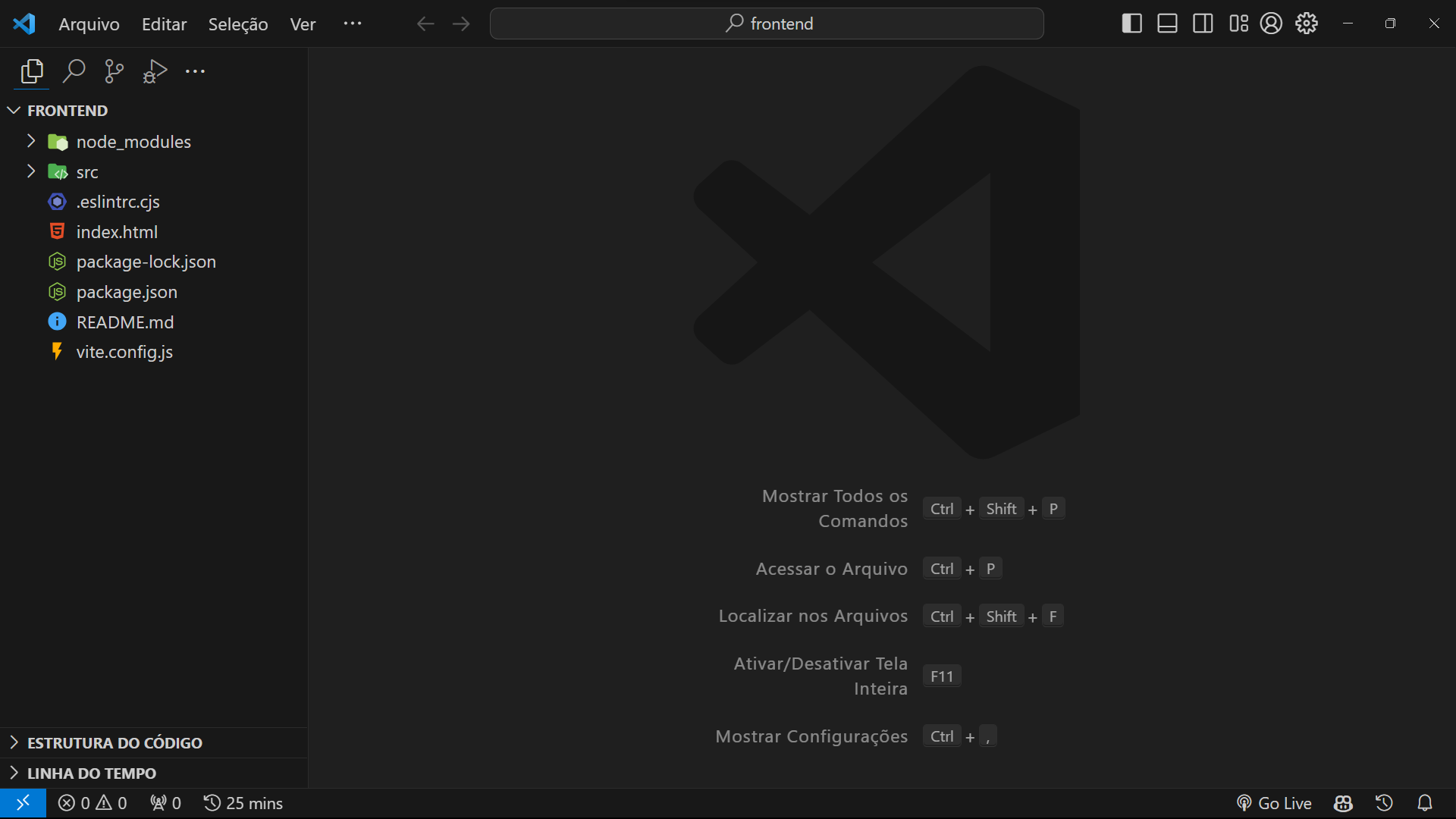
Antes de prosseguir, é necessário instalar o Node.js. Faça o download da versão LTS do Node.js em [Node.js (nodejs.org)](https://nodejs.org/en) (Figura 24) e siga as instruções de instalação para o seu sistema operacional.

Figura 24 - Página de download do Node.js usado na aplicação web

1. **Abertura do Projeto no VSCode:**

Abra o VSCode e clique em "File" (Arquivo). Selecione "Open Folder" (Abrir Pasta) e navegue até o local onde o código-fonte do front-end está armazenado. Neste caso, na pasta do código baixado do github no guia de instalação, dentro da pasta **src > frontend**.

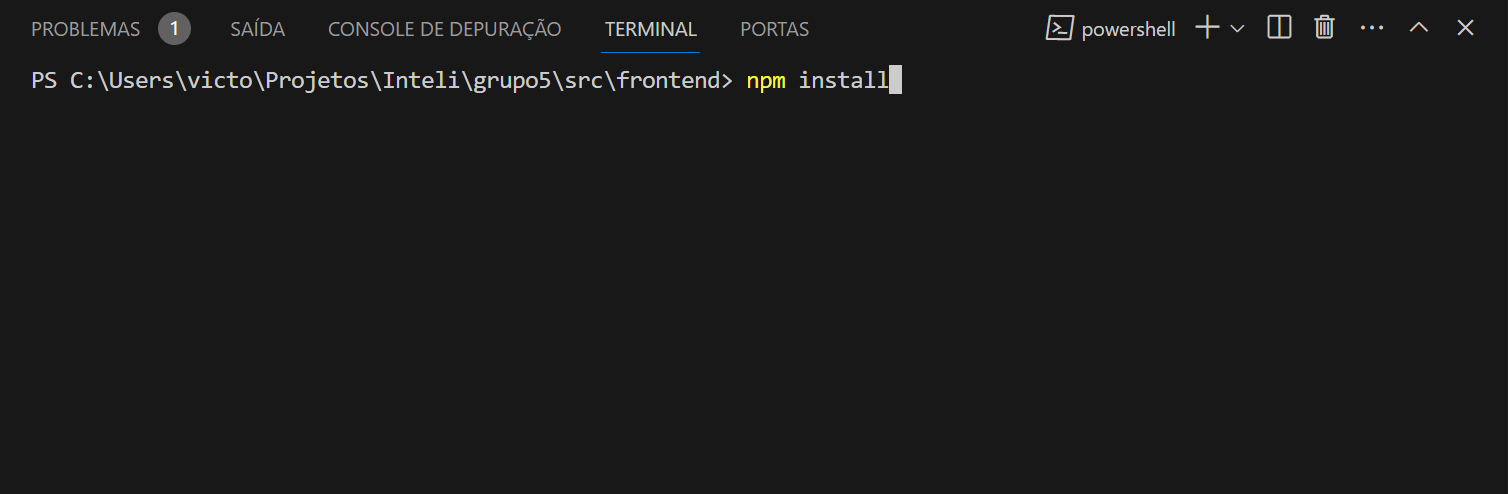
Figura 25 - Pasta do front-end aberta no VSCode



1. **Instalação de Dependências:**

Abra o terminal no VSCode (aperte as teclas “ctrl” e “j”) e execute o comando **npm install** para instalar as dependências necessárias definidas no arquivo **package.json**.

Figura 26 - Instalação das dependências do front-end no terminal



1. **Execução do Front-end:**

Utilize o comando **npm run dev** para iniciar o servidor de desenvolvimento do ReactJS. O front-end estará acessível em um navegador no endereço **http://localhost:5173**.

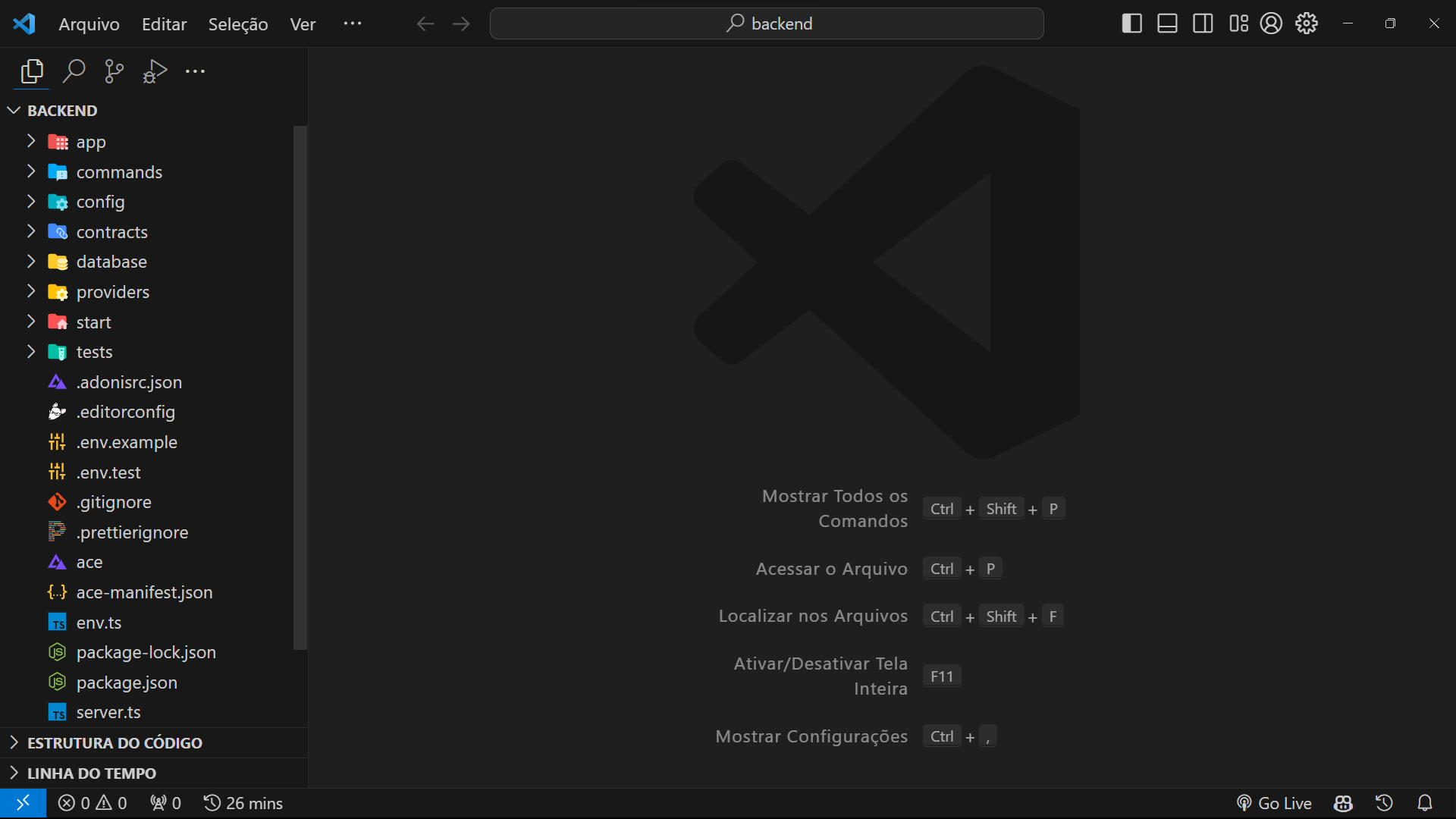
### 4.2.2 Configuração do Back-end

O back-end, desenvolvido em NodeJS e AdonisJS, é responsável pela lógica de negócios, manipulação de dados e comunicação entre o front-end e o banco de dados. Siga os passos a seguir para configurar o back-end:

1. **Abertura do Projeto no VSCode:**

Similar ao front-end, abra o VSCode e selecione "Open Folder" (Abrir Pasta), navegando até o local do código-fonte do back-end (**src > backend**).

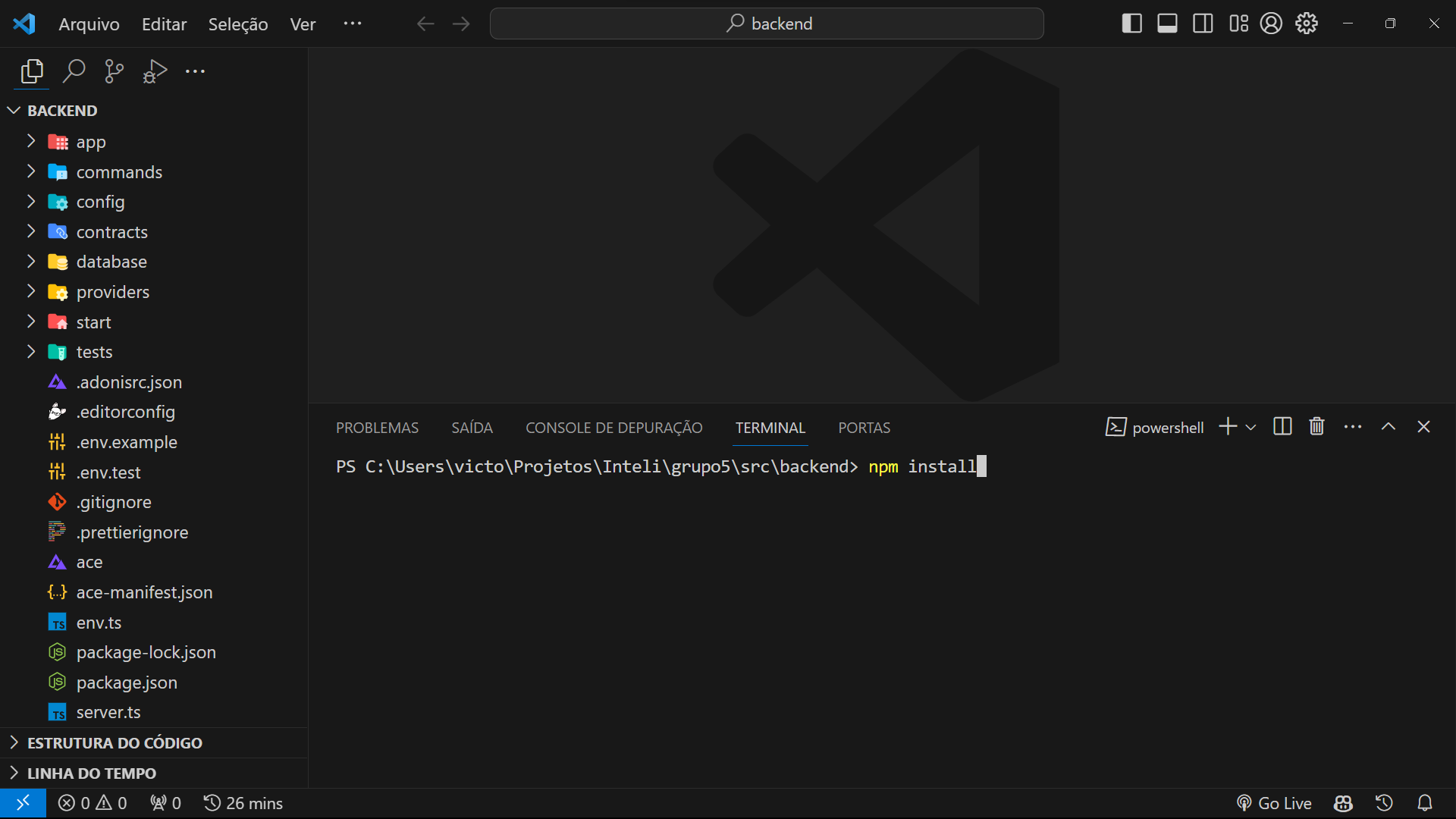
Figura 27 - Pasta do back-end aberta no VSCode



1. **Instalação de Dependências:**

Abra o terminal no VSCode (aperte as teclas “ctrl” e “j”) e execute o comando **npm install** para instalar as dependências necessárias definidas no arquivo **package.json**.

Figura 28 - Instalação das dependências do back-end no terminal



1. **Execução do Back-end:**

Utilize o comando **node ace serve --watch** para iniciar o servidor do AdonisJS. O back-end estará acessível em [**http://localhost:3333**](http://localhost:3333)**.**

## 4.3 Banco de Dados

Configurar o banco de dados PostgreSQL e o serviço ElephantSQL envolve várias etapas, desde a criação de uma conta até a definição de tabelas essenciais para o funcionamento do sistema. Siga as instruções detalhadas abaixo para garantir uma configuração adequada.

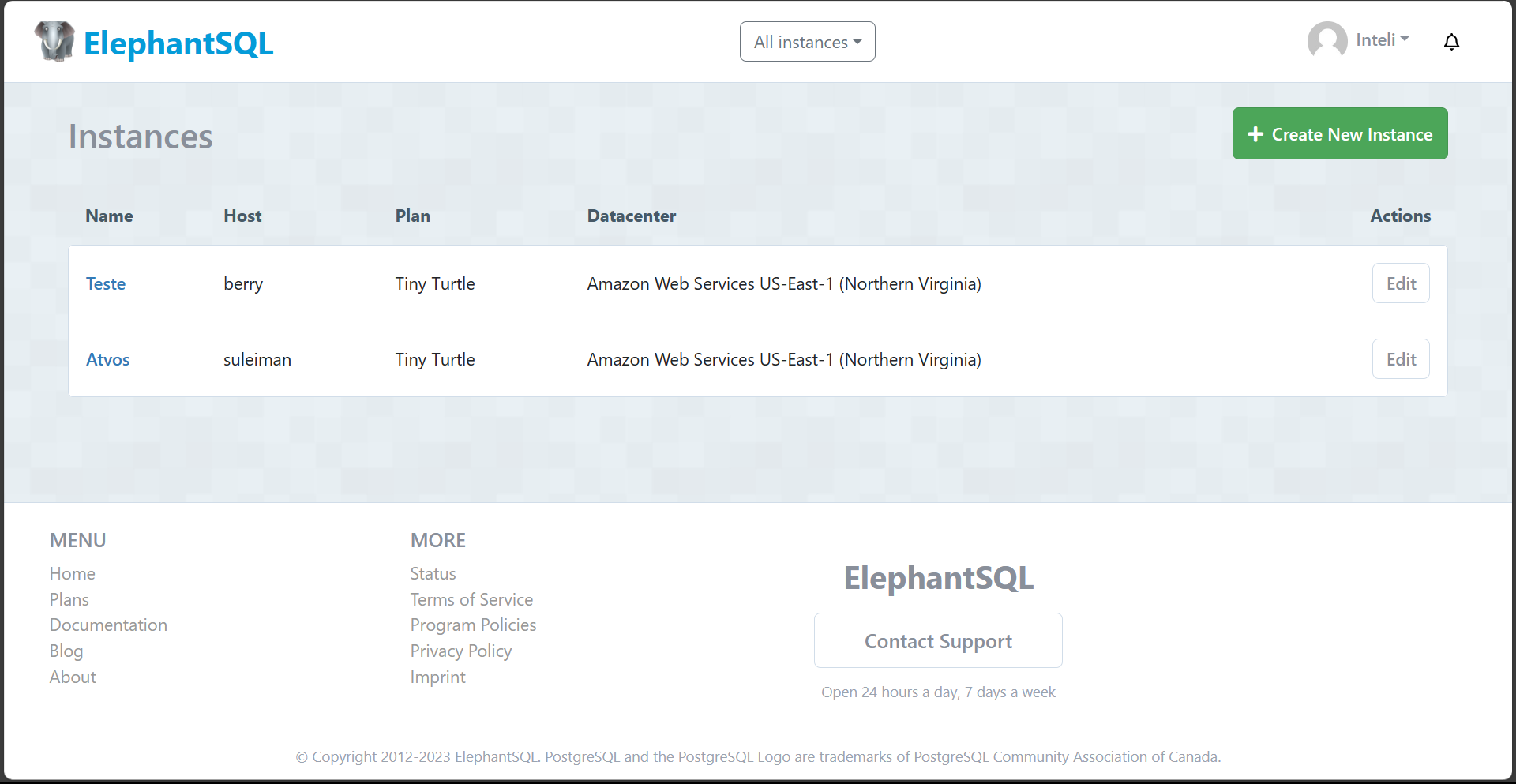
### 4.3.1 Criação de Conta no ElephantSQL

* Acesse o site ElephantSQL <https://www.elephantsql.com/>.
* Preencha o formulário de registro com suas informações, o qual está disponível no link [ElephantSQL](https://customer.elephantsql.com/signup). Após concluir, clique em "Sign Up".
* Você receberá um e-mail com as instruções para a criação da conta.

### 4.3.2 Criação de uma Instância no ElephantSQL

* Faça login na sua conta ElephantSQL.
* No painel de controle, clique em "**Create New Instance**".
* Escolha o plano desejado.
* Insira um nome para a sua instância e clique em "**Create** **Instance**".
* Aguarde até que a instância seja provisionada.

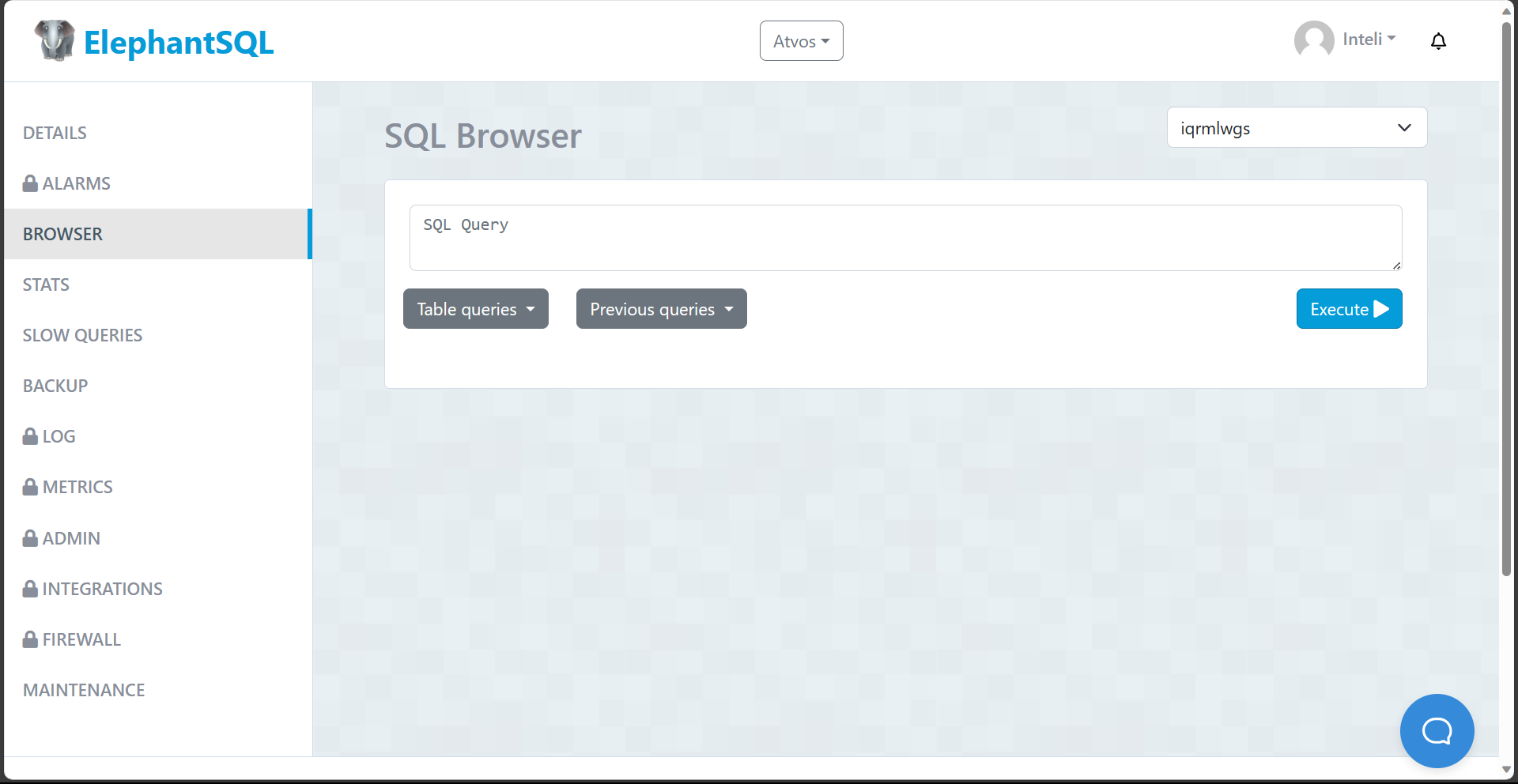
Figura 29 - Listagem de exemplo das instâncias criadas



### 4.6.3 Configuração do Banco de Dados PostgreSQL

Uma vez criada a instância, acesse a opção Browser, a qual abrirá uma página semelhante a esta (Figura 30):

Figura 30 - Página SQL Browser da instância criada no ElephantSQL



A seguir, será necessário criar as tabelas necessárias para o funcionamento do projeto. Dito isso, a tabela a seguir (Tabela 2) mostra quais entidades devem existir, bem como os seus respectivos atributos e tipos.

Tabela 2: Tabelas necessárias para o funcionamento do projeto

| Entidade | Nome da Coluna | TIpo | Pode ser nulo? |
| --- | --- | --- | --- |
| **almoxarifado** | id | integer | Não |
| **almoxarifado** | nome | character | Não |
| **almoxarifado** | created\_at | date | Não |
| **almoxarifado** | updated\_at | date | Não |
| **pecas** | rfid | text | Não |
| **pecas** | almoxarifado\_id | integer | Não |
| **pecas** | tipo | character varying | Não |
| **pecas** | disponivel | boolean | Não |
| **pecas** | created\_at | date | Não |
| **pecas** | updated\_at | date | Não |
| **pecas** | returned\_at | date | Sim |
| **oficinas** | id | integer | Não |
| **oficinas** | nome | character | Não |
| **oficinas** | created\_at | date | Não |
| **oficinas** | updated\_at | date | Não |
| **responsaveis** | rfid | character varying | Não |
| **responsaveis** | nome | character | Não |
| **responsaveis** | created\_at | date | Não |
| **responsaveis** | updated\_at | date | Não |
| **responsaveis** | perfil | character varying | Sim |
| **responsaveis** | status | boolean | Sim |
| **pecas\_oficinas** | id | integer | Não |
| **pecas\_oficinas** | peca\_rfid | character | Não |
| **pecas\_oficinas** | oficina\_id | integer | Não |
| **pecas\_oficinas** | created\_at | date | Não |
| **pecas\_oficinas** | updated\_at | date | Não |
| **oficinas\_responsaveis** | id | integer | Não |
| **oficinas\_responsaveis** | oficina\_id | integer | Não |
| **oficinas\_responsaveis** | responsavel\_rfid | character | Não |
| **oficinas\_responsaveis** | created\_at | date | Não |
| **oficinas\_responsaveis** | updated\_at | date | Não |
| **oficinas\_responsaveis** | deleted\_at | date | Não |
| **pecas\_responsaveis** | id | integer | Não |
| **pecas\_responsaveis** | peca\_rfid | character | Não |
| **pecas\_responsaveis** | responsavel\_rfid | character | Não |
| **pecas\_responsaveis** | created\_at | date | Não |
| **pecas\_responsaveis** | updated\_at | date | Não |

A título de exemplificação, na tela de SQL Browser, para criar a tabela **almoxarifado**, o seguinte comando SQL pode ser aplicado:

**CREATE TABLE almoxarifado (**

**id SERIAL PRIMARY KEY,**

**nome VARCHAR(255) NOT NULL,**

**created\_at DATE NOT NULL,**

**updated\_at DATE NOT NULL**

**);**

Portanto, ao seguir essas instruções, você terá configurado com sucesso o banco de dados PostgreSQL utilizando o serviço ElephantSQL, pronto para suportar a lógica de dados essencial para a aplicação.

# 5. Guia de Operação

Precedentemente, este guia demonstra o funcionamento de todas as principais partes do sistema IoT, incluindo os componentes físicos, a aplicação web, o banco de dados, o cliente MQTT e uma tabela do Google Sheets. Nesse sentido, o guia está dividido em ações/funcionalidades específicas do sistema como um todo.

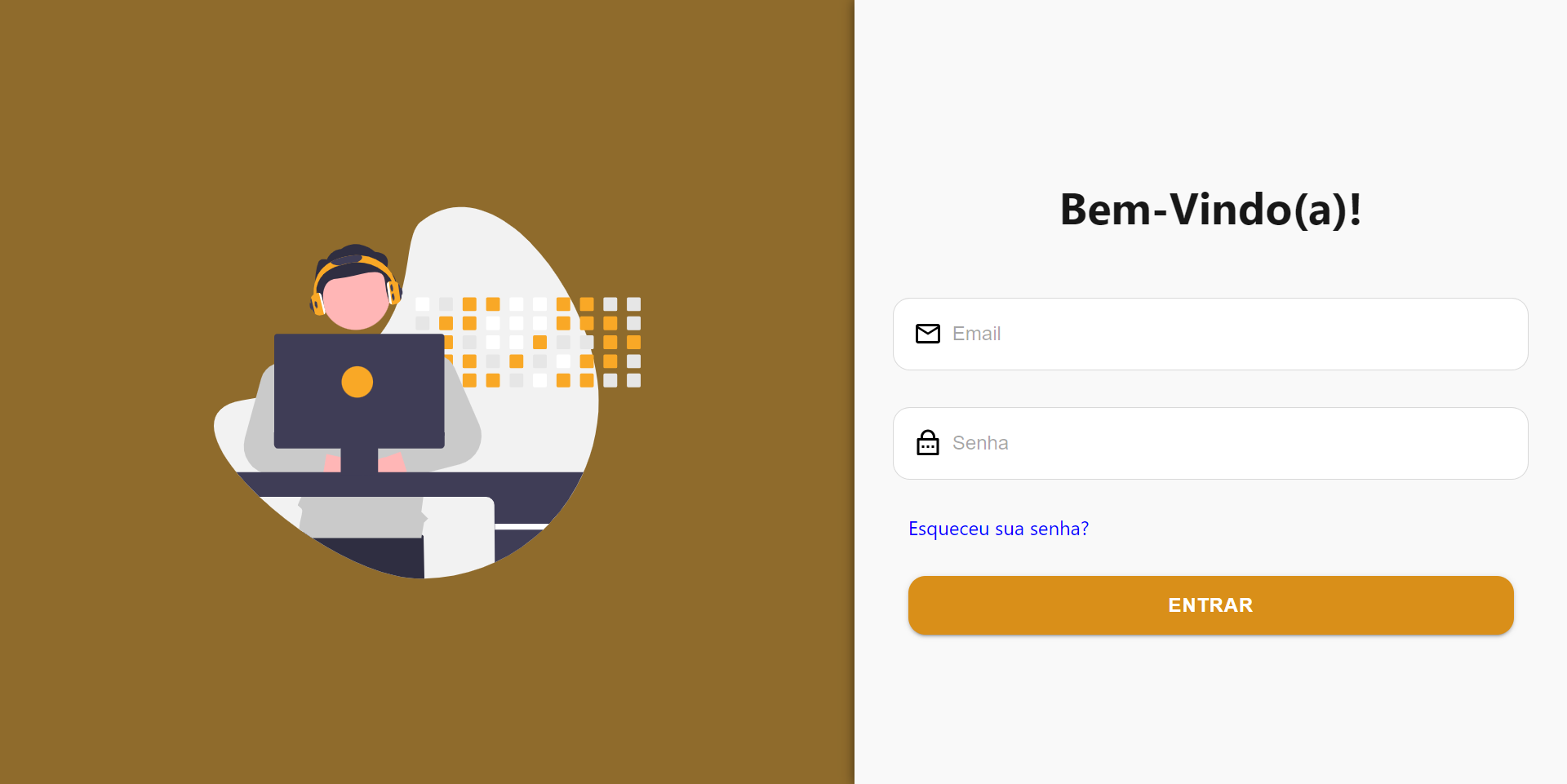
## 5.1 Cadastro de usuários

No que diz respeito ao cadastro de usuários, todos os componentes do sistema IoT serão utilizados. A seguir, há as etapas detalhadas para tal funcionalidade.

1. **Login na Aplicação Web**

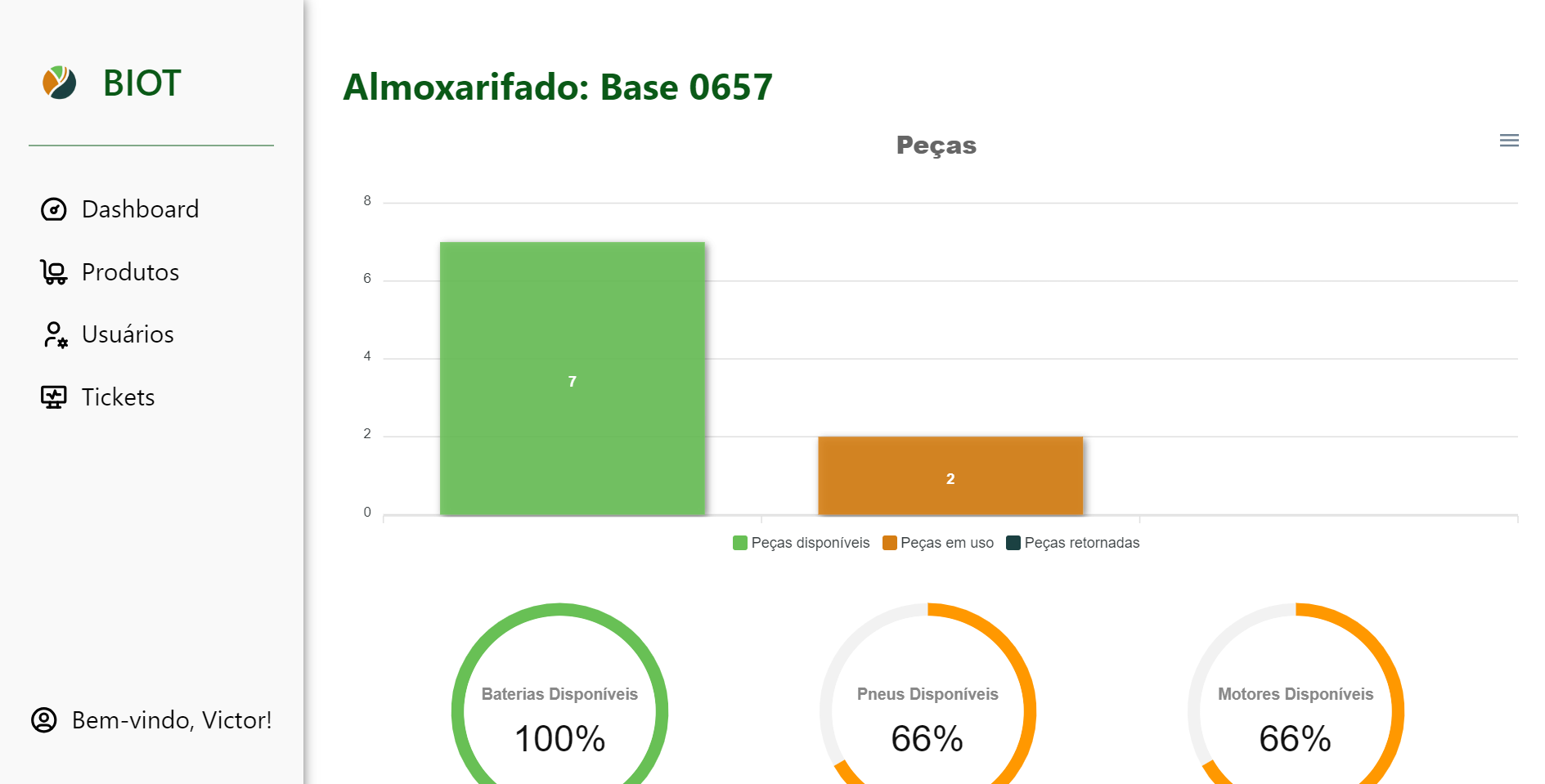
A priori, é necessário que o usuário (com perfil de líder de operação ou de almoxarife) realize o acesso à aplicação web. Para isso, a aplicação precisa estar rodando na máquina do usuário ou em algum servidor disponibilizado pela empresa. A imagem a seguir (Figura 31) demonstra a página de login do sistema web.

Figura 31 - Página de *Login* da Aplicação Web



Sendo assim, após preencher o e-mail e a senha previamente cadastrados, o usuário terá acesso ao *dashboard* da aplicação (como demonstrado na figura abaixo).

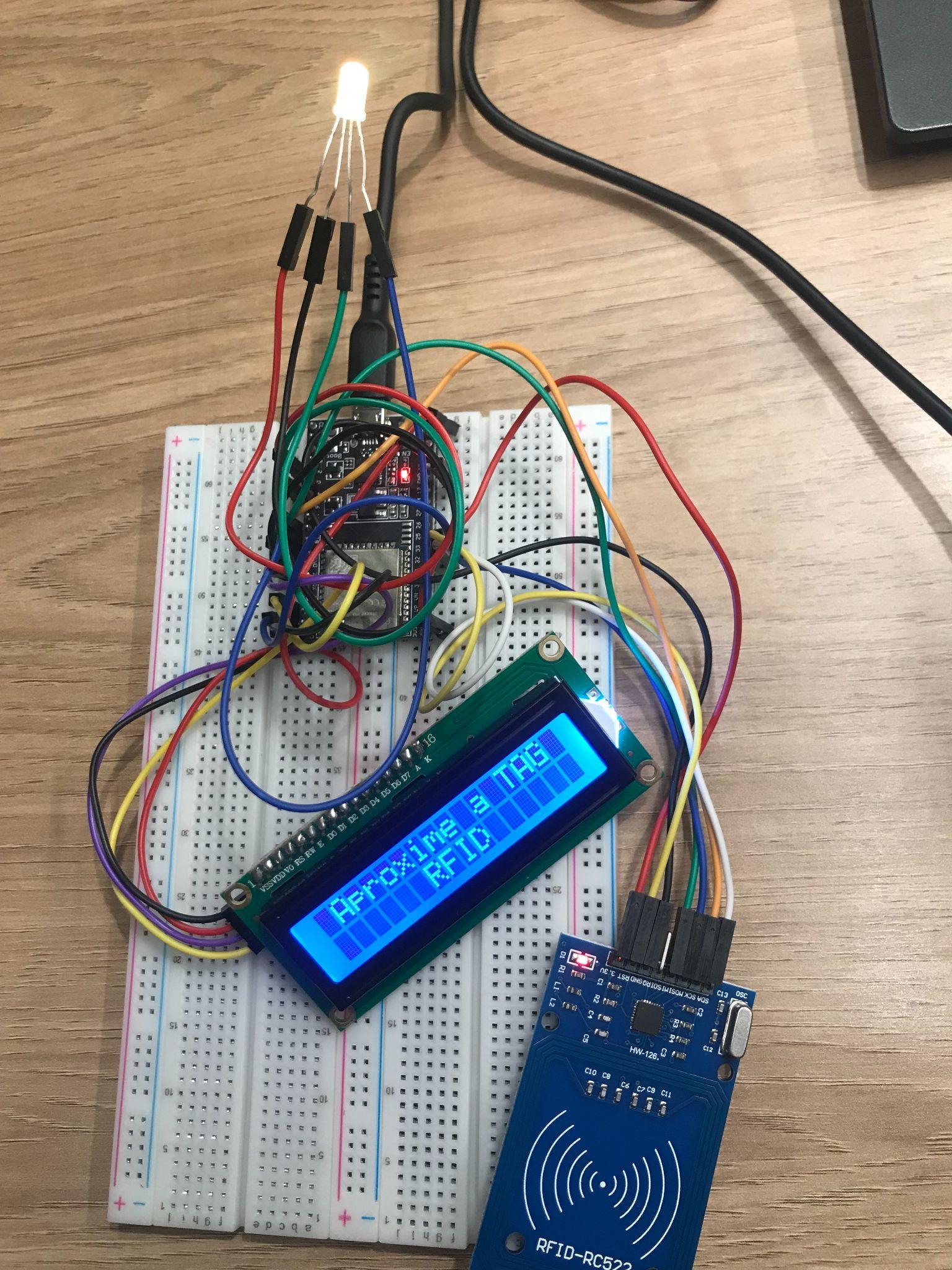
Figura 32 - Página principal da Aplicação Web



1. **Inicialização do identificador de RFID**

Para prosseguir, é necessário que o sistema que realiza leitura de RFIDs esteja em plena execução. Para isso, recomenda-se que esteja conectado a uma fonte de alimentação (pode ser o próprio computador via cabo USB). A imagem abaixo exemplifica o sistema em funcionamento.

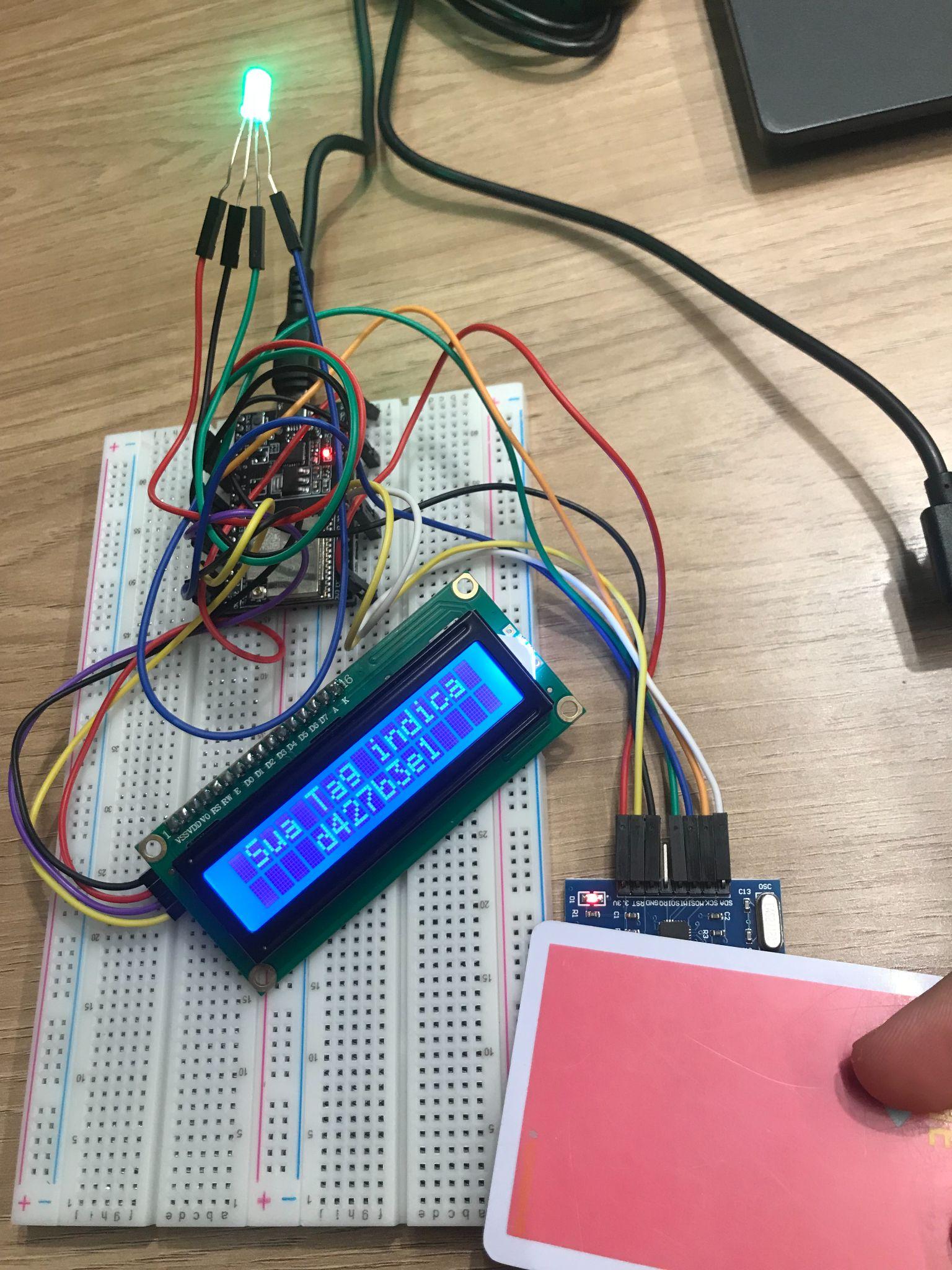
Figura 33 - Módulo IoT de identificação de RFID iniciado



1. **Leitura do RFID do cartão de identificação do usuário.**

Nesta etapa, o cartão de identificação do usuário deve ser aproximado à solução de leitura de RFID. Uma vez aproximado, o display LCD deste sistema informará qual o RFID (uma string que identifica unicamente aquela TAG) do cartão aproximado.

Figura 34 - Módulo IoT de identificação de RFID iniciado



1. **Acesso à página de cadastro de usuários.**

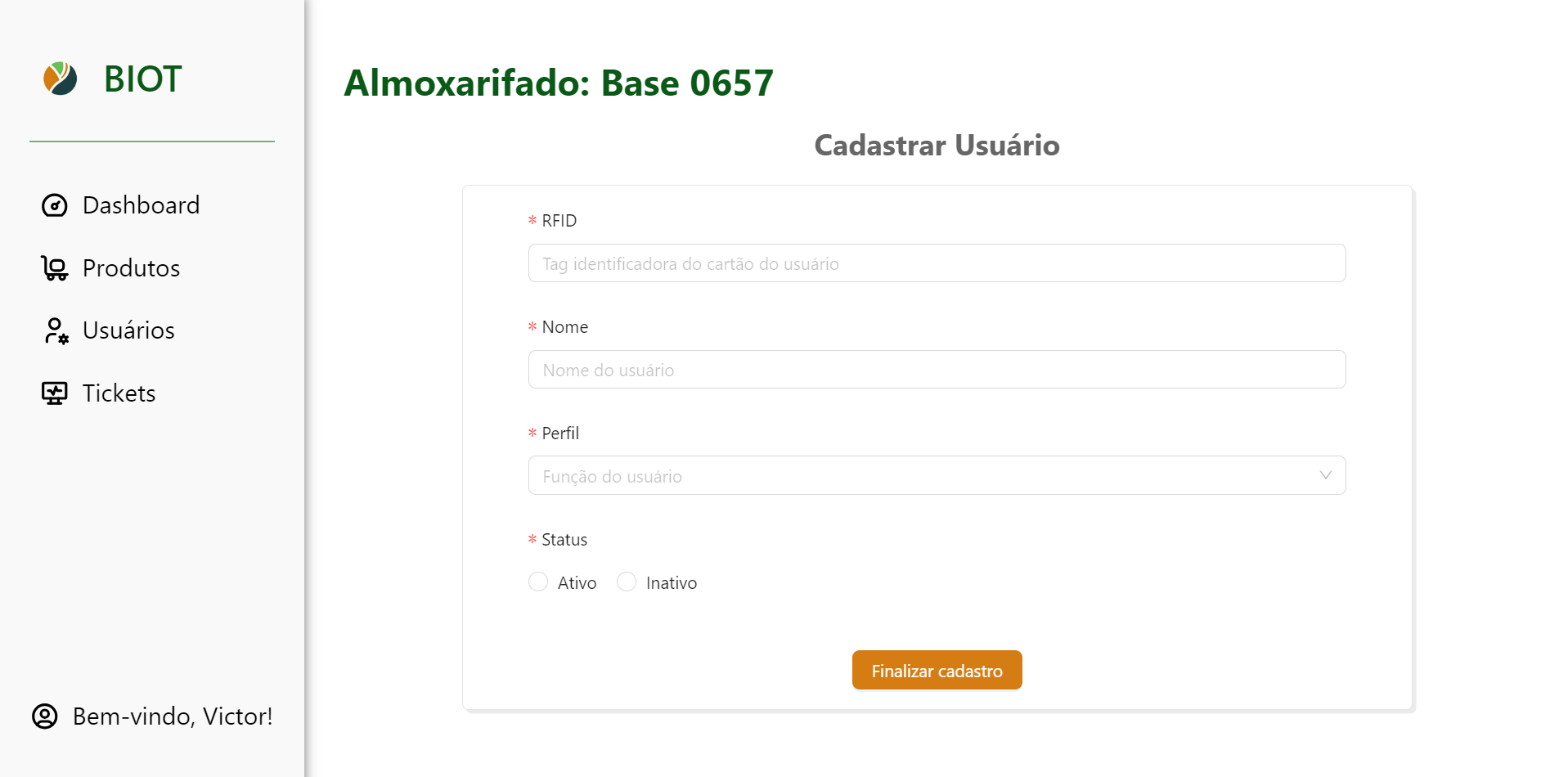
Com a dashboard aberta, é necessário acessar a página de cadastro de usuários. Para isso, na barra lateral, há um link chamado “Usuários”. Ao acessar este link, a seguinte tela será carregada:

Figura 35 - Página de listagem de usuários cadastrados



Sendo assim, basta clicar no botão de Adicionar (com o símbolo de “+”), que a tela de cadastro de usuários será aberta.

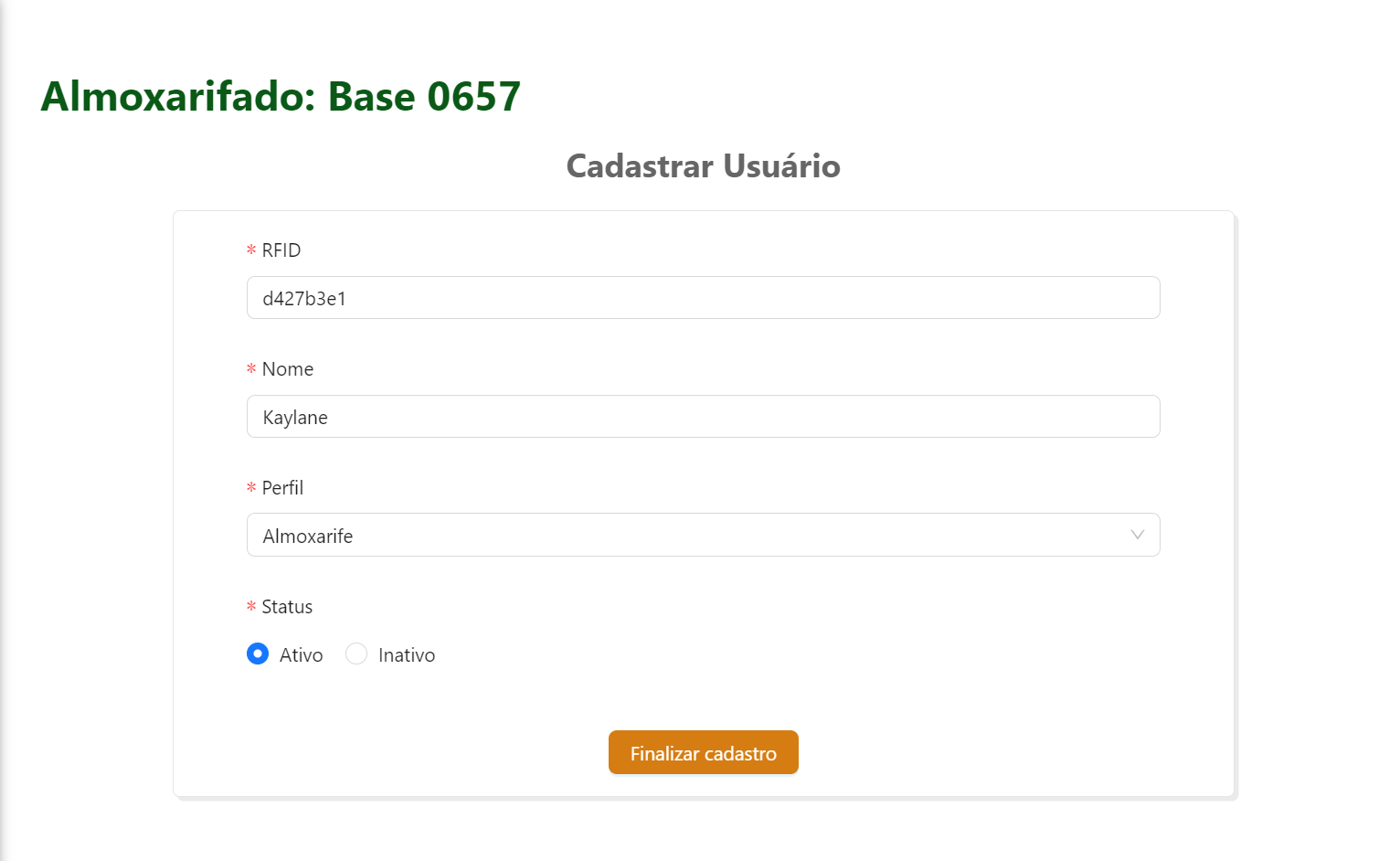
Figura 36 - Página de cadastro de usuário.



1. **Cadastro do usuário**

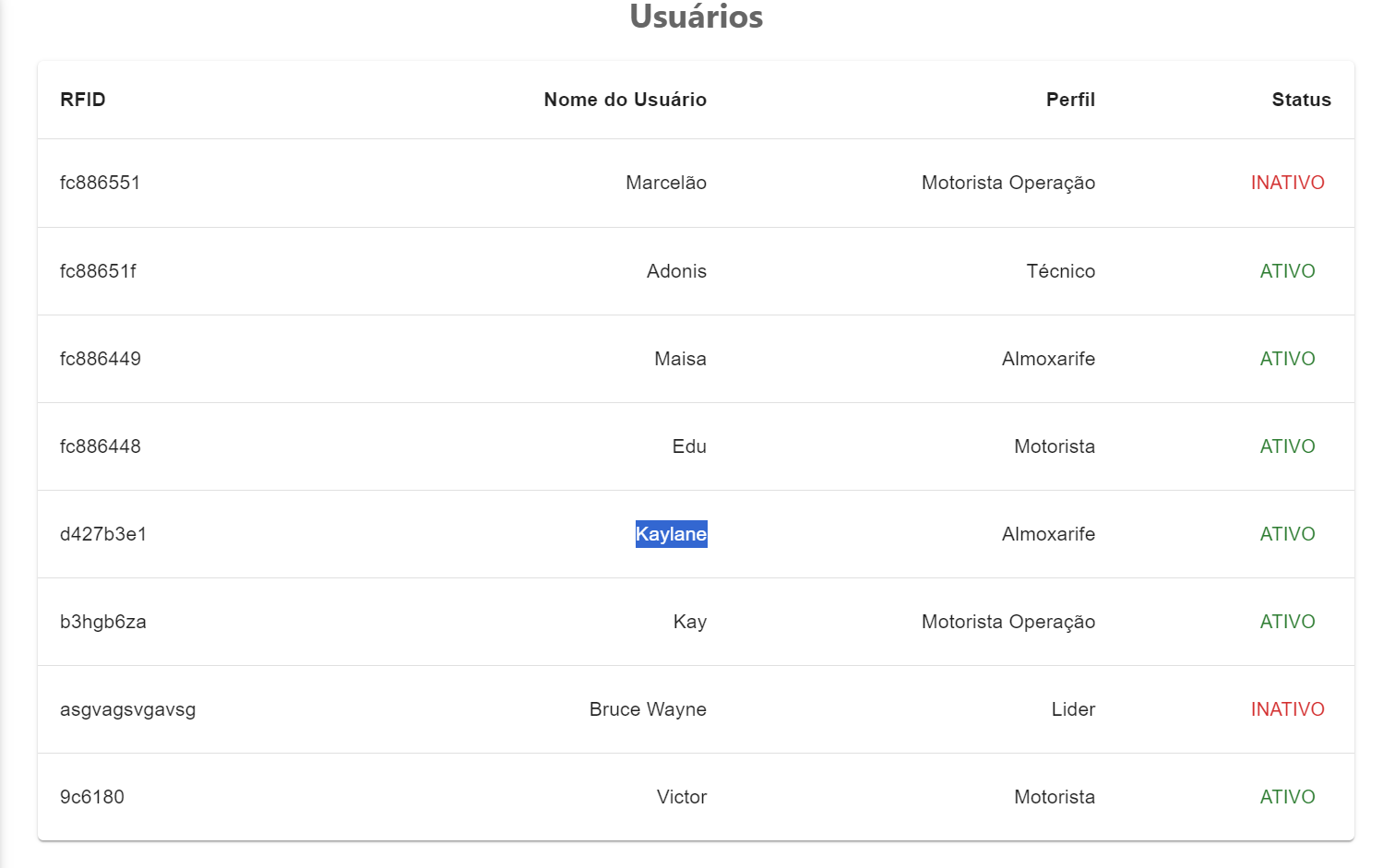
Na tela de cadastro, já com o RFID mostrado no display do identificador (passo 3), é possível cadastrar o usuário em questão. Para este exemplo, a almoxarife Kaylane será cadastrada.

Figura 37 - Preenchimento dos dados do usuário



Finalizando o cadastro, os dados do usuário estarão disponíveis na página de listagem:

Figura 38 - Dados do usuário cadastrado na página de usuários



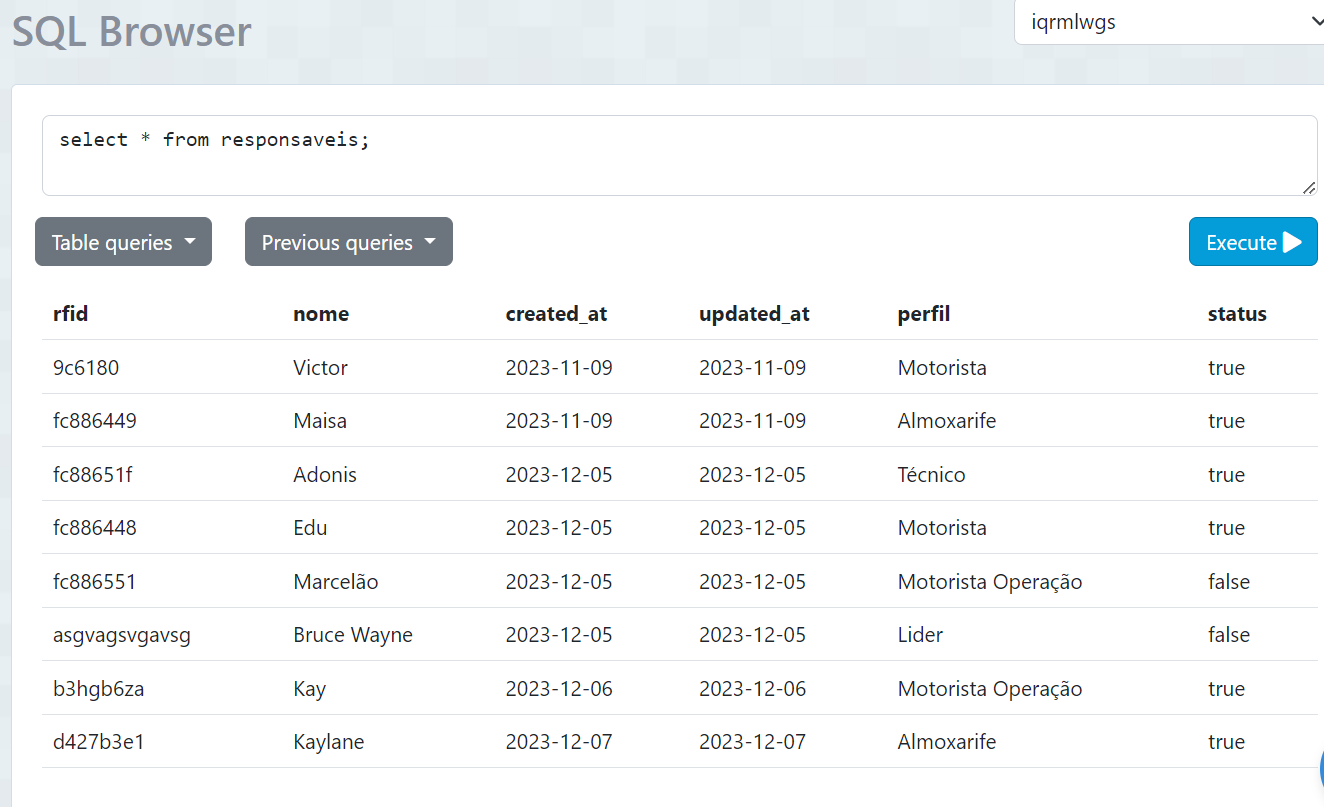
### 5.1.1 Processos internos

Conforme demonstrado nos passos acima, o cadastro do usuário envolve componentes físicos e dashboard. Mas, embora visualmente sejam apenas esses módulos que participam do processo, internamente há dois outros módulos funcionando.

O primeiro, back-end da aplicação, é responsável por receber a requisição e preparar os dados para o envio ao modelo. O modelo, por sua vez, envia a *query* de cadastro ao banco de dados.

O banco de dados, por sua vez, é o segundo módulo envolvido nesse processo. Ele é acionado pelo back-end (conforme citado no parágrafo anterior), de modo que é responsável por salvar, de fato, os usuários. A imagem a seguir expõe a listagem dos usuários do ponto de vista dos usuários.

Figura 39 - Dados do usuário cadastrado na página de usuários



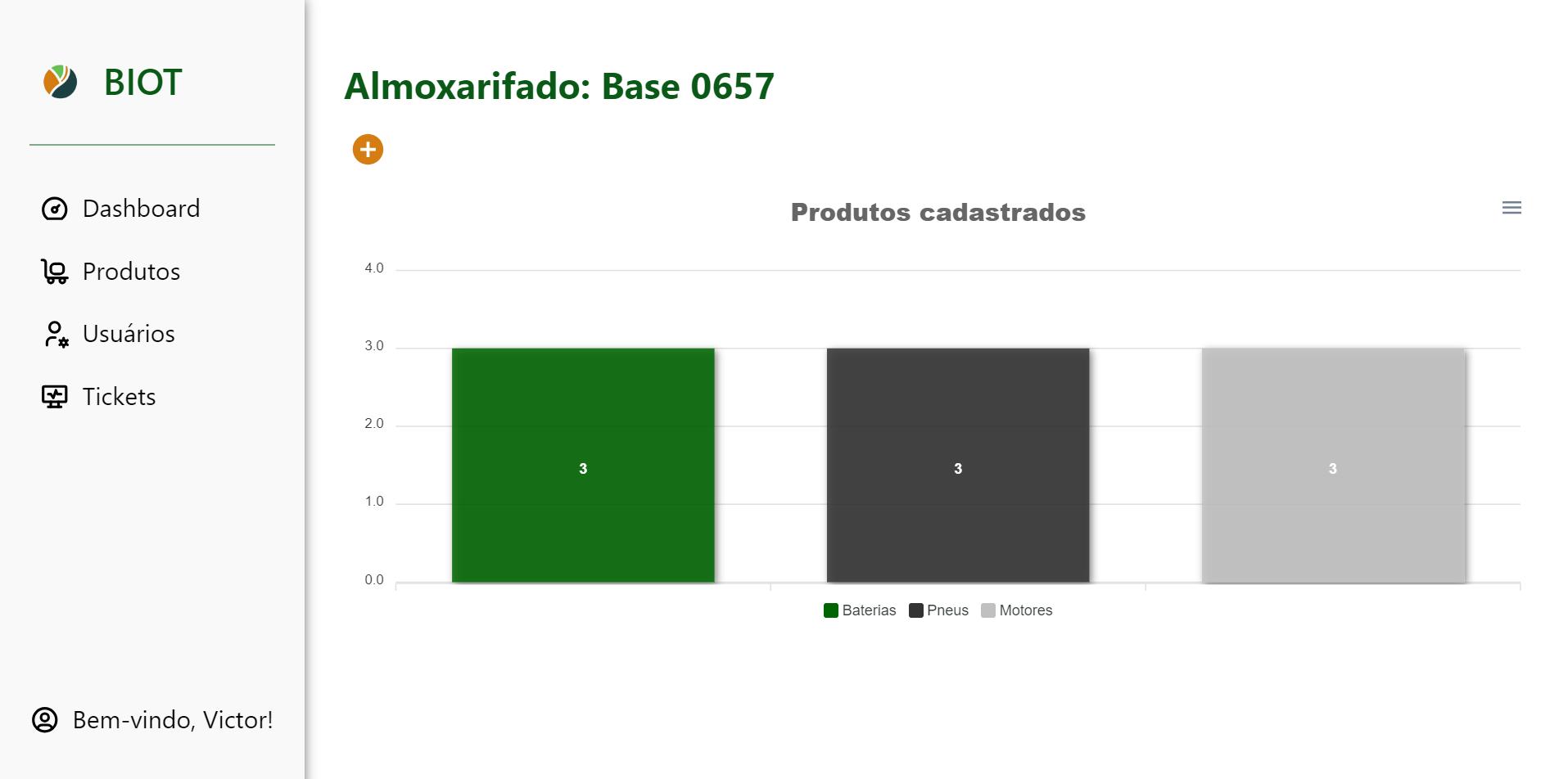
## 5.2 Cadastro de peças

Semelhante ao cadastro de usuários, é possível cadastrar as peças por meio da TAG de identificação, aproximando do módulo de identificação de RFID, e, na sequência, passando os dados para a tela de cadastro de produtos. Nesse sentido, como os passos de leitura da tag são iguais aos de cadastro de usuários, **os passos a seguir já pressupõem que a leitura do RFID foi realizada**.

1. **Acesso à página de cadastro de peças (produtos).**

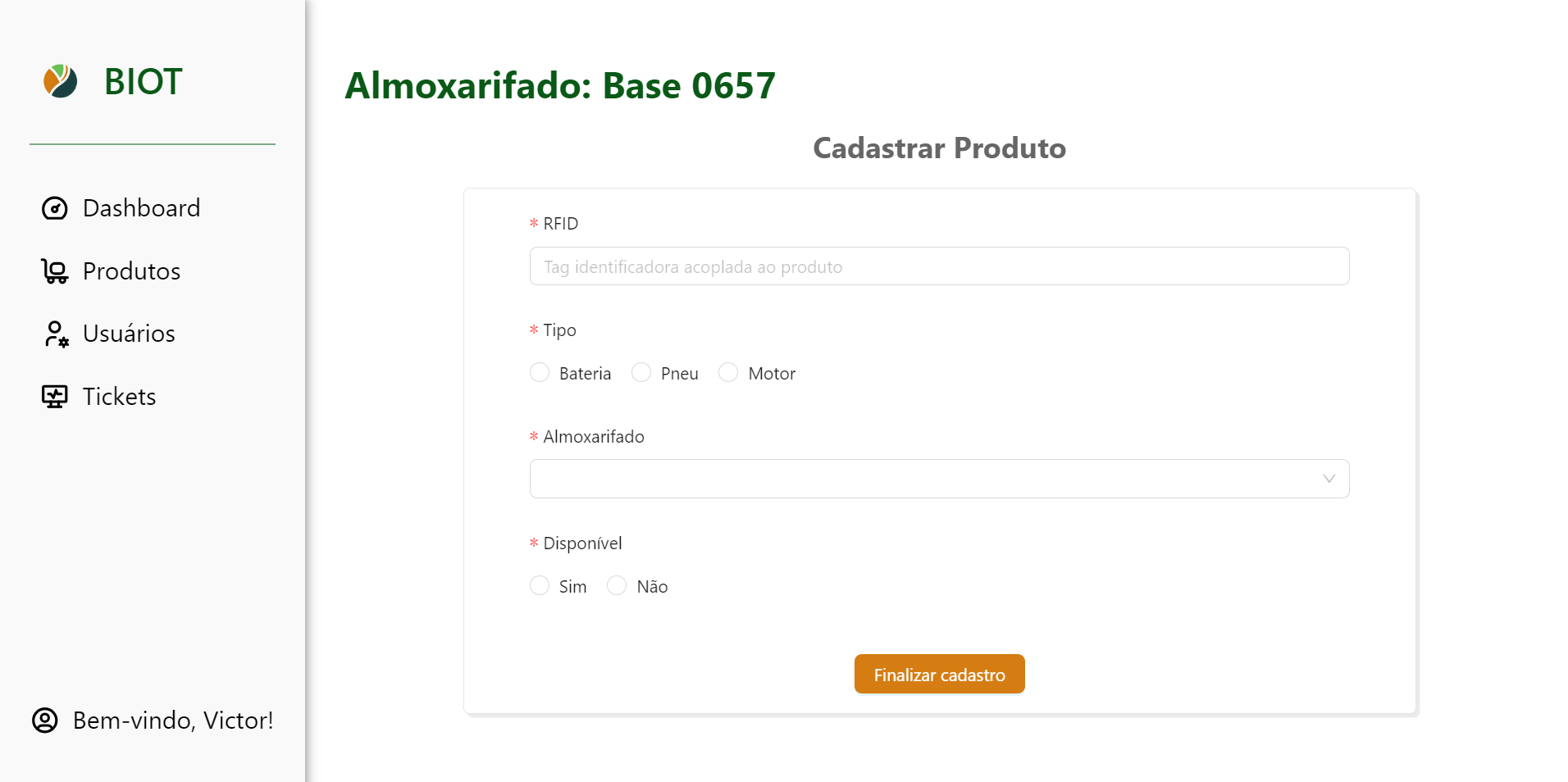
Com a dashboard aberta, é necessário acessar a página de cadastro de peças. Para isso, na barra lateral, há um link chamado “Produtos”. Ao acessar este link, a seguinte tela será carregada:

Figura 40 - Página de listagem de produtos cadastrados

****

Sendo assim, basta clicar no botão de Adicionar (com o símbolo de “+”), que a tela de cadastro de produtos será aberta.

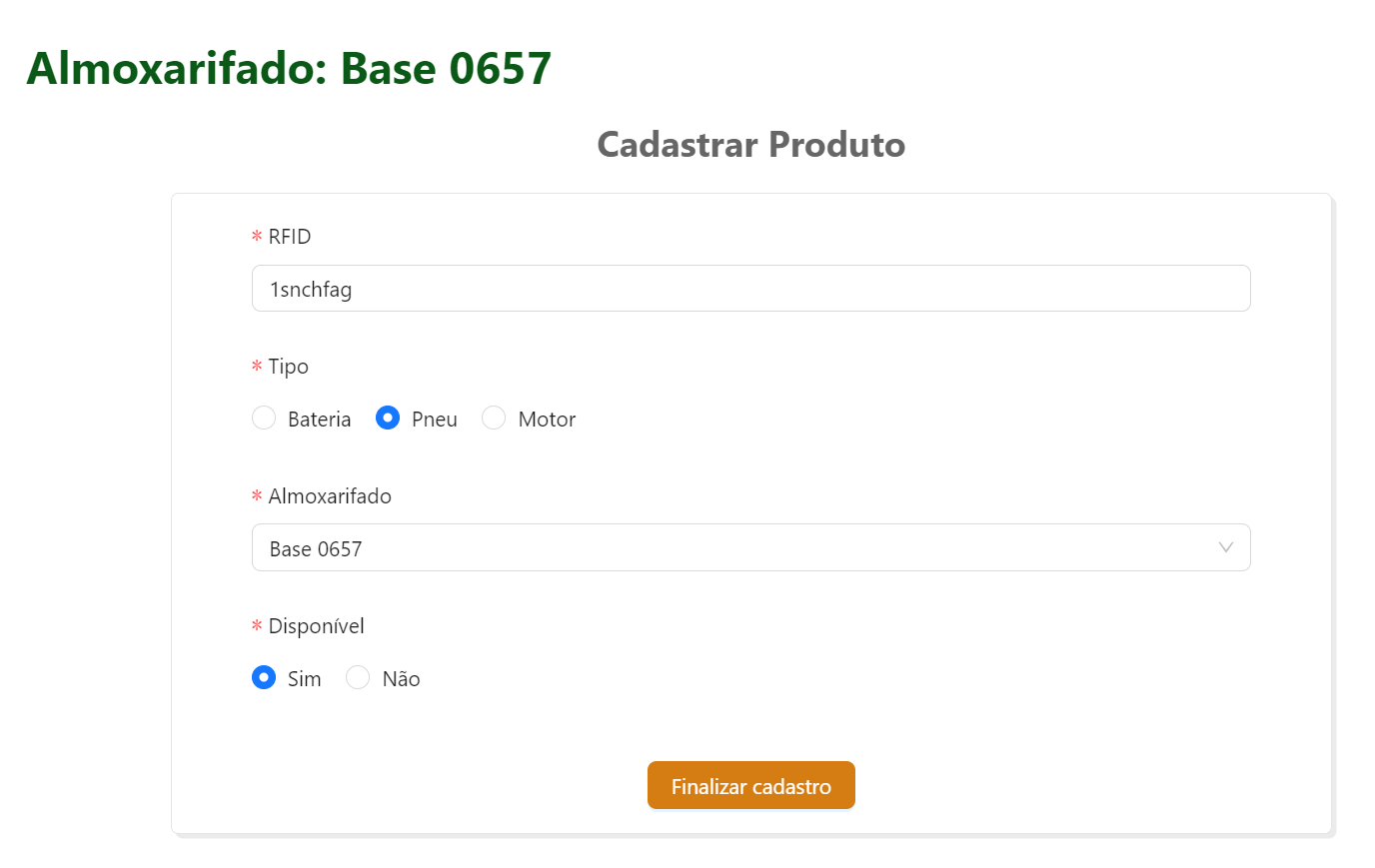
Figura 41 - Página de cadastro de produtos



1. **Cadastro de peças**

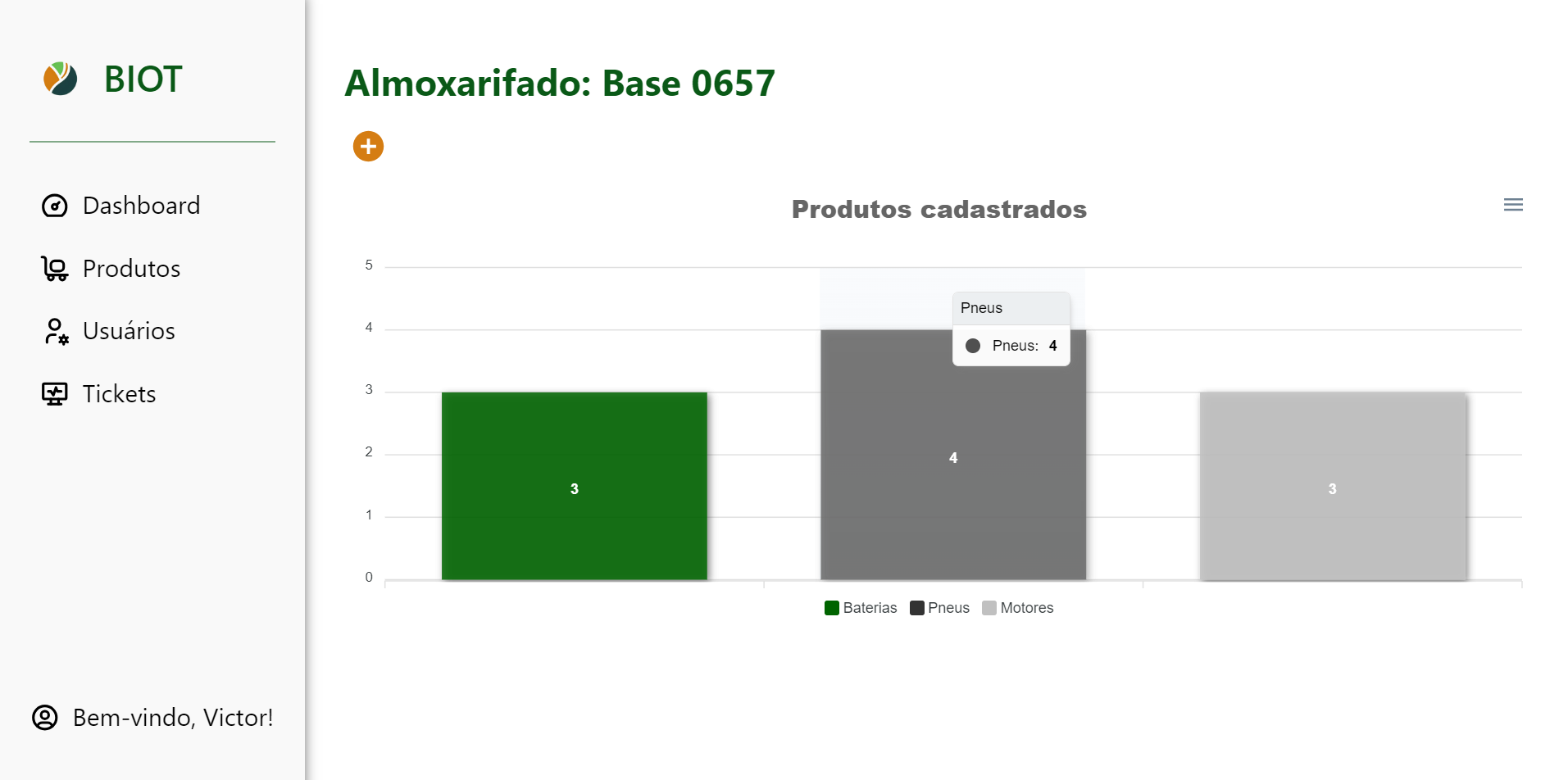
Na tela de cadastro, já com o RFID mostrado no display do identificador, é possível cadastrar a peça em questão. Para este exemplo, um pneu será cadastrado.

Figura 42 - Preenchimento dos dados da peça



Finalizando o cadastro, a peça será incrementada na página de listagem de produtos. Como no caso deste guia foi cadastrado um pneu, o gráfico adicionou um item na barra de pneus:

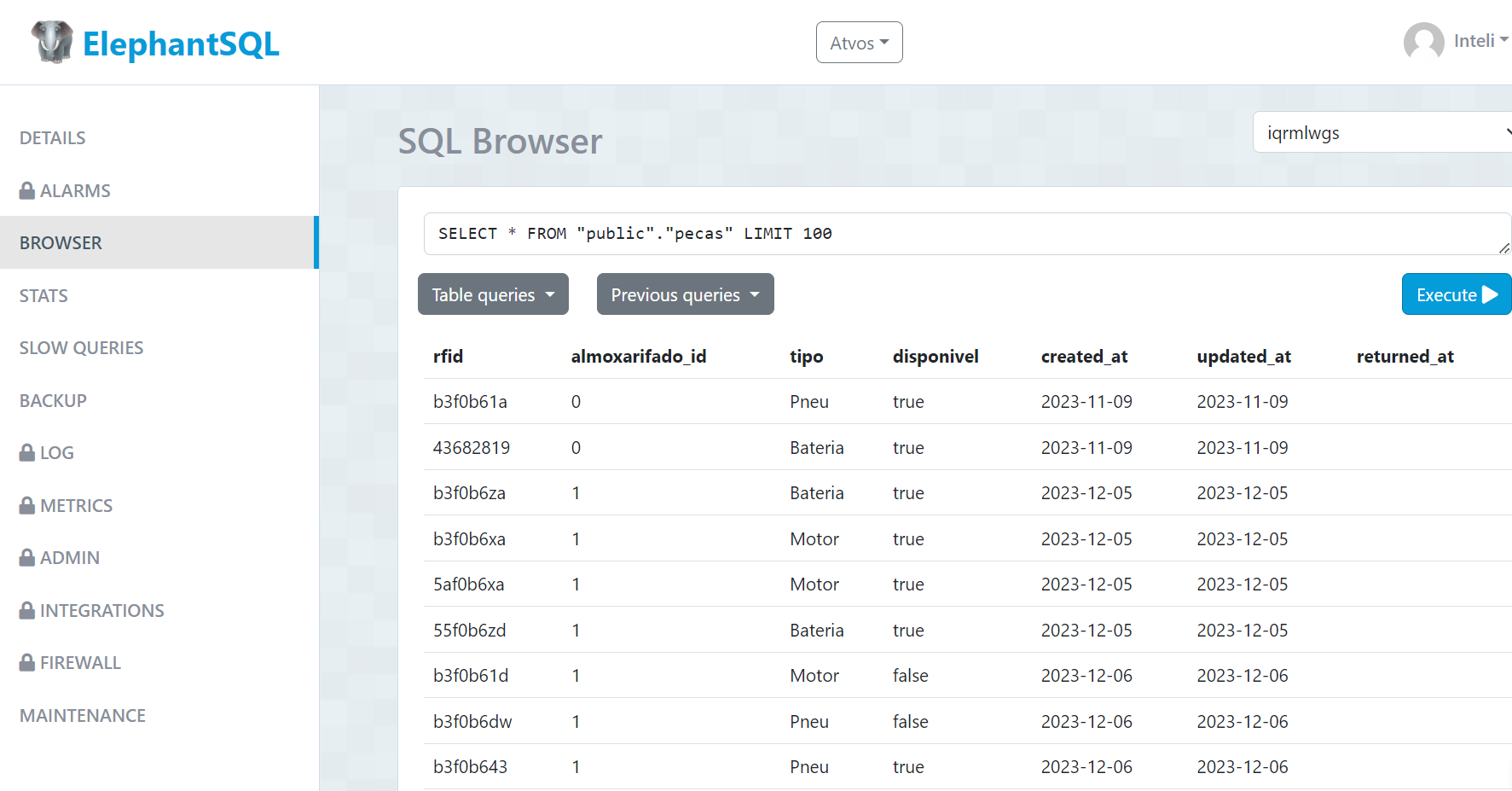
Figura 43 - Pneu cadastrado adicionado ao gráfico de produtos



### 5.2.1 Processos internos

Os processos internos são iguais aos do cadastro de usuário, mudando apenas a rota do back-end, os dados cadastrados e a tabela do banco de dados (neste caso, deixa de ser “responsaveis” e torna-se “pecas”). Nesse sentido, a título de esclarecimento, a imagem a seguir demonstra a listagem de peças cadastradas do ponto de vista do banco de dados, ou seja, mostra todas as informações dos produtos cadastrados na tabela citada.

Figura 44 - Listagem de peças através do banco de dados



## 5.3 Registro de Peças

O Registro de peças, também conhecido como troca de responsabilidade, é realizado através do módulo principal deste projeto (o qual possui a caixa protetora). Através dele, é possível aproximar a peça (bateria, motor ou pneu) do sensor e, em seguida, o usuário previamente identificado será o novo responsável pela peça, de modo a obter um histórico de todo o ciclo de vida da peça nas operações.

1. **Inicialização do módulo IoT**

Para a troca de responsabilidade, é crucial que o módulo IoT esteja em pleno funcionamento. Portanto, é necessário ligar o interruptor e verificar se o display e o LED são acionados corretamente.

1. **Identificação do usuário**

Com o sistema funcionando, o usuário (previamente cadastrado) precisa se identificar aproximando seu cartão de identificação do sensor RFID do módulo.

1. **Registro da peça**

Após a identificação, o usuário (almoxarife ou motorista do caminhão oficina) já pode aproximar a TAG RFID da peça no sensor. Ao fazer isso, caso a peça esteja cadastrada, será automaticamente vinculada sob sua responsabilidade. Caso haja um antigo responsável, este registro será deletado.

Como é um processo denso e com bastante interações visuais entre o módulo IoT e o usuário, foi gravado um vídeo que demonstra seu funcionamento. Tal vídeo pode ser acessado através do seguinte link: <https://www.youtube.com/shorts/Oem9aMJOQh8>.

### 5.3.1 Processos internos

Como o processo de troca de responsabilidade é mais relacionado à parte física da solução, outros módulos (para além da aplicação web) são envolvidos no processo. Sendo eles:

* Broker MQTT: Responsável pela comunicação online do módulo com outros serviços por meio do protocolo MQTT. Tal Broker está disponível na AWS Core IoT.
* Cliente MQTT: Responsável por escutar as transações (troca de mensagens) do broker com o módulo e, a partir daí, executar determinadas tarefas. No caso da troca de responsabilidade, o cliente é responsável por ler os dados dos sensores e identificar ou não o usuário e as peças, mediante consultas ao banco de dados, verificando a existência tanto do usuário quanto dos produtos.
* Banco de dados: Fornece, por meio de consultas, informações sobre a existência do registro de usuários, peças ou responsabilidades por meio de suas tabelas.

Sendo assim, o cadastro de usuários, o cadastro de peças e a troca de responsabilidades são as principais operações realizadas pela solução proposta pelo grupo BIOT Solutions em parceria com a empresa Atvos.

# 6. Troubleshooting

A priori, a tabela a seguir (Tabela 3) evidencia os principais pontos de falha no módulo IoT (toda a parte física da solução, a qual será disponibilizada em caminhões ou nos almoxarifados), bem como no *dashboard* (aplicação web). Dito isso, além da falha, a tabela elucida possíveis soluções para cada falha, destacando o que pode ser feito tanto do ponto de vista de conexão, quanto do ponto de vista de acionamento de equipes.

Tabela 3 - Troubleshooting

| **#** | **Problema** | **Possível solução** |
| --- | --- | --- |
| 1 | O Módulo IoT acoplado em um caminhão oficina não inicia devido à falta de alimentação energética. | Nesse caso, o ideal é manter, para cada módulo acoplado em um caminhão oficina, um pacote com, no mínimo, 10 baterias. Dito isso, quando uma bateria descarregada, a troca por uma nova é a opção mais viável, bastando levantar a caixa protetora da solução, desconectar a bateria antiga e acoplar a nova. |
| 2 | LED RGB ou Display LCD não funcionam como esperado. No caso do LED, é crucial que as cores azul, amarelo, verde e vermelho sejam demonstradas corretamente. Já no caso do Display, mensagens informativas devem ser transmitidas ao motorista e ao almoxarife (a depender da localização do módulo). | Nesse caso, é provável haver um problema na conexão interna à caixa protetora. Sendo assim, recomenda-se que um técnico seja chamado ao local, de modo que este retire a fonte de alimentação do protótipo e, em seguida, avalie se as conexões cabeadas entre os componentes estão com alguma interferência. Caso seja identificado um problema nessas conexões, o técnico pode efetuar a ligação dos componentes ou, a depender da gravidade, substituir o módulo por um novo e remontar o antigo. |
| 3 | O módulo não consegue se conectar ao wi-fi, apresentando constantemente a mensagem “aguardando conexão”. | Recomenda-se a verificação, no Código-fonte do projeto, de modo a validar se as credenciais de acesso à rede estão adequadas ao esperado.  Além disso, é recomendado que o técnico reinicie o próprio ESP32 (microcontrolador responsável por todas as operações do módulo físico) e tente a conexão. |
| 4 | A TAG RFID acoplada em alguma peça (bateria, motor ou pneu) não é lida pelo sensor RFID corretamente. | Recomenda-se, para este caso, que a TAG seja aproximada de formas diferentes, haja vista que o sensor pode apresentar falhas a depender da proximidade da TAG com o sensor, ou a depender da forma como a TAG é aproximada. Ademais, devido às especificações técnicas do sensor utilizado nesta solução, recomenda-se que a peça seja aproximada o mais perto possível do sensor. |
| 5 | O cartão de identificação do motorista responsável pelo caminhão oficina ou do almoxarife não é identificado pelo módulo IoT, de modo que a mensagem “Credencial não registrada” é informada pelo display LCD. | Recomenda-se que o líder do almoxarifado ou da frente de operação, cujo nível de acesso à dashboard é mais elevado, liberando as páginas de configuração, seja acionado. O líder, por sua vez, deve verificar no sistema se o usuário em questão está cadastrado e, em caso afirmativo, deve verificar se está ativo. Além disso, caso não esteja cadastrado, o líder tem a opção de realizar o cadastro do usuário. |
| 6 | A peça não é encontrada pelo sistema ao ser aproximada do sensor do módulo IoT. | Semelhante ao problema de número 5, o ideal a ser feito neste caso é o acionamento do líder da operação ou do almoxarifado, de modo que o cadastro de tal peça seja validado na dashboard e, caso necessário, seja realizado. Outrossim, a peça pode estar indisponível e, para isso, recomenda-se que o status seja alterado. |
| 7 | O registro de responsabilidade das peças (quando uma peça é aproximada do módulo acoplado em algum caminhão oficina) não é efetuado, de fato, no banco de dados. | Nesse caso, é importante avaliar se o banco de dados está em execução, conforme esperado, bem como é necessário avaliar se o *Broker* MQTT (disponibilizado na AWS Core IoT) está em pleno funcionamento, de modo a viabilizar a comunicação via protocolo MQTT entre o protótipo físico, o cliente MQTT e a Dashboard do usuário. |
| 8 | Peças com falha ou com danos físicos | É recomendado o acionamento dos responsáveis técnicos imediatamente após a percepção de alguma falha ou algum dano físico no módulo IoT. |
| 9 | Falha no cadastro de usuários ou no cadastro de peças | Para esses casos, recomenda-se a consulta ao banco de dados, por parte da equipe técnica do projeto, visando constatar se o cadastro foi realmente bem-sucedido. Em casos afirmativos, não há preocupação, sendo válida a recomendação de reiniciar o dispositivo físico e atualizar a aplicação web.  Caso o cadastro não tenha sido concluído, é necessário tentar um recadastro e, em situações nas quais o erro persiste, é crucial verificar a conexão com a internet, o funcionamento do cliente MQTT, a disponibilidade do broker e o funcionamento adequado do sensor RFID. |

Nesse viés, conforme a tabela acima, foram elencados 7 principais problemas que podem ocorrer na solução apresentada neste manual. Por conseguinte, fica evidente a necessidade de uma equipe técnica nas frentes de operações e nos almoxarifados, visando a solução de problemas específicos que pressupõem conhecimento aprofundado na solução.

Destarte, os erros que envolvem conexão/comunicação entre as partes do sistema carecem de um olhar mais detalhado, haja vista que podem indicar problemas genéricos (como a conexão com a internet) ou problemas específicos de alguma parte da arquitetura (broker MQTT, cliente MQTT, back-end, front-end, entre outros).

Portanto, vale salientar que os erros supramencionados sejam constantemente atualizados, sobretudo no que tange às propostas de soluções, possibilitando a identificação de novos riscos e as suas respectivas tratativas.