# Desenvolvimento de um Sistema de Monitorização e Controlo para Casas Inteligentes

Barbara Fonseca
*Universidade do Minho*
*Escola de Engenharia*
Guimarães, Portugal
a97778@alunos.uminho.pt

Camila Pinto
*Universidade do Minho*
*Escola de Engenharia*
Guimarães, Portugal
a91687@alunos.uminho.pt

Eduarda Dinis
*Universidade do Minho*
*Escola de Engenharia*
Guimarães, Portugal
a95573@alunos.uminho.pt

Pedro Martins
*Universidade do Minho*
*Escola de Engenharia*
Guimarães, Portugal
a96748@alunos.uminho.pt

***Abstract*—Nos dias de hoje, o avanço da tecnologia tem impulsionado o desenvolvimento tecnológico em várias indústrias, sendo a domótica uma das áreas que tem explorado essas oportunidades para proporcionar maior conforto e segurança aos habitantes. A integração dos sistemas IoT está a transformar o conceito de *smart home* numa realidade do presente, permitindo o controlo remoto de eletrodomésticos, câmaras de vigilância e iluminação bem como receber em tempo real notificações sobre esses elementos. Estes avanços têm adaptado as habitações para se tornarem mais eficientes, seguras e autónomas, e de igual forma contribuir para a comodidade, conforto e redução de despesas dos seus habitantes. Neste cenário, foi desenvolvida uma smart home, com o objetivo de entender cada componente da arquitetura de um sistema IoT e obter um produto final capaz de controlar dispositivos de medição, e receber dados em tempo real sobre eles. Para comprovar o funcionamento do nosso sistema, recorremos à ferramenta de software *Visual Studio Code* para testar o código, e à ferramenta *MYSQL* para elaborar a base de dados e testar a sua conexão com o sistema desenvolvido.**

***Palavras-Passe— domótica, IoT, smart home.***

## I. Introdução

Atualmente, com o aparecimento de tecnologia cada vez mais recente e inovadora, um sistema *IoT (Internet of Things)* [1]numa *smart home* já é um conceito do presente com o propósito de transformar uma casa numa habitação mais eficiente, inteligente e autónoma, o que permite proporcionar uma comodidade e conforto maior aos moradores, bem como a redução de despesas mensais em muitos dos casos[2].

A Internet das Coisas (IoT) é a rede de coisas físicas e aparelhos virtuais que comunicam e interagem entre si. O termo "coisas" refere-se a uma ampla gama de dispositivos como veículos, dispositivos vestíveis, sensores físicos, virtuais entre outros[3]. A principal característica deste sistema é a capacidade de conectar objetos do mundo físico ao mundo digital.

A aplicabilidade deste sistema no contexto de uma *smart home* assenta na comunicação com múltiplos dispositivos sensores e atuadores que funcionam como interface com o mundo físico real, permitindo a recolha de informação. Estas informações podem ser processadas e utilizadas para controlar dispositivos atuadores, como sistemas de iluminação, segurança e aquecimento. O sistema *IoT* permite que este controlo seja realizado remotamente através de dispositivos móveis, o que permite aos utilizadores a gerir e monitorizar a sua casa à distância, o que contribui para a eficiência energética e aumenta o conforto dos seus utilizadores [4].

No contexto deste trabalho, o estudo da aplicabilidade dos sistemas IoT, traduz-se no desenvolvimento de um sistema sensor, constituído pelos sistemas de controlo de temperatura e humidade, através de sensores de temperatura e humidade o sistema de controlo luminoso, através de sensores de luminosidade e o sistema de controlo de presença humana com a utilização de sensores de movimento. Os dados recolhidos por estes sensores, são armazenados numa base de dados e transmitidos para um servidor *web* o utilizador pode interagir com o sistema e consultar os dados.

Este artigo é composto por secções que estão organizadas da seguinte forma: na secção II são expostos alguns trabalhos relacionados. Na secção III são apresentados os protocolos de comunicação utilizados e a arquitetura do sistema. Na secção IV é descrita a implementação do sistema desenvolvido, ao passo que na secção V os testes e resultados obtidos. Eventualmente, na secção VI são apresentadas as principais conclusões. Por fim, encontram-se os Anexos, que estão divididos em Anexo A, B,C,D ,e E.

## II. Trabalho Relacionado

### A. *Desenvolvimento de um Sistema de Controlo e Monitorização Residencial Através de Tecnologias Associadas à Internet das Coisas*

Nesse projeto foi implementado um sistema de controlo e monitorização residencial baseado numa Plataforma IoT[5]. Desta forma, o objetivo passou por desenvolver um sistema de baixo custo, acessível a qualquer pessoa, com capacidade para aumentar a eficiência energética, e integrar um sistema de um carregamento de um veículo.

Este sistema utiliza os microcontroladores com módulo Wi-Fi (NodeMCU ESP8266 ESP-12E) e com módulo BLE (FireBeetle ESP32). Como sensores/atuadores utiliza o sensor DHT11, sensor de tensão ZMPT101B, sensor de corrente Gravity 20A, módulo relé, entre outros.

Para comunicação foi escolhido o protocolo MQTT (Message Queuing Telemetry Transport) através do modelo publish/subscribe. Para o armazenamento de dados foi implementada uma base de dados no hub/gateway. Posteriormente, foi também implementado um servidor

HTTP e uma Web B. Neste projeto foram utilizadas linguagens de programação como: Java; PHP; Python; C/C++.

### B. *Automação Residencial Inteligente Integrada Baseada na Internet das Coisas*

Neste artigo é desenvolvido um sistema multifuncional, de baixo custo e flexível para monitorizar e controlar as casas inteligentes[6].

Para implementar este sistema, é utilizado como placa de desenvolvimento o node-MCU ESP32 com conectividade com a Internet e BLE (*Bluetooth Low Energy*). Para demonstrar as diferentes funcionalidades do sistema, utilizam sensores como, temperatura e humidade (DHT-22), luminosidade (LDR), sensores de movimento (PIR), e o sensor MQ-6.

O sensor PIR é responsável pela deteção de movimento, ou seja, pela deteção de presença humana. Caso seja detetado movimento é devolvido 1, senão é devolvido 0. O sensor DHT-22 tem a capacidade de medir a temperatura e humidade do ar, e com este é possível recorrer a bibliotecas de Arduino, a temperatura pode ser medida entre -40ºC a 100ºC e a humidade entre 0% a 100%[6].

Os dados recolhidos pelos sensores são, posteriormente, carregados para Firebase via ESP-32, e é desenvolvida também uma aplicação com auxílio do IDE Android Studio.

A aplicação fornece funcionalidades como alerta de incêndio, estado de temperatura e humidade, monitorização e controle de luzes domésticas, ventiladores, fechadura de porta[5]. Uma funcionalidade interessante é que, quando a tensão do sensor de luminosidade e a saída do sensor de movimento estão acima do limite, o controlador acende as luzes.

## III. Conceção do Sistema

### A. *Arquitetura*

Para a correta elaboração do sistema de monitorização, é essencial a existência de uma arquitetura concisa, organizada e confiável. Neste sentido, para elaborar o sistema de monitorização, juntamente com uma interface gráfica iterativa para testar funcionalidades e monitorizar os sistemas sensores, dividimos o sistema em várias camadas, como ilustra a Figura 1.

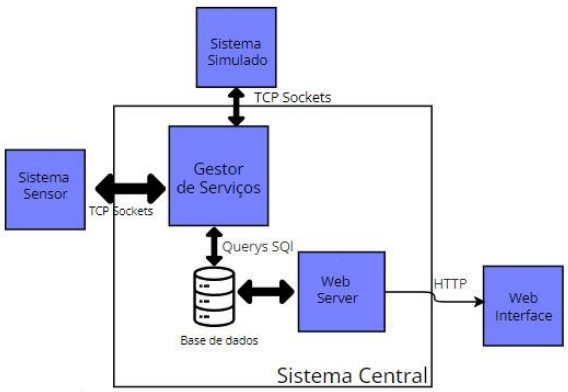


Figura 1 - Arquiterura implementada.

#### 1) *Sistema Sensor*

O sistema sensor foi desenvolvido utilizando uma placa de desenvolvimento ESP32-DEVKITC-32E e é constituído por 3 sensores: PIR (*Passive Infrared*), utilizado para deteção de movimento[7]; LDR (*Light Dependent Resistor*) utilizado para avaliar a intensidade luminosa[8]; e o DHT11 (*Digital Temperature and Hummidity*) utilizado para monitorizar temperatura e humidade[5].

A conexão entre o sistema sensor e o ESP32-DEVKITC-32E é estabelecida através da ligação entre os pinos do sensor e os pinos do microcontrolador. Desta forma, os dados recolhidos pelos sensores devem ser processados e convertidos, sendo posteriormente enviados periodicamente para o servidor. Devido à capacidade de conectividade Wi-Fi fornecida pela placa ESP32, a comunicação entre esta e o servidor é estabelecida através de Wi-Fi.

O sistema sensor estabelece uma ligação ao gestor de serviços através de *sockets* TCP, sendo responsável por enviar os dados das suas medições para o gestor de serviços, quando recebe a trama *start* e de acordo com o período de amostragem definido. Além disso, o sistema sensor é capaz de interromper o envio dos dados ao receber a trama stop.

#### 2) *Sistema Simulado*

O sistema simulado desempenha a mesma função que o sistema sensor, no entanto, em vez de medir dados reais, gera dados aleatórios com o propósito de testar se diversos sistemas locais estão operacionais simultaneamente.

Para estabelecer uma ligação com o gestor de serviços, o sistema simulado utiliza *sockets* TCP, através dos quais envia e interrompe os dados obtidos, dependendo da trama recebida.

#### 3) *Gestor de Serviços*

O gestor de serviços desempenha um papel fundamental nas ligações com o sistema sensor, o sistema simulado e o web server. As conexões entre o gestor de serviços e estes sistemas são feitas através de um *socket* TCP, indicando o *host* e a porta, sendo que a cada sistema foi atribuída uma porta diferente e o *host* está disponível para receber conexões de qualquer endereço e de mais do que um sistema simulado.

O servidor *web* envia as tramas start e stop ao gestor de serviços, que lida com estas requisições com auxílio de *threads* e envia o respetivo comando para o sistema simulado e o sistema sensor.

Além disto, o gestor de serviços desempenha um papel fundamental na iteração com a base de dados, estabelecendo uma ligação servidor-cliente por *sockets* TCP. Ao receber os dados dos sistemas, o gestor de serviço armazena os mesmos na base de dados, através de uma query SQL.

#### 4) *Web Server*

O servidor *web* é um serviço que interage com a base de dados, estabelecendo uma ligação servidor-cliente por *sockets* TCP. Além disso, também se encontra ligado com o gestor de serviços via *sockets* TCP.

No servidor *web* desenvolvemos uma API *web*, isto é, uma interface de processamento de aplicações entre um servidor da Web e um navegador da Web[9]. Assim, esta é

responsável por lidar com solicitações HTTP e fornecer uma interface para interagir com o servidor.

Neste bloco definimos as diversas rotas que conectam o *web server* com a *web interface*, permitindo o acesso pelos vários utilizadores e do administrador. Desta forma, através destas rotas o *web server* recebe indicações que envia para o gestor de serviços.

O *web server* liga-se ao *web interface* por meio de uma comunicação pelo protocolo HTTP, este recebe solicitações HTTP dos clientes e envia as respostas adequadas.

De forma geral, o bloco do *web server* é responsável pela visualização e manipulação dos dados guardados pela base de dados no sistema central.

*5) Web Interface*

A *web interface* é a parte da aplicação acessível pelos utilizadores através de um navegador web. Permite aos utilizadores interagir com a aplicação, enviar dados e receber informações.

Na interface, distinguem-se 2 tipos de utilizadores no sistema: utilizadores e o administrador. O administrador é autoridade máxima, pode realizar qualquer funcionalidade, os utilizadores podem consultar os dados que foram monitorizados pelos seus sensores.

Ao administrador são atribuídas funcionalidades como a consulta dos sistemas sensores (id dos sistemas sensores) guardados na base de dados juntamente com os utilizadores (id utilizador e email). Além disso permite alterar o período de amostragem e executar os comandos, trama *start* (criar um sistema) e trama *stop*, que são enviadas para todos os sistemas sensores/simulados.

Aos utilizadores são concebidas funcionalidades como a consulta dos sistemas sensores disponíveis (id dos sistemas sensores), que estão associados ao utilizador e guardados na base de dados, possibilidade de selecionar um desses ids para consultar os dados, e consultar os valores medidos pelos sistemas sensores; e, por fim consultar o histórico de dados dos últimos 6 meses, através da visualização de gráficos para cada sensor.

*6) Base de Dados*

A base de dados é relacional e foi implementada através da tecnologia *MySQL*. Esta é acedida pelo *web server* e pelo gestor de serviços e possui 4 entidades: o utilizador, o sistema sensores, o sistema sensores utilizadores e a amostra.

No utilizador os dados guardados são: o identificador do utilizador, o e-mail, a palavra-passe e um booleano para verificar se é o administrador. No sistema sensores: o identificador do sistema. Na amostra: o identificador do sistema, a data e a hora do evento, e os dados dos sensores, como temperatura, humidade, movimento e luminosidade, é lhe associada a um identificador de utilizador. Se algum dado das medições estiver vazio, significa que não existe esse sensor no sistema. No sistema sensores utilizadores, estão associados os identificadores do utilizador e do sistema sensor, e serve para estabelecer a associação entre o sistema sensor e o utilizador.

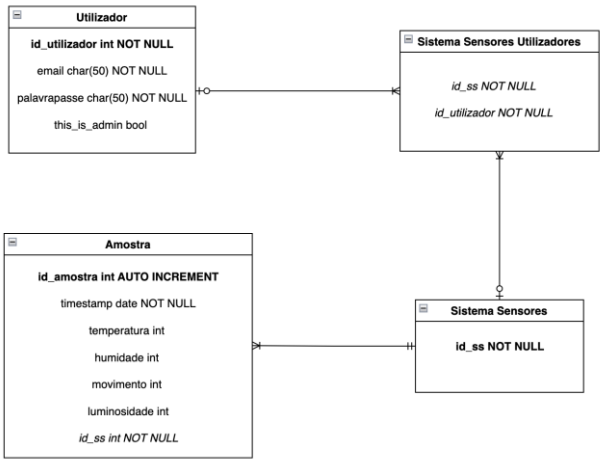


Figura 2 - Diagrama de entidades e relacionamentos.

*B. Protocolos de Comunicação*

Para estabelecer o protocolo de comunicação entre o sistema sensor e o gestor de serviços foi necessário definir um conjunto de mensagens. Desta forma definiu-se as mensagens de START e STOP.

As tramas são constituídas por um campo designado tipo (que pode variar entre START e STOP) e um campo *payload*, que contém o conteúdo útil a ser transmitido, referente a cada tipo de trama.

*1) Trama START*

A trama START representa o desejo do gestor de serviços de receber os valores proveniente dos sensores. Desta forma, quando o gestor de serviços receber informação do *web server*, evia o comando e o intervalo de amostragens pretendido. O campo de identificação é representado por 2 bytes, o primeiro correspondente ao caracter ASCII "c" simbolizando um comando e o caracter ASCII "s" que simboliza *start.* O payload envia o tempo de amostragem, que corresponde a 2 bytes.

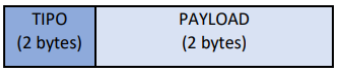


Figura 3 - PDU start.

*2) Trama STOP*

Contrariamente ao sucedido na trama START, a trama STOP representa o desejo do gestor de serviços de interromper a amostragem dos valores. Por exemplo, se for pretendido alterar o período de amostragem, é enviada uma trama STOP para identificar esta ação. O campo de identificação é representado por 2 bytes, o primeiro correspondente ao caracter ASCII "s" simbolizando um comando e o caracter ASCII "t" que simboliza stop . O payload envia o tempo de amostragem, que corresponde a 2 bytes, e tem sempre o valor de 0.

| TIPO (2 bytes) | PAYLOAD (2 bytes) |
|---|---|

Figura 4 - PDU stop.

## IV. Implementação

### A. *Sistema sensor*

O sistema sensor foi desenvolvido na ferramenta de desenvolvimento *Arduino IDE*, e é constituído pela placa ESP32-DEVKITC-32E, o sensor PIR, DHT-11 e LDR. Para simbolizar os atuadores usamos dois leds. Caso a temperatura exceda o *threshold* um dos leds é acionado e caso seja detetado movimento o outro led executa o mesmo comando.

Para conectar a placa ESP32 ao computador utilizamos um cabo *USB* e para ligar a placa ESP32 aos sensores utilizamos uma *breadboard*.

Na plataforma de desenvolvimento recorremos á instalação de bibliotecas de modo lidar com cada sensor: biblioteca do sensor DHT-11[9], PIR[10] e LDR[11].

Na imagem x é apresentado o circuito montado do sistema.

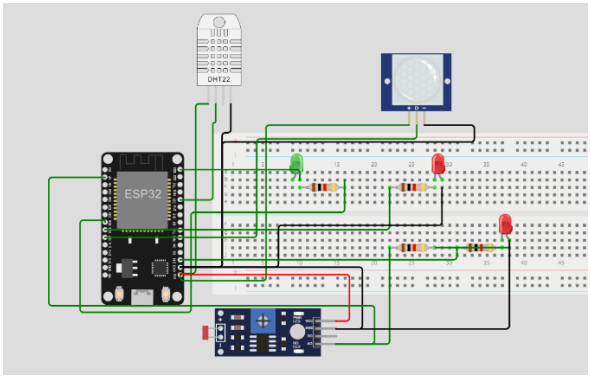


Figura 5 - Montagem do circuito.

O sistema sensor estabelece uma ligação TCP com o gestor de serviços, agindo como um cliente. Aguarda receber mensagens do servidor, e caso receba o comando "cs" e retira o período de amostragem. Quando recebe a trama start, são lidos dados dos sensores de 1 em 1 segundo consoante o número de amostras (definidas pelo período de amostragem), posteriormente, é feita uma média desses valores e estes são enviados para o gestor de serviços consoante o período de amostragem. Caso receba o comando "st" termina o envio de dados, que depois pode ser reiniciado quando recebe novamente a trama *start.*

O sistema sensor é identificado com um id_ss com valor 0.

No Anexo B, é apresentado o fluxograma do funcionamento do sistema sensor.

### B. *Sistema simulado*

O sistema simulado foi desenvolvido no editor *Visual Studio Code*, com recurso à linguagem de programação *Python*. Este sistema estabelece uma ligação TCP com o gestor de serviços, agindo como um cliente.

O gestor de serviços permite a conexão de mais do que um sistema simulado. O sistema aguarda receber uma mensagem do servidor, e caso receba o comando "cs" retira o período de amostragem e procede ao envio de dados, caso receba o comando "st" termina o envio de dados, podendo o envio de dados ser reiniciado caso o comando "cs" seja novamente recebido. Cada sistema simulado é identificado com um id_ss.

Este sistema gera dados aleatórios, simulando o sistema sensor, assim envia os dados na forma:

*id_ss* *timestamp* *Temperatura* *Humidade* *Movimento* *Luminosidade*

No Anexo B, é apresentado o fluxograma do funcionamento do sistema simulado.

### C. *Gestor de serviços*

O gestor de serviços foi desenvolvido no editor *Visual Studio Code* utilizando a linguagem de programação *Python*. Para implementar esse bloco, criamos um servidor TCP.

O gestor de serviços fica responsável por receber conexões dos clientes TCP e estabelecer conexões tanto com o Arduino quanto com os sistemas simulados que forem iniciados. Além disso, ele também pode estabelecer conexão com o *web server*.

Esse bloco desempenha o papel de ponte entre o servidor web, o Arduino e os sistemas simulados, sendo responsável pela troca de comandos entre eles. Ele é capaz de receber um comando **cs** (que indica o *start*), juntamente com o tempo de amostragem, ou **st** (que indica o *stop*) e enviá-lo para os sistemas sensores. Em seguida, ele recebe os dados desses sistemas sensores e processa-os utilizando a função **separar_dados(message**).

Além disso, o gestor de serviços é capaz de verificar na base de dados se o **id_ss** já existe utilizando a função **verificar_ss(id_ss)**, e armazena na base de dados os dados processados anteriormente. Caso receba um comando **st**, envia esse comando para os sistemas sensores, fazendo com que eles interrompam o envio de dados, mas permaneçam em modo de escuta para serem reiniciados posteriormente.

No Anexo B, é apresentado o fluxograma do funcionamento do gestor de serviços.

### D. *Web Server*

O *web server* foi desenvolvido no editor *Visual Studio Code* com recurso à linguagem de programação *Python*.Para implementar este bloco utilizamos a ferramenta Flask, uma framework de *Python.*

O servidor *web* estabelece uma ligação TCP com o gestor de serviços e a base de dados. Para conectar o servidor *web* à base de dados utilizamos a biblioteca *connector.*

Neste bloco são definidas funções para lidar com a base de dados, como por exemplo, a **read_ss_id(connection)**, que consulta a base de dados e recupera o id_ss dos sistemas sensores; a **delete_ss(connection,id_ss)**, que consulta a base de dados e remove o id_ss referido da tabela Sistema sensores utilizadores e em seguida, retira o id do Sistema sensores; entre outras.

Posteriormente, estas funções são chamadas nas rotas criadas, por exemplo, a rota **"/delete_ss"** é acionada por uma solicitação POST, onde é obtido o id_ss da solicitação e a função **delete_ss(connection,id_ss)**, chamada para remover da base de dados. O mesmo processo acontece com a solicitação POST para o *start* e *stop*, porém, nestas é obtido

o período de amostragem da solicitação e o mesmo é enviado para o gestor de serviços.

No Anexo B, é apresentado o fluxograma do funcionamento do sistema sensor.

### *E. Web Interface*

O *web server* foi desenvolvido no editor *Visual Studio Code* com recurso à linguagem de programação *Python, JavaScript, CSS,* e *HTML.*

Para implementar o design gráfico da página recorremos à linguagem *CSS e HTML* para criar elementos como tabelas, botões, caixas e gráficos. Os botões têm ids atribuídos a eles e são usados para disparar ações quando clicados. Para definir as ações a serem executadas por eventos, como o botão acionado, desenvolvemos o código em *JavaScript.* Quando um botão é acionado, é enviado o método POST para o servidor, para este executar operações específicas e também receber dados do servidor, como o id do sistema sensor, informações dos utilizadores de modo a atualizar as tabelas da página e exibir esses dados.

A nossa interface é constituída por diferentes páginas: a página inicial, ilustrada na Figura 6; a página login, ilustrada na Figura 7; a página registar, ilustrada na Figura 8; a página contactos, ilustrada no Anexo C; a página do administrador, ilustrada na Figura 9 e por fim, a página do utilizador ilustrada na Figura 10.

Na página inicial, o administrador/utilizador pode optar por fazer login, registar ou consultar os contactos, qualquer das ações desejadas exibirá outra página.

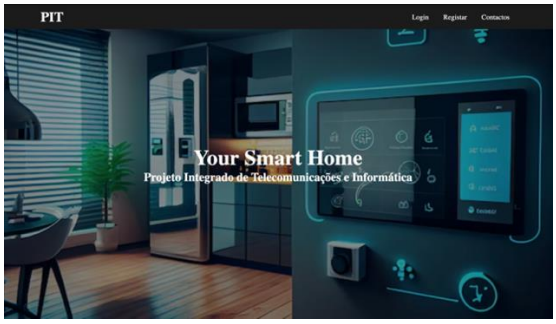


Figura 6 - Interface página incial.

Caso a opção selecionada seja o login, existe um email específico para o administrador, que se encontra guardado na base de dados e é **"camila@example.com"**. Quando este é introduzido no *login*, é feita a verificação na base de dados, pela variável **this_is_admin** e é permitido o acesso à página de administrador. Caso contrário, é permitido apenas o acesso à página do utilizador.

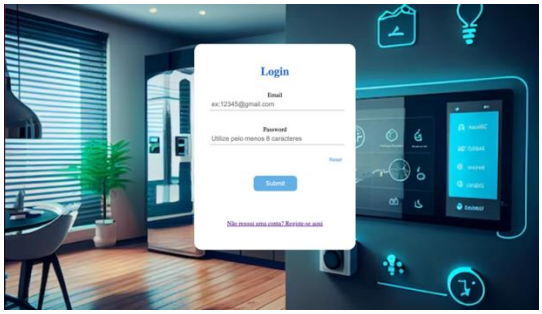


Figura 7 - Interface login.

Para o utilizador se registar é necessário que este introduza o email e a password. Caso o registo seja bem-sucedido é mostrada uma mensagem de sucesso; caso contrário é notificado do insucesso. Tanto na página de login, registar e contactos é dada a opção de "Voltar", que regressa à página inicial.

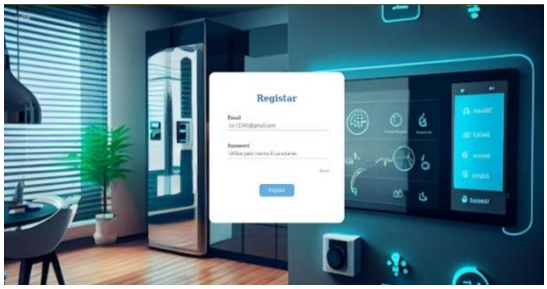


Figura 8 - Interface registar.

Na página de administrador, são dadas as funcionalidades enunciadas na secção III deste artigo. Para criar um sistema, o administrador define um período de amostragem e carrega no *start*, os dados vão começar a ser gerados e guardados na base de dados, para parar a geração de dados carrega no *stop*. Os dados dos sistemas sensores gerados vão ser guardados na base de dados, e o id_ss dos mesmos é guardado especificamente na tabela **SistemaSensor**. Na interface a tabela definida é atualizada, atualizando a página. O mesmo procedimento é estabelecido para a listagem dos utilizadores, se for registado um novo utilizador, este é guardado na base de dados e o administrador consegue ter o conhecimento dos mesmos. O administrador pode indicar o sistema que pretende eliminar, e ao carregar no botão "Remover", o mesmo é removido automaticamente da tabela e da base de dados.

Na interface é ainda indicado um botão de *logout*, que regressa para a página inicial.

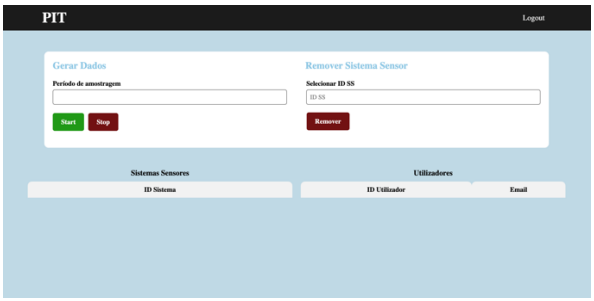


Figura 9 - Interface página administrador.

Ao utilizador é apresentada uma página com uma tabela que apresenta sempre o último valor lido da base de dados de um dado sistema.

Esse sistema é escolhido através duma *selection box* onde o utilizador pode verificar quais os sistemas que estão associados à sua conta.

Há um link que permite a visualização do histórico dos dados dos sensores através de 4 gráficos.

No menu principal há também um botão para adicionar dispositivos que abre uma página que permite a visualização

dos dispositivos disponíveis para associação através duma tabela com a enumeração dos mesmos. Caso sejam inseridas letras e/ou números de ids que não estejam disponíveis uma mensagem de erro aparecerá.

Os gráficos atualizam mediante os valores que vão aparecendo na tabela e a tabela vai sendo atualizada mediante o *refresh* da página, sendo que esta não se atualiza automaticamente.

Na interface é ainda indicado um botão de *logout*, que regressa para a página inicial.


Figura 10 - Interface página do utilizador.

## V. TESTES E RESULTADOS

Os testes foram realizados através da plataforma *Visual Studio Code*, juntamente com a ferramenta de software *MYSQL*.

Considerando que o administrador pretende gerar valores, utilizámos dois sistemas simulados e o sistema sensor. A conexão destes ao gestor de serviços pode ser comprovada na Figura 11.

```
camila@camila:~/Desktop/Fase_final$ python3 gestordeservicos.py
Servidor TCP escutando em :1238
Servidor TCP escutando em :9571
Servidor TCP escutando em :9689
Conexão estabelecida com ('192.168.1.18', 65137) no dispositivo 1238
oi_1
Conexão estabelecida com ('127.0.0.1', 60496) no dispositivo 9571
oi_2
Conexão estabelecida com ('127.0.0.1', 60508) no dispositivo 9571
oi_2
Conexão estabelecida com ('127.0.0.1', 42578) no dispositivo 9689
oi_3
```

Figura 11 - Teste de conexão.

Na página inicial, ao optarmos pelo login, utilizamos o mail **camila@example.com** e temos acesso à página do administrador. Caso o administrador pretenda criar um sistema simulado, seleciona o "*start*" e o período de amostragem (neste caso, 5), e a trama é enviada para o gestor de serviços, que por sua vez envia para os sistemas, Figura 12.

Para a trama stop o procedimento é igual, e a geração de dados é parada, podendo posteriormente ser retomada.

```
Recebido do Web Server: cs, 5
Mensagem enviada para o dispositivo na porta 1238: cs, 5
Mensagem enviada para o dispositivo na porta 9571: cs, 5
Mensagem enviada para o dispositivo na porta 9571: cs, 5
Mensagem enviada para o dispositivo na porta 9689: cs, 5
oi_3
Recebido do SS: 2 2023-05-31 19:51:31 Temperatura 12.67 Humidade 93.88 Movimen
to 0 Luminosidade 550.2
['2', '2023-05-31', '19:51:31', '12.67', '93.88', '0', '550.2']
Conectado à base de dados.
Conectado à base de dados.
Conectado à base de dados.
Dados enviados para a base de dados
oi_2
Recebido do Arduino: 0 2023-5-31  18:51:35 Temperatura 23.80 Humidade 63.00 Mo
vimento 0 Luminusidade 0.00
['0', '2023-5-31', '18:51:35', '23.80', '63.00', '0', '0.00']
Conectado à base de dados.
Conectado à base de dados.
Conectado à base de dados.
Dados enviados para a base de dados
```

Figura 12 - Teste do funcionamento do start.

Estes dados são guardados na base de dados, na tabela Amostras e o id_ss na tabela SistemaSensor, como é mostrado na Figura 13.

| id_amostra | data_hora | temperatura | humidade | movimento | luminosidade | id_ss |
|---|---|---|---|---|---|---|
| 1 | 2023-05-3119:51:31 | 13 | 94 | 0 | 550 | 2 |
| 2 | 2023-5-3118:51:35 | 24 | 63 | 0 | 0 | 0 |
| 3 | 2023-05-3119:51:36 | 25 | 55 | 0 | 620 | 2 |
| 4 | 2023-05-3119:51:36 | 43 | 92 | 1 | 535 | 1 |
| 5 | 2023-5-3118:51:40 | 24 | 63 | 0 | 0 | 0 |
| 6 | 2023-05-3119:51:41 | 64 | 63 | 1 | 237 | 1 |
| 7 | 2023-05-3119:51:41 | 34 | 89 | 0 | 282 | 2 |
| 8 | 2023-5-3118:51:45 | 24 | 63 | 0 | 0 | 0 |
| 9 | 2023-05-3119:51:46 | 26 | 62 | 0 | 128 | 1 |
| 10 | 2023-05-3119:51:46 | 56 | 5 | 1 | 564 | 2 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Figura 13 - Teste da tabela Amostras.

Estes valores vão ser listados também na tabela, Figura 14.

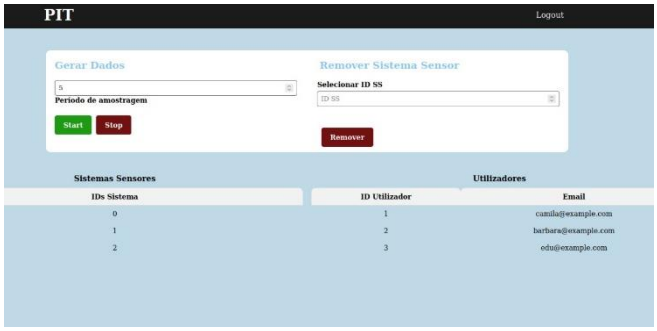


Figura 14 - Teste da interface administrador.

Por outro lado, se no login acedermos com outro email somos redirecionados para a página do utilizador. Assumindo que pretendemos adicionar um sistema à nossa conta, selecionamos "Adicionar dispositivo" obtemos os sistemas disponíveis, ilustrado na Figura 15.


Figura 15 - Teste de seleção de id.

Associando a conta ao sistema simulado com id 2, conseguimos visualizar o último valor deste id na base de dados (podemos comprovar os dados na Figura 13), dado pela Figura 16.


Figura 16 – Teste dos valores do id selecionado.

Na figura 17(exemplo da Temperatura) é possível observar de que maneira os dados irão aparecer nos gráficos.

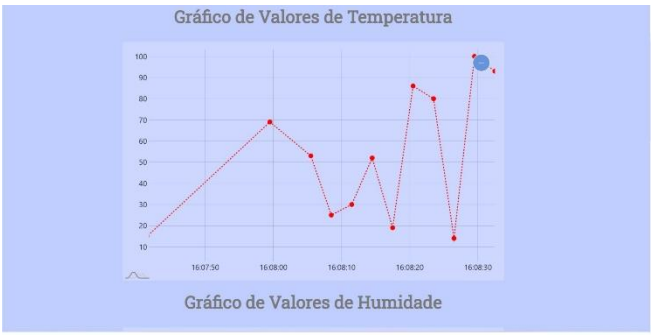


Figura 17 – Teste dos dados no gráfico.

## VI. CONCLUSÃO

Como já referido anteriormente, nos dias de hoje, o avanço da tecnologia tem impulsionado o desenvolvimento tecnológico na área da domótica com o intuito de proporcionar maior conforto e segurança aos habitantes.

Nesse contexto, visando representar o cenário de uma s*mart home*, desenvolvemos um sistema sensor com auxílio a uma placa ESP32, três diferentes sensores e leds para simbolizar atuadores. Conectamos o sistema físico ao mundo digital através do conceito da Internet das Coisas.

Para permitir a monitorização e o controlo dos dados recolhidos pelos sensores, desenvolvemos uma interface web que permite o administrador iniciar e parar o envio dos dados, remover sistemas sensores e executar comandos. E permite aos utilizadores consultar os sensores associados à sua conta e os dados medido por esses sensores, e permite consultar o histórico de dados dos últimos 6 meses.

Em suma, nosso projeto demonstrou a viabilidade de conectar dispositivos físicos numa *smart home* e integrá-los ao mundo digital através da Internet das Coisas. Ao fornecer uma interface web para monitoramento e controlo, proporcionamos aos utilizadores uma maneira conveniente de interagir com o sistema e obter informações significativas sobre o ambiente da casa em tempo real.

Para trabalhos futuros, gostaríamos de nos concentrar em aprimorar a integração de dispositivos e aprimorar a usabilidade da interface web, bem como explorar novos recursos, como a biblioteca *REACT*[13],e funcionalidades para desenvolver a *smart home*..

## REFERÊNCIAS


[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” IEEE Communications Surveys and Tutorials, vol. 17, no. 4, pp. 2347–2376, Oct. 2015.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] A. M. Mendes, “Torne a sua casa numa Smart Home e reduza a sua despesa energética”, Doutor Finanças, 15-fev-2023. https://www.doutorfinancas.pt/vida-e-familia/torne-a-sua-casa-numa-smarthome-e-reduza-a-sua-despesa-energetica/.

[3] View of Desenvolvimento de produtos IOT / IOT products development. (n.d.). Com.Br. Retrieved May 25, 2023, from https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/23059/18534

[4] Hayes, A. (2017, January 13). *Smart home: Definition, how they work, pros and cons*. Investopedia. https://www.investopedia.com/terms/s/smart-home.asp

[5] R. E. F. Figueiredo, “Desenvolvimento de um sistema de controlo e monitorização residencial através de tecnologias associadas à Internet das Coisas”, 2019.

[6] U. Pujaria, P. Patil, N. Bahadure, e M. Asnodkar, “Internet of things based integrated smart home automation system”, SSRN Electron. J., 2020.

[7] D. C. 12v G. Alarm, “SPIR Sensor Module Memo,” Sparkfun.com. https://www.sparkfun.com/datasheets/Sensors/Proximity/SE-10.pdf.

[8] Botnroll.com. https://www.botnroll.com/pt/luz-imagem/3617-vt90n2-ldr-1k-5mm-sensor-deluz.html.

[9] (N.d.). Amazon.com. Retrieved May 26, 2023, from https://aws.amazon.com/pt/what-is/api/

[10] DHT-sensor-library: Arduino library for DHT11, DHT22, etc Temperature & Humidity Sensors. https://github.com/adafruit/DHT-sensor-library

[11] R. Tillaart, Libraries/PIR at master · RobTillaart/Arduino. https://github.com/RobTillaart/Arduino/tree/master/libraries/PIR

[12] Q. Comte-Gaz, Arduino-Light-Dependent-Resistor-Library: Photocell (LDR) library for Arduino (optimized for GL55xx series). https://github.com/QuentinCG/Arduino-Light-Dependent-Resistor-Librarhttps://github.com/QuentinCG/Arduino-Light-Dependent-Resistor-Librar

[13] React native. (n.d.). Reactnative.dev. Retrieved May 29, 2023, from https://reactnative.dev/

## Anexo A - Fluxogramas

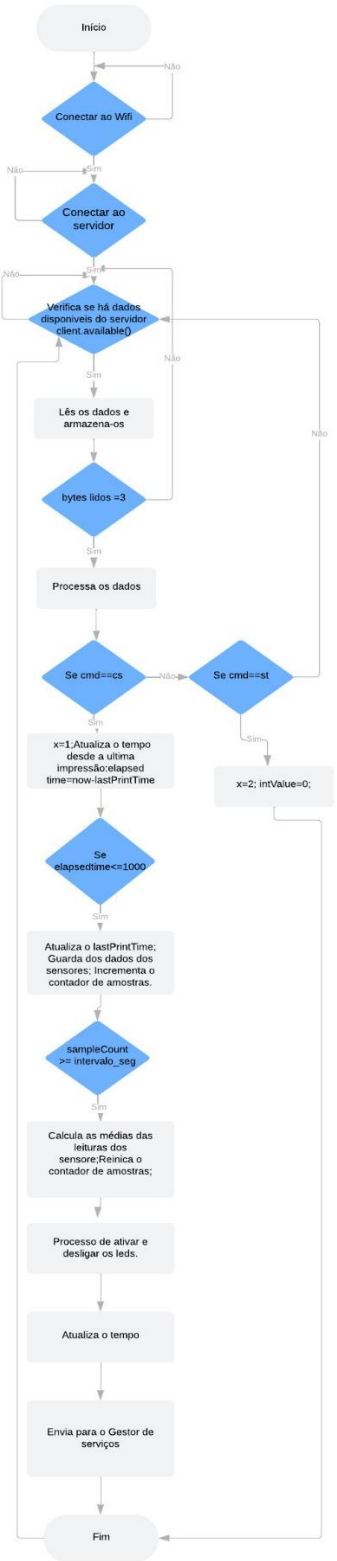


Figura A1 – Fluxograma do sistema sensor.

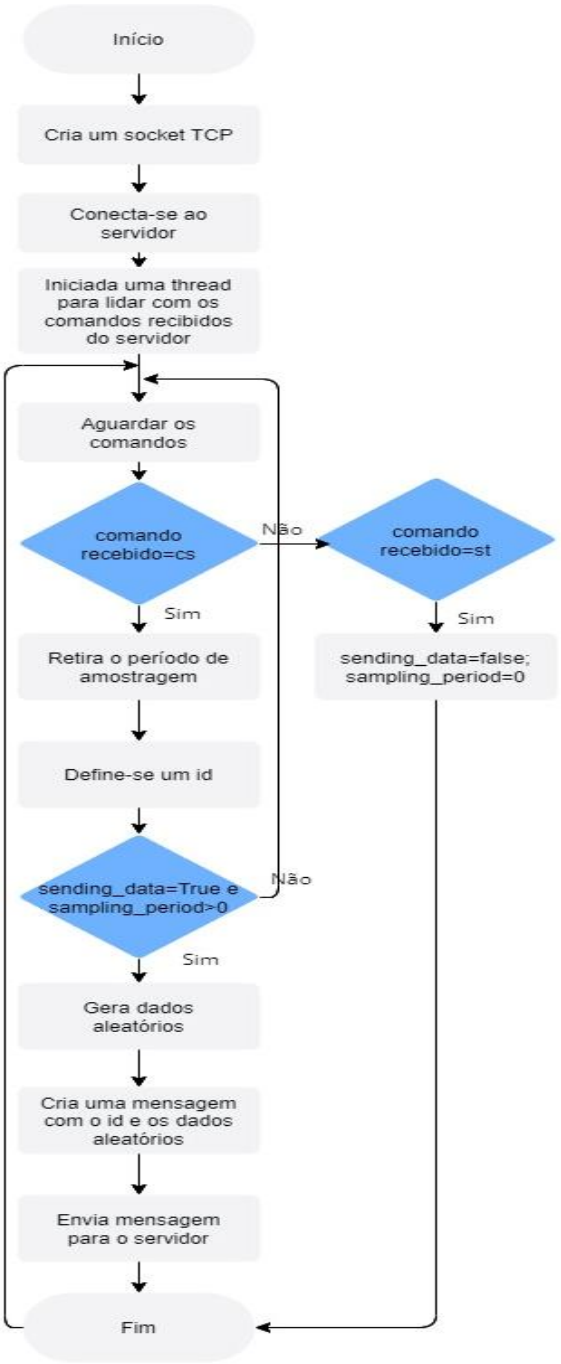


Figura A2 – Fluxograma do sistema simulado.

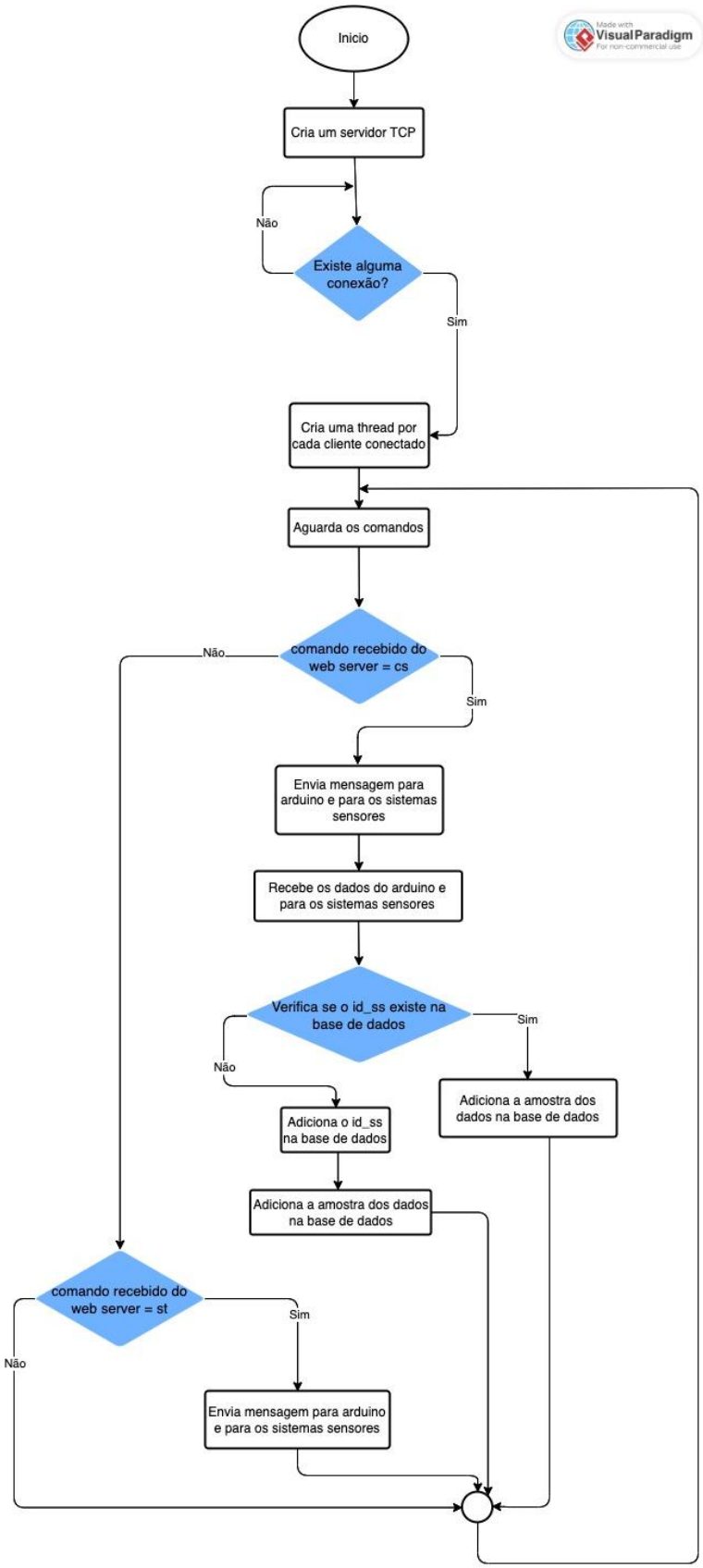


Figura A3 – Fluxograma do gestor de serviços.

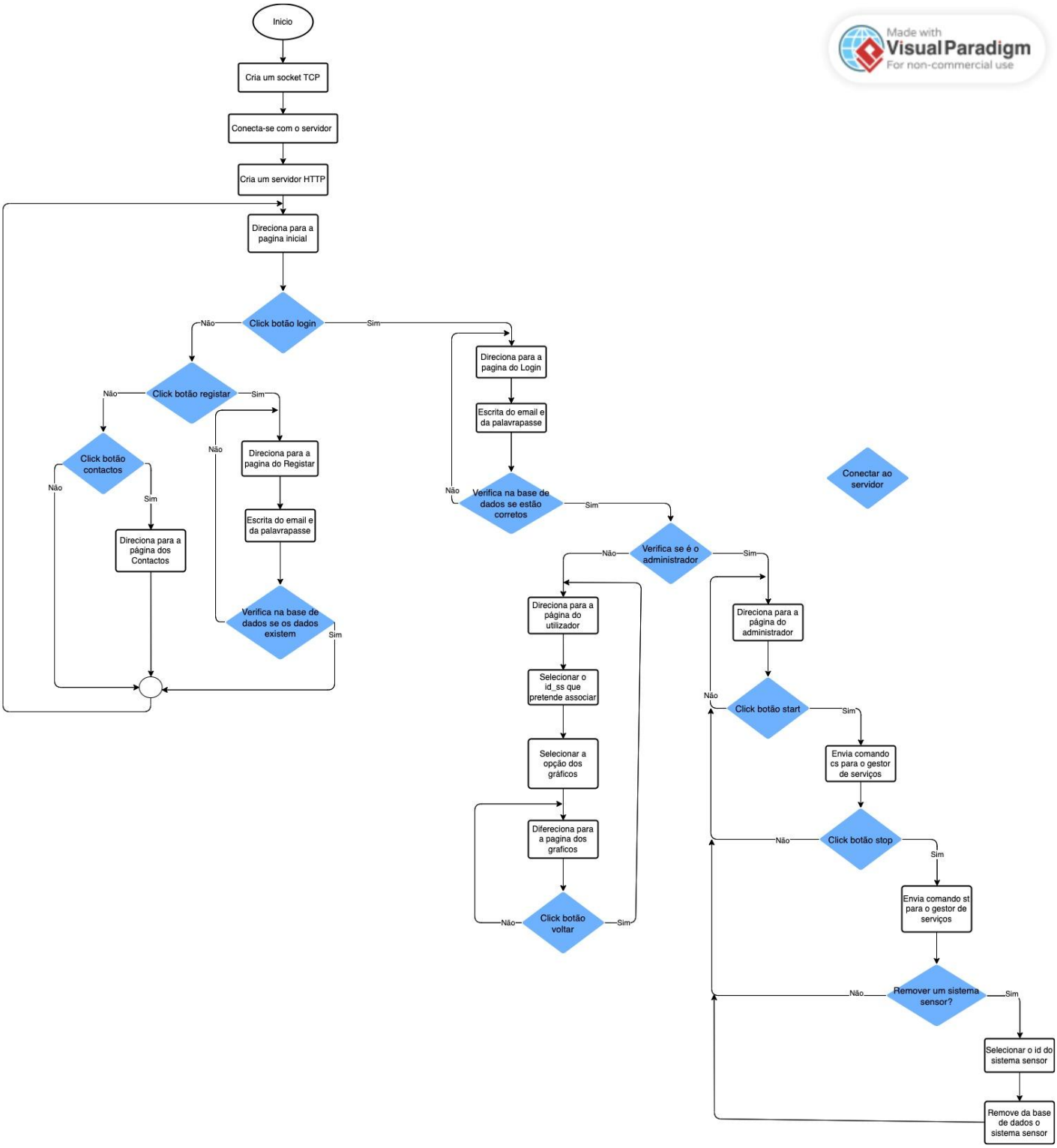


Figura A4 – Fluxograma do web server.

Anexo B - Diagramas de caso de uso

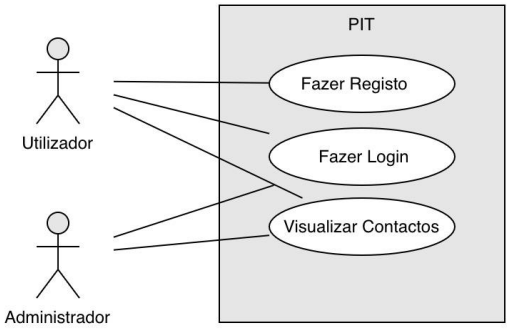


Figura B1 - Caso de uso incial.

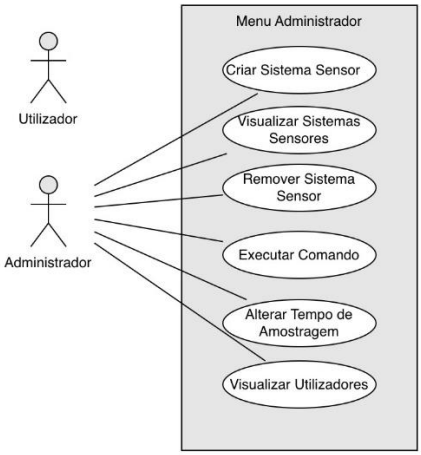


Figura B2 - Caso de uso menu administrador.

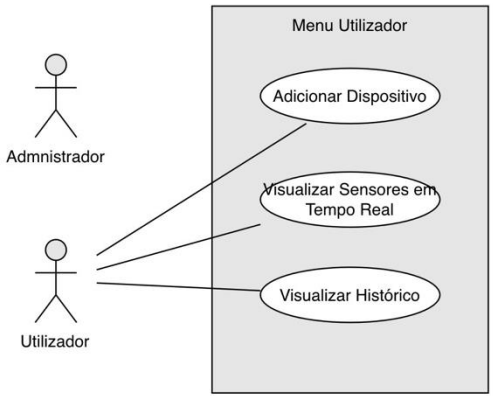


Figura B3 - Caso de uso menu utilizador.

Anexo C - Diagramas de sequência

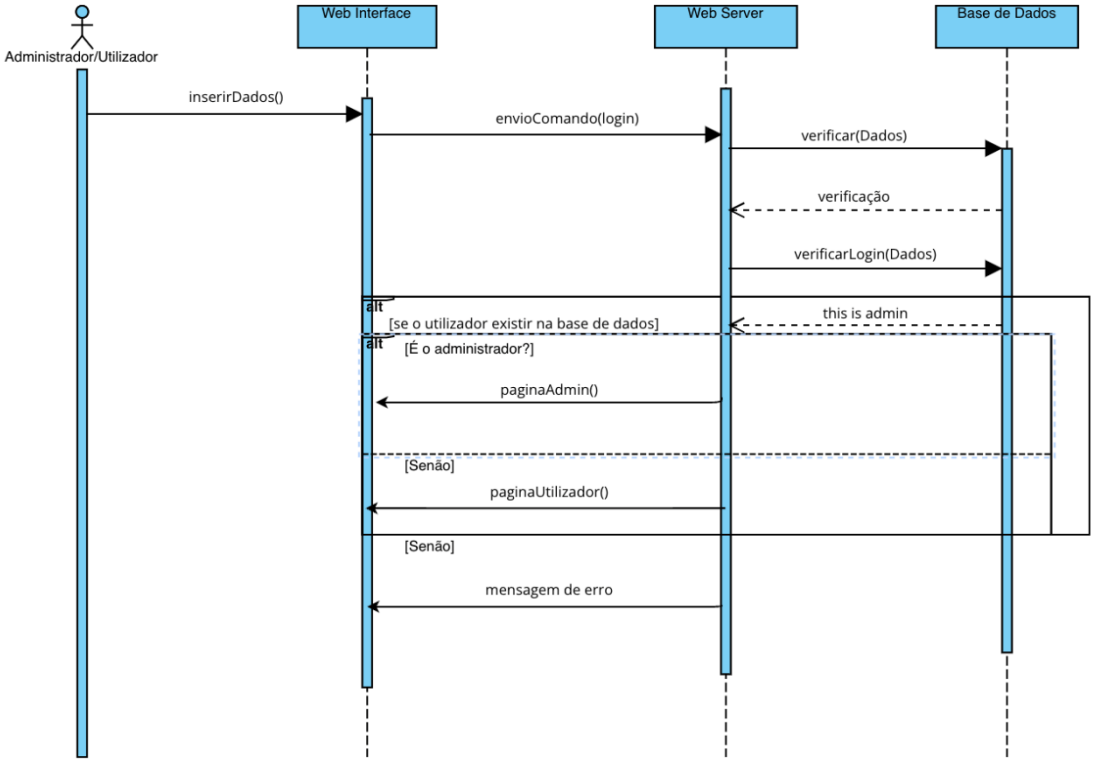


Figura C1 - Diagrama de Sequência-Login.

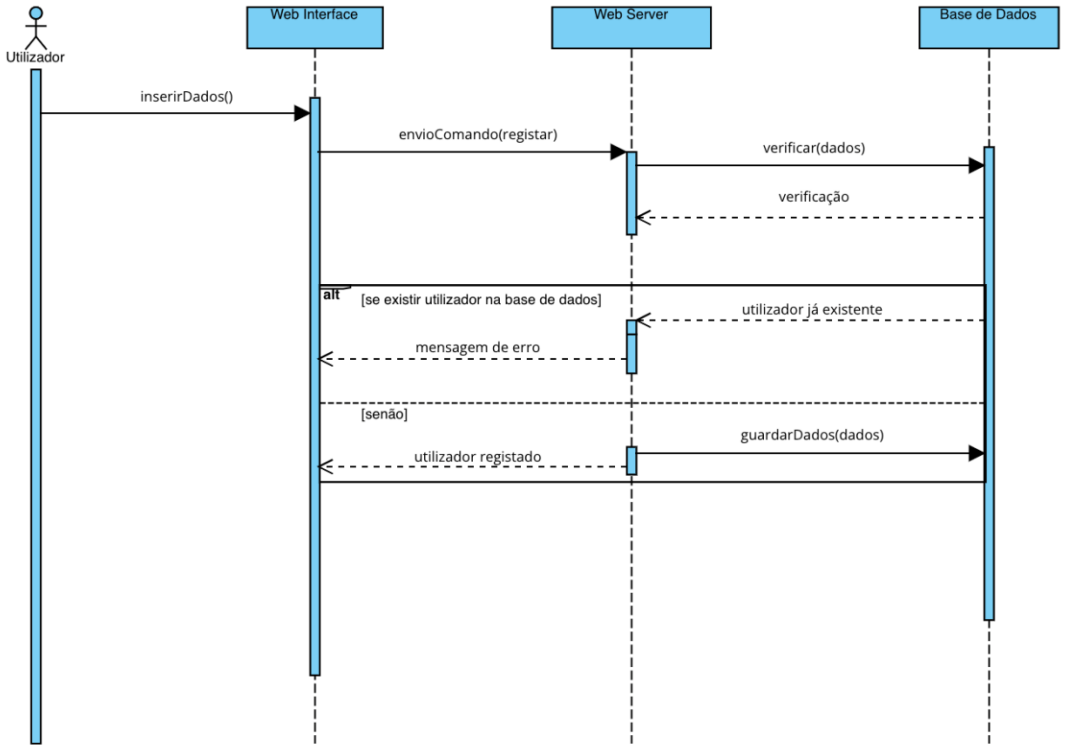


Figura C2 - Diagrama de Sequência-Registar.

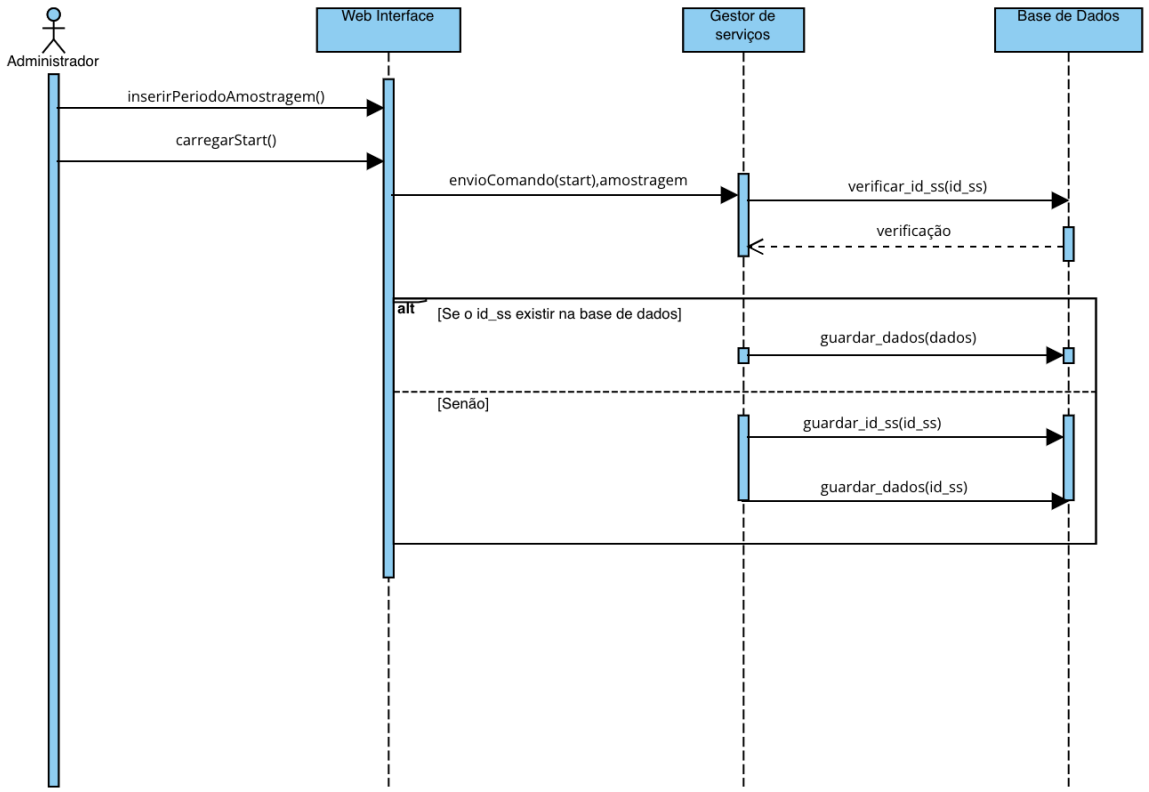


Figura C3 - Diagrama de Sequência-Start.

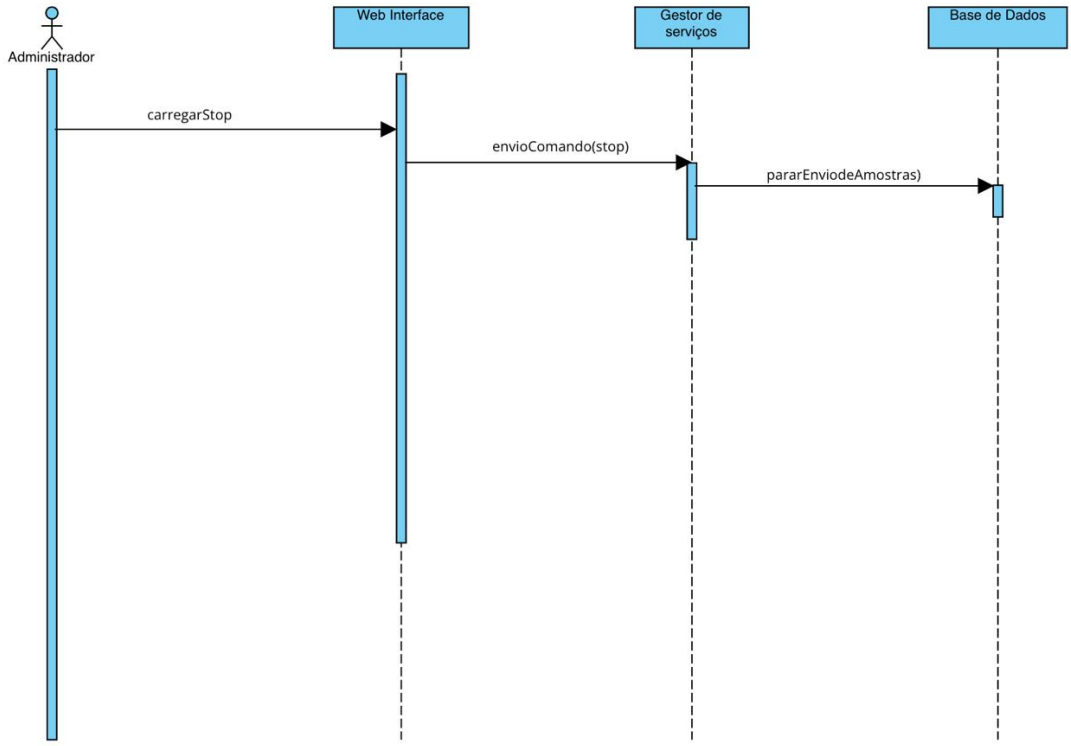


Figura C4 - Diagrama de Sequência-Stop.

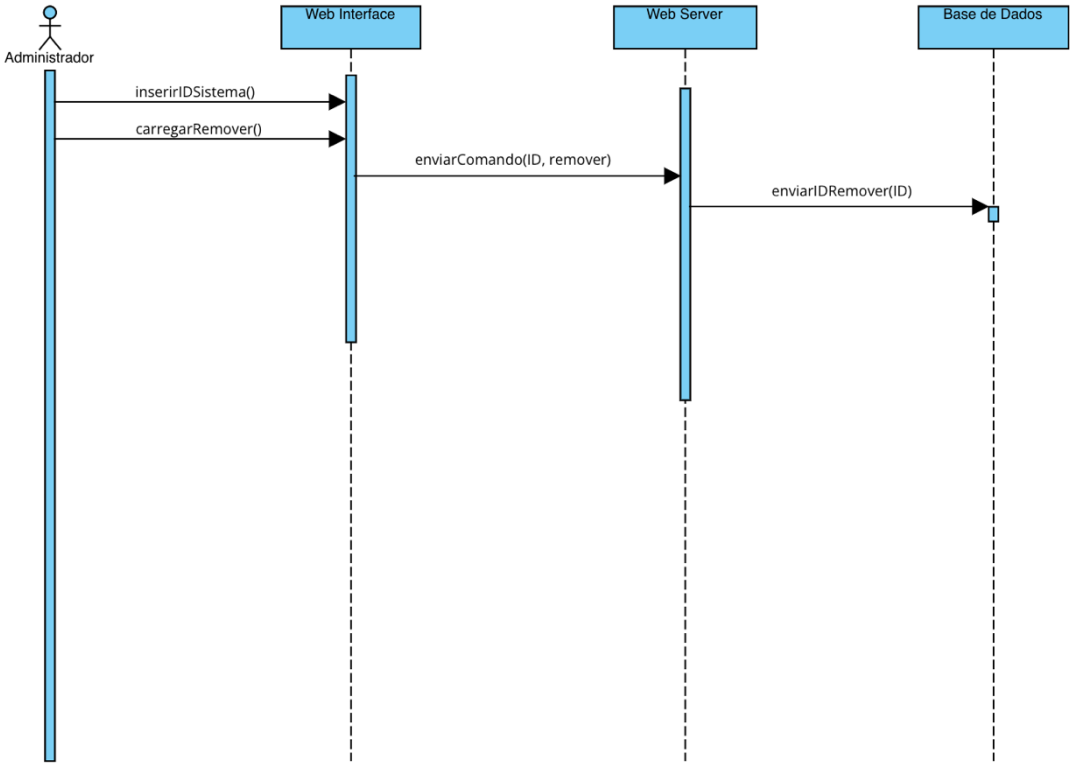


Figura C5 - Diagrama de Sequência-Remover Sensor.

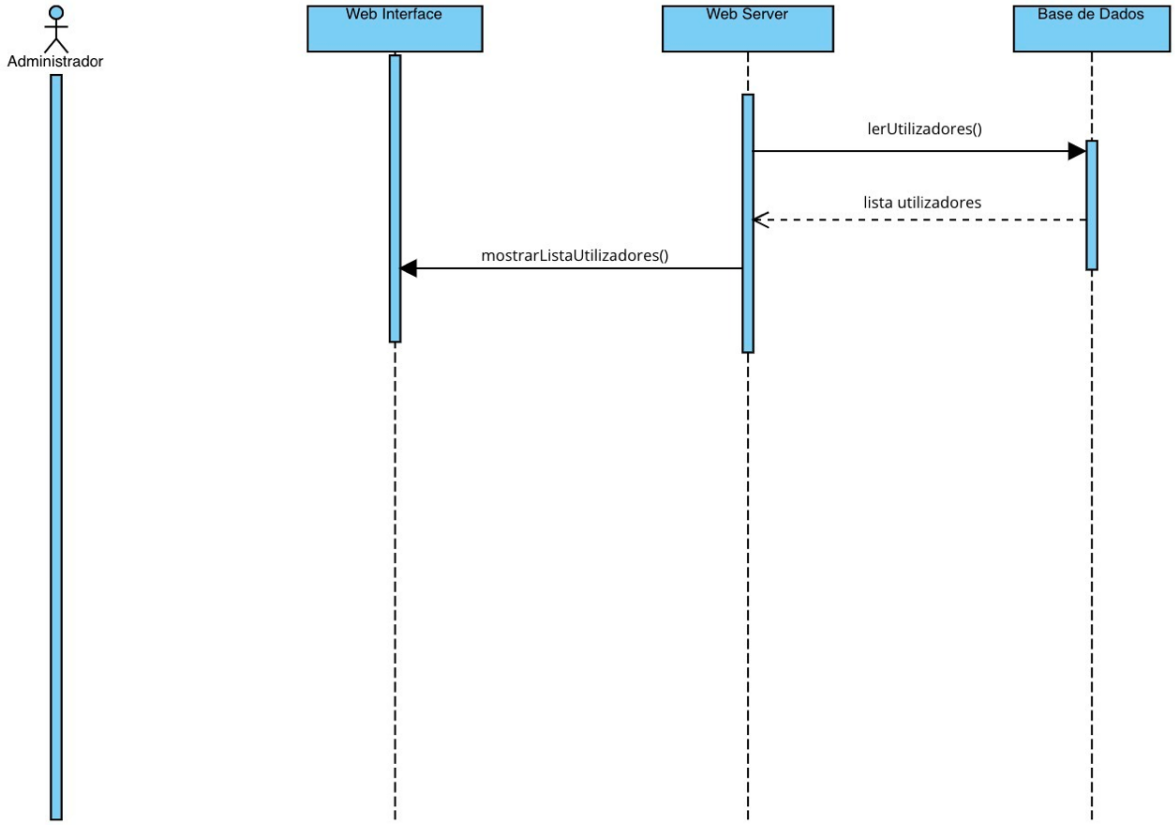


Figura C6 - Diagrama de Sequência-Listar Utilizadores.

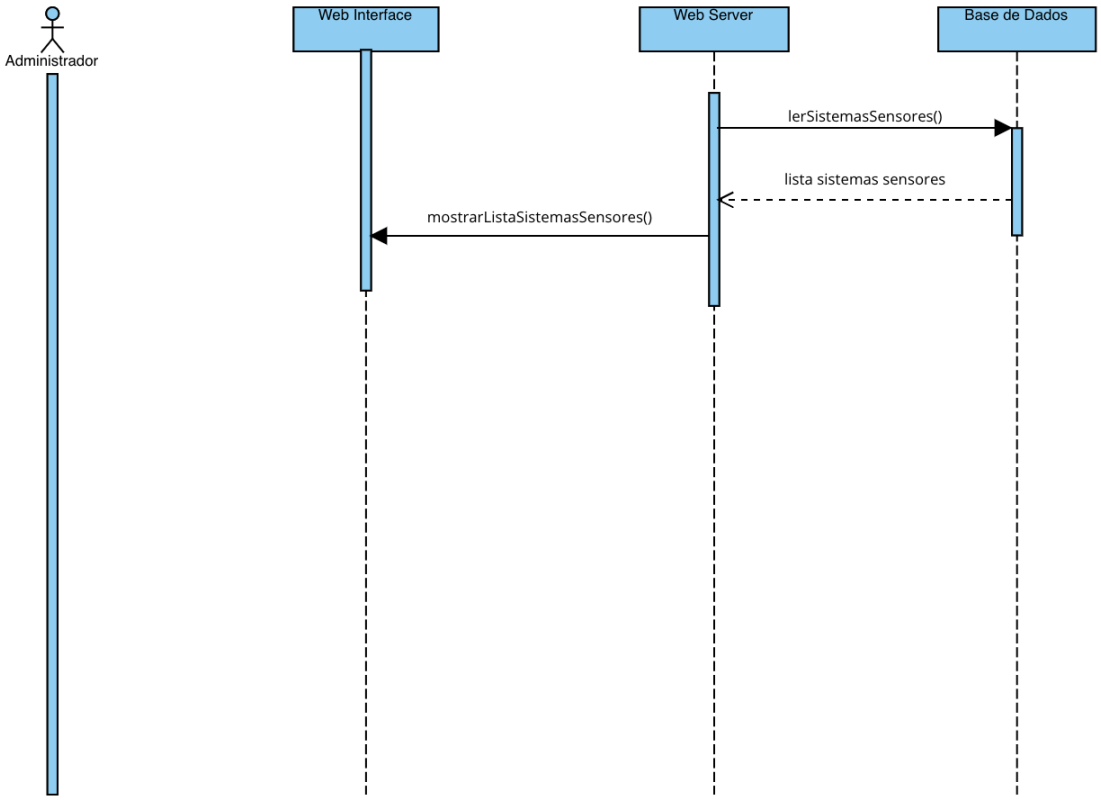


Figura C7 - Diagrama de Sequência-Listar Sensores.

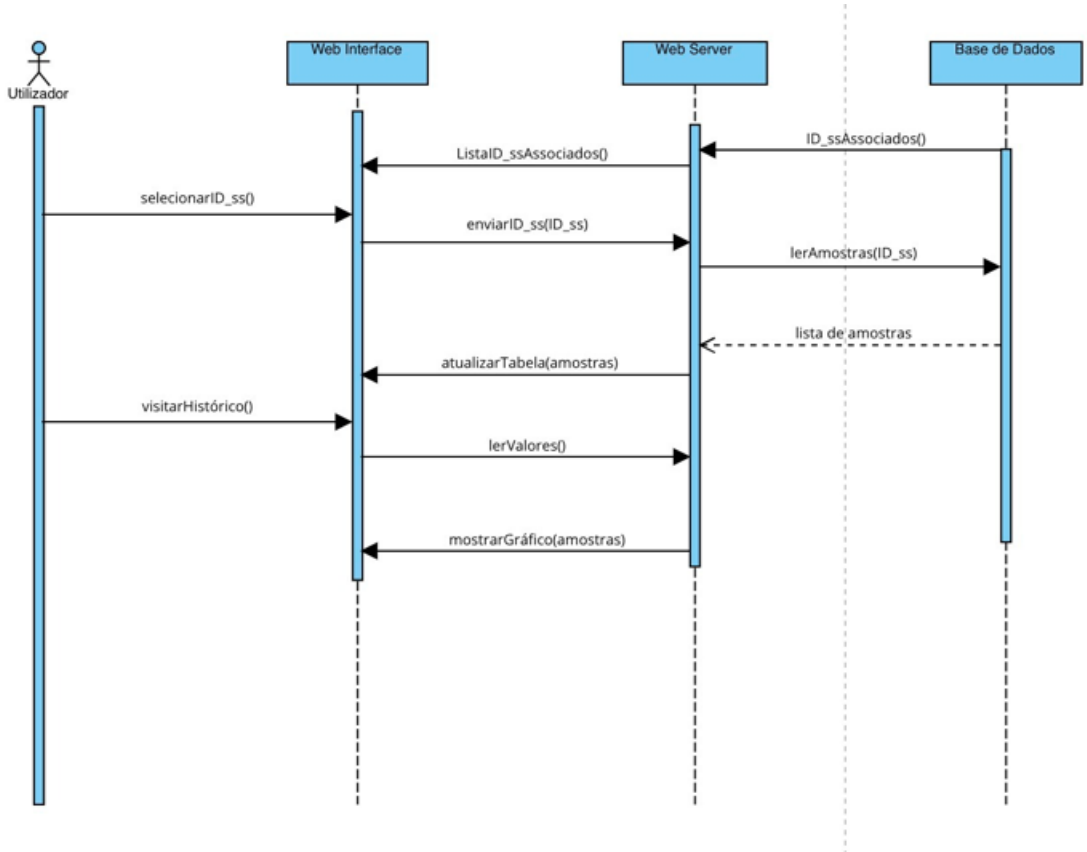


Figura C8 - Diagrama de Sequência- Visualizar histórico.

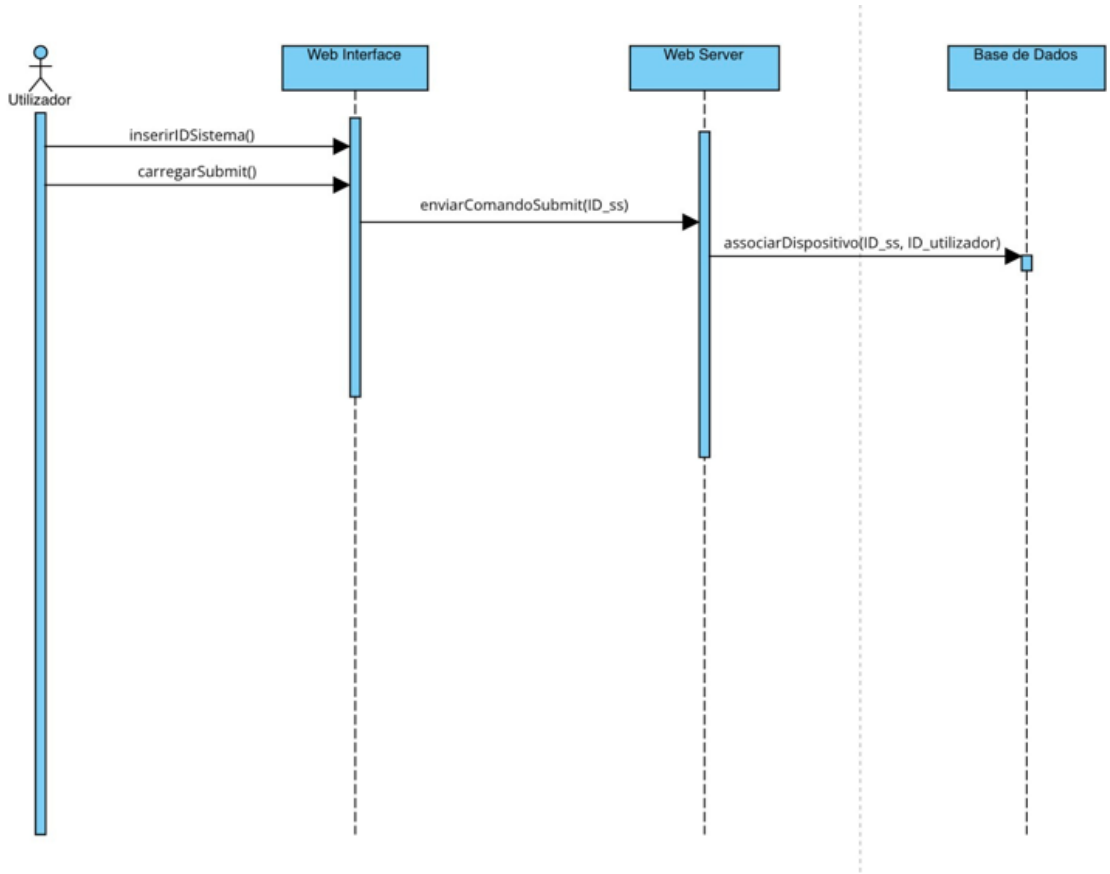


Figura C9 - Diagrama de Sequência- Adicionar dispositivo.

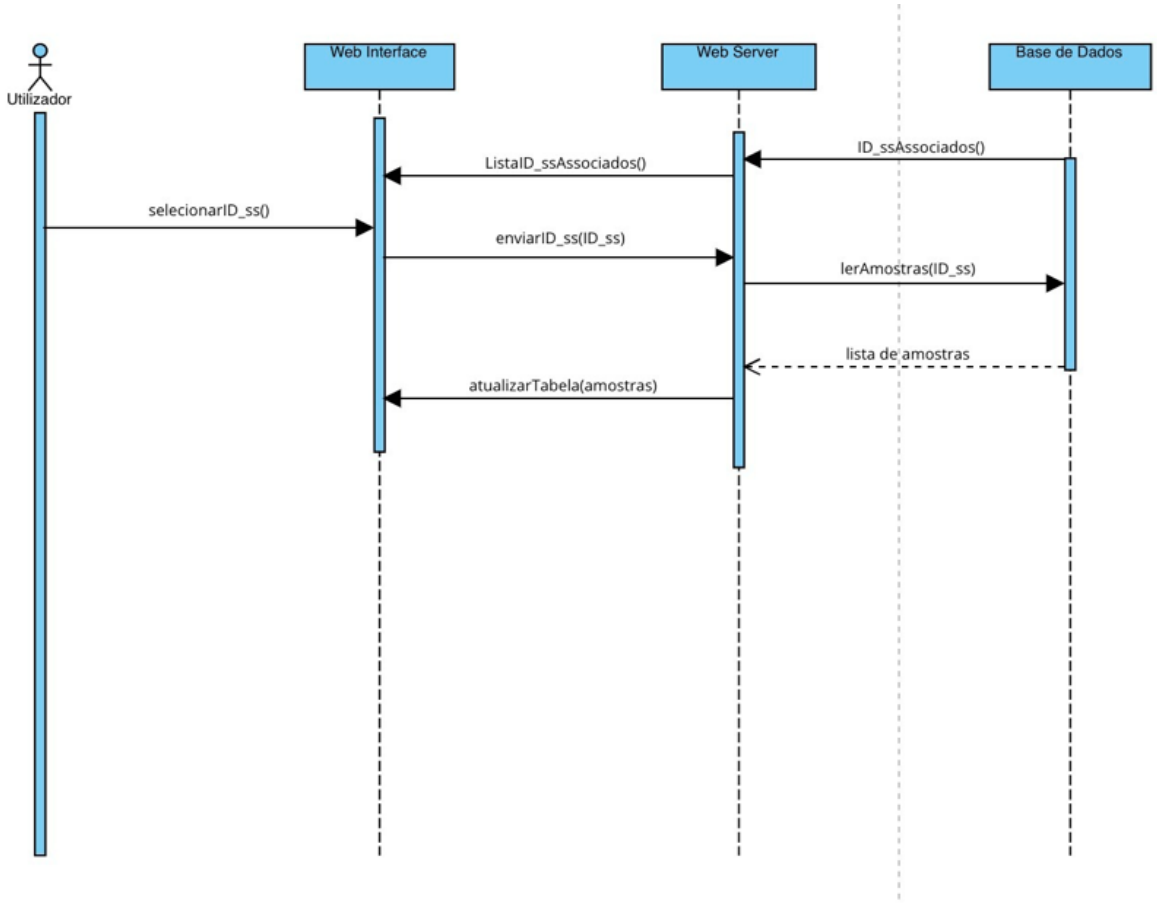


Figura C10 - Diagrama de Sequência- Visualizar Sensores em Tempo Real

Anexo D - Páginas extra *web interface*


Figura D1 - Interface página contactos

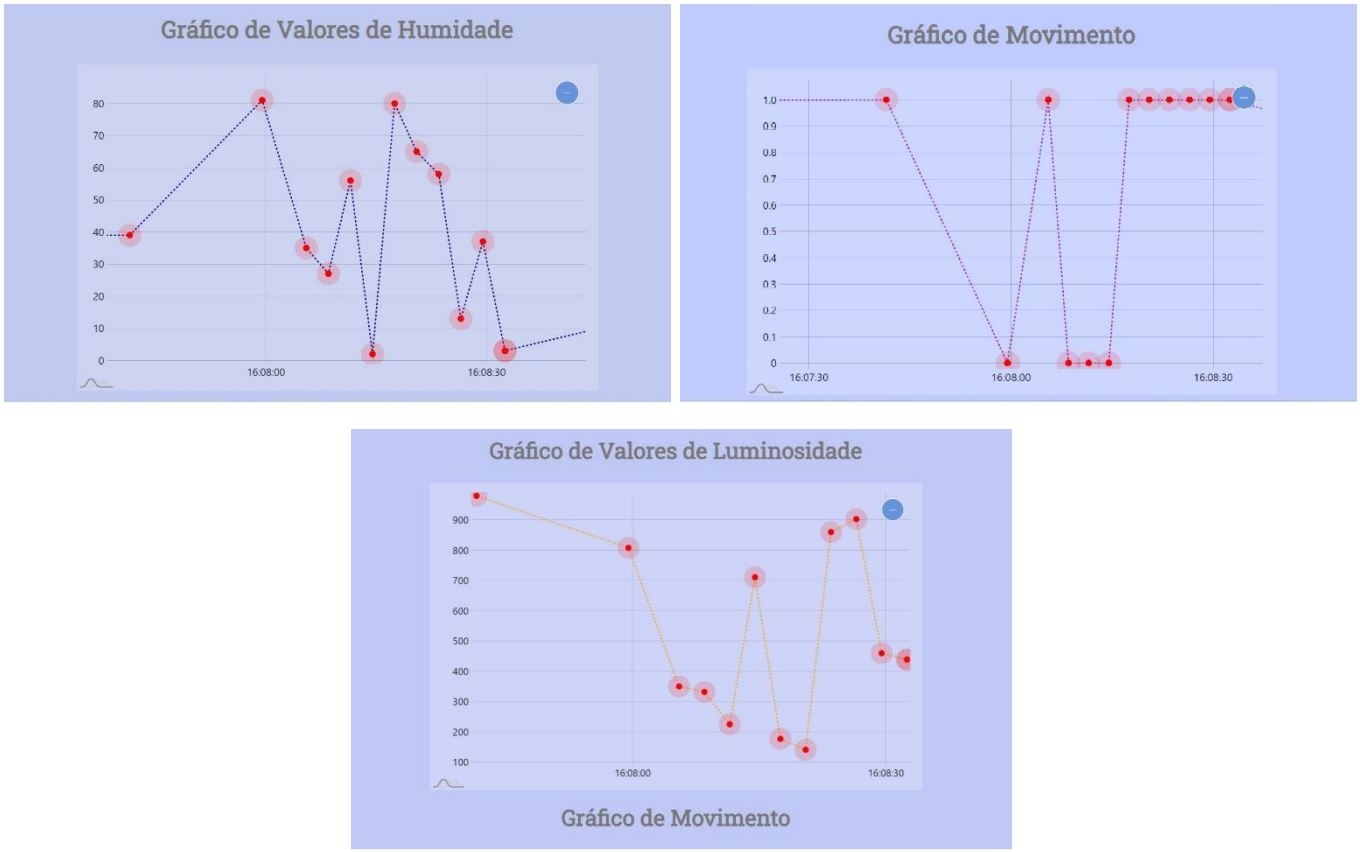


Figura D2 – Gráficos da página de utilizador.

Anexo E - Testes extra

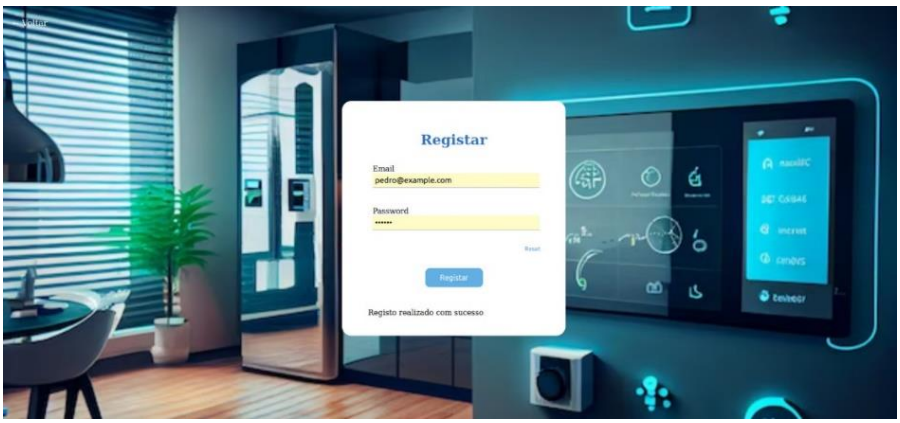


Figura E1 - Registo de um utilizador novo.

| # | id_utilizado | email | palavrapasse | this_is_admi |
|---|---|---|---|---|
| 1 | 1 | camila@example.com | camila | 1 |
| 2 | 2 | barbara@example.com | barbara | 0 |
| 3 | 3 | edu@example.com | edu | 0 |
| 4 | 4 | pedro@example.com | camila | 0 |
| * | NULL | NULL | NULL | NULL |

Figura E2 - Registo de um utilizador novo na base de dados.

```
camila@camila:~/Desktop/Fase_final$ python3 ss_pit.py
Recebido comando start com período de amostragem 5
Enviando: 2 2023-05-31 19:51:31 Temperatura 12.67 Humidade 93.88 Mo
vimento 0 Luminosidade 550.2
Enviando: 2 2023-05-31 19:51:36 Temperatura 25.33 Humidade 54.87 Mo
vimento 0 Luminosidade 620.24
Enviando: 2 2023-05-31 19:51:41 Temperatura 34.06 Humidade 89.21 Mo
vimento 0 Luminosidade 281.63
Enviando: 2 2023-05-31 19:51:46 Temperatura 56.03 Humidade 5.45 Mov
imento 1 Luminosidade 563.56
Recebido comando stop, parando o envio de dados
```

Figura E3 - Comando start e stop no sistema simulado.

```
19:48:56.535 -> Connecting to NOS-2B80_EXT
19:48:57.630 -> ..WiFi connected
19:48:59.022 -> Connected to server
19:51:31.712 -> start
19:51:31.812 -> Temp: 23.50, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:32.773 -> Temp: 23.50, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:33.800 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:34.763 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:35.792 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:35.792 -> 02023-5-3118:51:35Temperatura24.00,Humididade: 63.00,Movimento: 0.00, Luminosidade -465.00
19:51:36.787 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:37.815 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:38.779 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:39.807 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:40.769 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:40.769 -> 02023-5-3118:51:40Temperatura24.00,Humididade: 63.00,Movimento: 0.00, Luminosidade -465.00
19:51:41.798 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:42.794 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:43.789 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:44.786 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:45.814 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:45.814 -> 02023-5-3118:51:45Temperatura24.00,Humididade: 63.00,Movimento: 0.00, Luminosidade -465.00
19:51:46.775 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:47.804 -> Temp: 24.00, Humidity: 63.00, Motion: 0, Light: 0.00
19:51:48.500 -> stop
```

Figura E4 - Comando start e stop no sistema sensor.

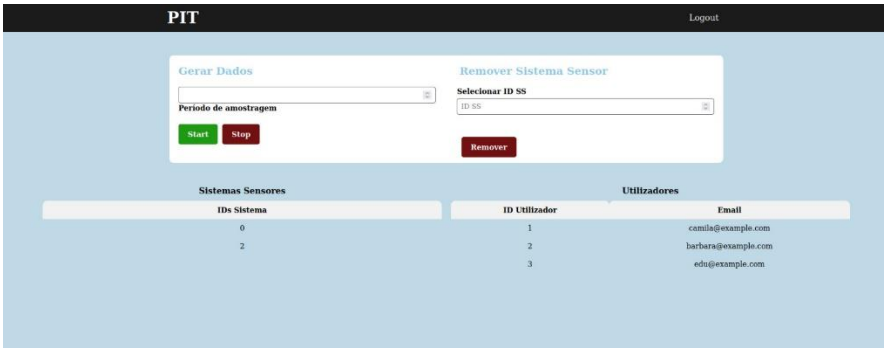


Figura E5 - Remoção do id 1 na interface.


Figura E6 - Remoção do id 1 na base de dados-tabela Sistema Sensor.

| # | id_amostra | data_hora | temperatura | humidade | movimento | luminosidad | id_ss |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 2023-05-3119:51:31 | 13 | 94 | 0 | 550 | 2 |
| 2 | 2 | 2023-5-3118:51:35 | 24 | 63 | 0 | 0 | 0 |
| 3 | 3 | 2023-05-3119:51:36 | 25 | 55 | 0 | 620 | 2 |
| 4 | 5 | 2023-5-3118:51:40 | 24 | 63 | 0 | 0 | 0 |
| 5 | 7 | 2023-05-3119:51:41 | 34 | 89 | 0 | 282 | 2 |
| 6 | 8 | 2023-5-3118:51:45 | 24 | 63 | 0 | 0 | 0 |
| 7 | 10 | 2023-05-3119:51:46 | 56 | 5 | 1 | 564 | 2 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Figura E7 - Remoção do id 1 na base de dados-tabela