

U.C. de Projeto Integrado de

Telecomunicações

Ano Letivo: **2021/2022**

**Relatório da Fase C**

**Grupo 2**

* Catarina Neves, a93088
* Eduardo Cardoso, a89627
* José Gomes, a93083
* Luís Oliveira, a89380

01/06/2022

Universidade do Minho

Mestrado Integrado em Engenharia de Telecomunicações e Informática

Índice

[1. Introdução 5](#_Toc104758791)

[2. Trabalho Relacionado 6](#_Toc104758792)

[2.1 ESP32 WebSocket: Control ESP32 Pins 6](#_Toc104758793)

[2.2 ESP32 Robot Car Using Websockets 7](#_Toc104758794)

[3. Etapas do trabalho desenvolvido 8](#_Toc104758795)

[3.1. Protocolo de comunicação 8](#_Toc104758796)

[3.2. Sistema Central 10](#_Toc104758797)

[3.2.1 Sample 11](#_Toc104758798)

[3.2.2 Security 13](#_Toc104758799)

[3.2.3 Template 13](#_Toc104758800)

[3.2.4 Resources 13](#_Toc104758801)

[3.3. Transmissão de assinaturas temporais por BLE 14](#_Toc104758802)

[3.4. Servidor *Web* 15](#_Toc104758803)

[4. Análise de resultados e testes efetuados 17](#_Toc104758804)

[5. Conclusão 19](#_Toc104758805)

[5.1. Contribuição de cada aluno 19](#_Toc104758806)

[6. Lista de referências 20](#_Toc104758807)

Índice de figuras

Figura 1 - Tarefas propostas pela fase C. 5

Figura 2 – Variação de cor da tira LED. 6

Figura 3 - Serviço web implementado pelo autor. 7

Figura 4 - Lista de componentes necessários à construção do projeto. 7

Figura 5 - Declaração da porta, do endereço e do cliente usados para a comunicação com o Sistema Central. 8

Figura 6 - Envio da trama de dados e finalização da impressão na socket, no Gateway. 8

Figura 7 - Inicialização da socket servidor no sistema. 9

Figura 8 - Ficheiros presentes na diretoria sample. 10

Figura 9 - Ficheiros presentes na diretoria security. 10

Figura 10 - Ficheiro presente na diretoria template. 10

Figura 11 - Constituição da diretoria resources. 10

Figura 12 - Estrutura da classe WeatherSample. 11

Figura 13 - Atributos da tabela weather. 11

Figura 14 - Estrutura da classe WeatherSampleService. 12

Figura 15 - Estrutura da classe WeatherSampleService. 12

Figura 16 - Interações entre as camadas do gestor de serviço. 12

Figura 17 - Estrutura da classe ApplicationSecurityConfig. 13

Figura 18 - Estrutura da classe TemplateController. 13

Figura 19 - Ficheiro com as configurações da base de dados. 13

Figura 20 - Declaração da característica temporal no servidor. 14

Figura 21- População do array a ser enviado pela característica. 14

Figura 22 - Leitura do valor temporal. 14

Figura 23 - Funções de conversão da data para o formato necessário. 14

Figura 24 - Página com todas as amostras. 15

Figura 25 - Página inicial do servidor web. 15

Figura 26 - Amostras armazenadas na base de dados. 17

Figura 27 - Amostras impressas no serviço web. 17

Figura 28 - Script que efetua o parsing das amostras. 17

Figura 29 - Definição das amostras hardcoded. 18

Figura 30 - Armazenamento e impressão das amostras hardcoded. 18

Lista de siglas e acrónimos

**SQL** *Structured Query Language*

**LED** *Light Emitting Diode*

**RGB** *Red Green Blue*

**GPIO** *General Purpose Input/Output*

**IDE** *Integrated Development Environment*

**IOT** *Internet Of Things*

# Introdução

Serve o presente relatório como síntese do trabalho desenvolvido e implementado no decorrer desta última fase. Este contém a descrição das estratégias e algoritmos adotados pelo grupo, tal como os respetivos testes realizados. Além dos tópicos acima citados, faz-se referência a trabalhos ou projetos similares, que se enquadram na ótica deste projeto.

O relatório abordará cada etapa desta fase de forma detalhada, ou seja, será apresentada a resposta ou proposta de solução empregue pelo grupo em junção com as ferramentas que foram necessárias para a sua construção.

A figura 1 ilustra o sumário das diferentes tarefas propostas nesta fase.

Uma imagem com texto, símbolo

Descrição gerada automaticamente

Figura 1 - Tarefas propostas pela fase C.

Nesta fase, o objetivo é a conclusão do Sistema Central, com funcionalidades mais avançadas, tais como monitorização, administração e segurança; e a aprimoração do protocolo de comunicação entre este sistema e os *gateways* (*Gateway* e Sistemas Simulados), para uma aplicação mais eficiente e fiável.

O Sistema Central recolhe a informação enviada por todos os *gateways*, que por sua vez, será armazenada na base de dados SQL. Sempre que o serviço *web* queira atualizar os dados apresentados, este faz um pedido ao gestor de serviço que, de forma encadeada, irá buscar os dados à base de dados e enviará os mesmos para o serviço *web*.

# Trabalho Relacionado

Relativamente aos trabalhos relacionados, enquadram-se os projetos seguintes que serviram de base à construção deste projeto.

## 2.1 ESP32 WebSocket: Control ESP32 Pins

Este projeto [1] demonstra a implementação de um servidor ESP32, do tipo *WebSocket,* que é responsável pelo controlo dos pinos GPIO (*General Purpose Input/Output*) de uma placa ESP32. Para além da implementação do servidor, são apresentados os passos necessários à construção de uma página *web*, que permite efetuar o controlo dos pinos referidos.

Para realizar o efeito pretendido, o autor recorreu a uma fita LED (*Light Emitting Diode)* RGB (*Red Green Blue*), cujas cores são alteradas de forma remota (através do *browser*).

O autor recorreu ao protocolo de comunicação “WebSocket”, que permite a interação entre o cliente (*web browser*)e o servidor *web* (ESP32 WebSocket), isto é, ocorre troca de informação entre a página *web* e o servidor. Desta forma, o servidor manipula os pinos, consoante os dados que são introduzidos na página *web*.

O funcionamento deste projeto pode ser ordenado de acordo com as etapas seguintes:

1. Escolher a cor pretendida através de um web *browser.*
2. Envio da informação (cor selecionada) através do protocolo WebSocket.
3. A placa ESP32 (servidor WebSocket) recebe a informação e altera a cor do LED.

É de realçar o cuidado do autor, no tratamento do conjunto de eventos associados ao protocolo aplicado, nomeadamente: conexão e desconexão.

A figura 2 ilustra o resultado pretendido, aquando da finalização o projeto.

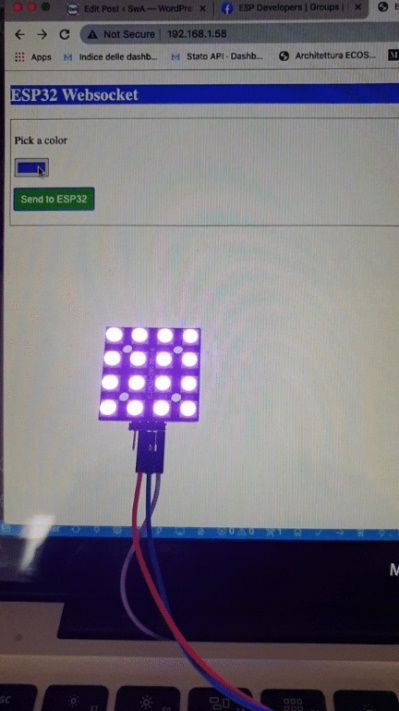


Figura 2 – Variação de cor da tira LED.

## 2.2 ESP32 Robot Car Using Websockets

Este projeto [2], da autoria de donsky, consiste num carro robot construído com o auxílio de uma placa ESP32 e de um motor LM298, que por sua vez, é controlado através de um smartphone. Este projeto foi desenvolvido com recurso ao IDE (Integrated Development Environment) PlatformIO, uma vez que este é favorável e adequado a projetos que se enquadram no domínio da IOT (Internet Of Things).

O cerne deste projeto encontra-se na utilização do protocolo de comunicação “*WebSockets”.* O protocolo citado é essencial e preenche todos os requisitos e tópicos, que são exigidos por aplicações do ramo da IOT, devido à pouca latência oferecida pelas *websockets.*

Neste contexto, o carro *robot* desempenha o papel de servidor e o *web browser* desempenha o papel de cliente. O cliente envia um conjunto de mensagens / instruções ao servidor, sendo posteriormente respondidas por este.

A figura 3 ilustra o serviço *web,* implementado pelo autor, que é responsável pelo controlo da direção do respetivo carro.



Figura 3 - Serviço web implementado pelo autor.

A placa ESP32 atua como servidor *websocket* e o *browser* do *smartphone* atua como cliente *websocket.* As mensagens do tipo WebSockets, provenientes do *browser*, são processadas pela placa referida. Concluído o seu processamento, a placa envia mensagens de controlo para o motor LM298, com o intuito de este controlar a rotação das rodas do carro. Para além do controlo da rotação das rodas, também é possível variar a velocidade das mesmas através da modulação por largura de pulso.

Sempre que um utilizador clica nos botões de *up, down. left* ou *right,* o *smartphone* envia uma mensagem do tipo *websocket* para o robot, que por sua vez será processada pela respetiva placa.

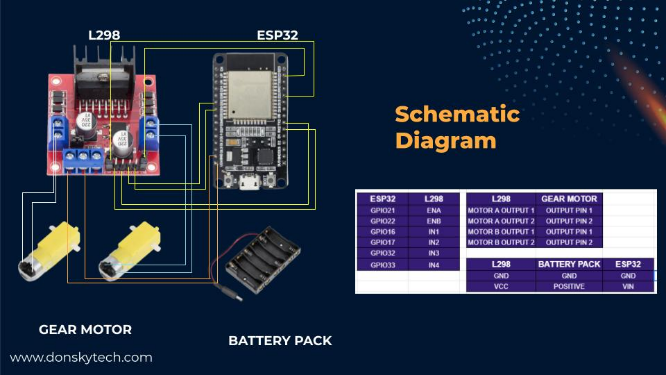
A figura 4 apresenta a listra de componentes que são necessários ao desenvolvimento deste projeto.

Figura 4 - Lista de componentes necessários à construção do projeto.

# Etapas do trabalho desenvolvido

Esta secção contém todas as etapas necessárias ao desenvolvimento desta fase, que se encontram descritas pelos mecanismos, ferramentas utilizadas e algoritmos implementados pelo grupo.

## 3.1. Protocolo de comunicação

Para estabelecer a comunicação entre o Sistema Central e o *Gateway* recorreu-se ao protocolo TCP/IP, de modo a transmitir as tramas de dados recolhidas.

Para tal, o grupo declarou *sockets* do tipo servidor no Sistema Central e *sockets* do tipo cliente no *Gateway*. A partir destas, tirando partido da programação concorrente, foi estabelecida a referida ligação, que permite o bom funcionamento das interações necessárias.

Numa fase mais avançada do projeto, as tramas não irão passar só os valores obtidos pelo Sistema Sensor (e as amostras temporais), como também transportarão informação relativa ao atual funcionamento dos sensores e incluirão ordens de começo, paragem ou retoma da atividade.

A figura 5 apresenta os parâmetros necessários à implementação das *sockets* referidas e o objeto responsável pelo envio da trama e pelo estabelecimento da ligação com o Sistema Central (dentro do código do *Gateway*).

Graphical user interface, text

Description automatically generated with medium confidence

Figura 5 - Declaração da porta, do endereço e do cliente usados para a comunicação com o Sistema Central.

A figura 6 ilustra as mensagens de diagnóstico, relativas à conexão com o Sistema Central, e o envio da amostra para o mesmo.

Graphical user interface, text, application

Description automatically generated

Figura 6 - Envio da trama de dados e finalização da impressão na socket, no Gateway.

A figura 7 ilustra o código do Sistema Central, que é responsável pela inicialização e execução do servidor.

Text

Description automatically generated

Figura 7 - Inicialização da socket servidor no sistema.

## 3.2. Sistema Central

Nas subsecções seguintes serão analisadas e descritas todas as caraterísticas, funcionalidades e matérias inerentes ao Sistema Central, que foram implementadas pelo grupo.

### 3.2.1 Mecanismos e políticas de segurança

Os mecanismos de autenticação, autorização e outros recursos de segurança foram elaborados com o apoio da *framework Spring Security* [3]*.* As políticas e estratégias de segurança adotadas pelo grupo, encontram-se organizadas na diretoria “*security*”.

A classe *WeatherSample* define a estrutura, representa o paradigma de qualquer amostra recolhida, isto é, contém todos os atributos e comportamentos inerentes à mesma.

A figura 12 ilustra os estados (variáveis de instância) e os comportamentos (métodos) relativos à classe *WeatherSample*.

Uma imagem com mesa

Descrição gerada automaticamente

Figura 12 - Estrutura da classe WeatherSample.

É de salientar que dentro da base de dados, cada amostra é identificada pelo seu código identificador (ID), que corresponde à chave primária da tabela *weather*.

Uma imagem com texto

Descrição gerada automaticamenteA figura seguinte ilustra os atributos da tabela, entidade *weather*.

Figura 13 - Atributos da tabela weather.

As classes *WeatherSampleController*, *WeatherSampleService* e a interface *WeatherSampleRepo* atuam em diferentes camadas da aplicação.

A interface *WeatherSampleRepo* atua na camada de acesso à base de dados, ou seja, é responsável pela manipulação e gerenciamento da base de dados relacional. Esta interface estende a interface *JpaRepository*¸ que recebe duas classes como argumentos: a classe *WeatherSample*, que define os atributos da tabela *weather* e a classe *Long*, que representa o tipo da chave primária da tabela referida.

É de realçar que não foram definidos quaisquer métodos para a interface *WeatherSampleRepo*, uma vez que os métodos herdados, são suficientes para a complexidade deste projeto.

Relativamente à classe *WeatherSampleService*, esta estabelece a ponte de interligação entre a camada de serviço e a camada de acesso à base de dados, isto é, define um conjunto de métodos que operam sobre os métodos herdados pela interface *WeatherSampleRepo*.

Uma imagem com texto

Descrição gerada automaticamenteA figura 14 ilustra os estados (variáveis de instância) e os comportamentos (métodos) relativos à classe *WeatherSampleService*.

Figura 14 - Estrutura da classe WeatherSampleService.

No caso da classe *WeatherSampleController*, que equivale à camada de controlo, esta efetua um conjunto de *web requests* ao serviço web, do tipo: *GET*, *POST*, *PUT* e *DELETE*. Cada *web request* resulta numa *query* colocada à base de dados.

Em semelhança com o que acontece com a classe *WeatherSampleService*, a camada de controlo também define um conjunto de métodos, anotados com os tipos de *web requests* associados, que operam sobre os métodos definidos pela classe, que é responsável pela camada de serviço.

Uma imagem com texto

Descrição gerada automaticamenteA figura 15 ilustra os estados (variáveis de instância) e os comportamentos (métodos) relativos à classe *WeatherSampleController*.

Figura 15 - Estrutura da classe WeatherSampleService.

Uma imagem com texto

Descrição gerada automaticamenteA interação entre as diversas camadas do gestor de serviço, que foram descritas e analisadas ao longo desta subsecção, pode ser ilustrada de acordo com a figura 16.

Figura 16 - Interações entre as camadas do gestor de serviço.

### 3.2.2 Security

A diretoria *security* [3] contém duas classes e dois enumerados, as classes qualificam e determinam a segurança do serviço *web*, enquanto, que os enumerados definem o tipo de utilizadores e respetivas habilidades ou permissões associadas*.*

A classe *ApplicationSecurityConfig* configura, filtra o protocolo HTTP (*Hypertext Transfer Protocol*), com recurso ao método *configure*, e regista o conjunto de utilizadores, que podem comunicar e interagir com o serviço *web*, através do método *userDetailsService*.

A figura 17 ilustra os estados (variáveis de instância) e os comportamentos (métodos) relativos à classe *ApplicationSecurityConfig*.

Uma imagem com texto

Descrição gerada automaticamente

Figura 17 - Estrutura da classe ApplicationSecurityConfig.

Cada utilizador, que se encontra registado na aplicação, é caraterizado pelo seu *username* e por uma *password*, que por sua vez é encriptada, de forma a reforçar a segurança do serviço *web*.

### 3.2.3 Template

A diretoria *template* contém apenas uma classe que possui dois métodos, que retornam as páginas *web* (***get*** *requests*), que foram previamente definidas na diretoria *resources*.

A figura 18 apresenta os estados (variáveis de instância) e os comportamentos (métodos) relativos à classe *TemplateController*.

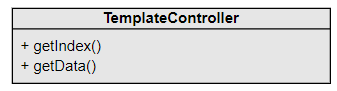


Figura 18 - Estrutura da classe TemplateController.

### 3.2.4 Resources

A diretoria *resources* abrange código relativo à componente gráfica da aplicação, isto é, o serviço *web* e possui o ficheiro de propriedades ou configurações referentes à base de dados relacional, como mostra a figura 19.

Uma imagem com texto, captura de ecrã, ecrã, preto

Descrição gerada automaticamente

Figura 19 - Ficheiro com as configurações da base de dados.

## 3.3. Transmissão de assinaturas temporais por BLE

De maneira a obtermos o *timestamp* no servidor recorreu-se às capacidades do protocolo BLE, criando uma característica na qual será escrito o valor em segundos desde 1 de janeiro de 1970 às 00:00:00, como mostra a figura 20.

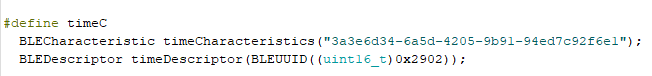


Figura 20 - Declaração da característica temporal no servidor.

De seguida, este valor é escrito na referida característica, como demonstra a figura 21.

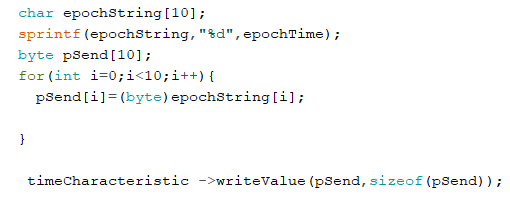


Figura 21- População do array a ser enviado pela característica.

Assim sendo, o servidor lê os valores nestes impressos, evidenciada na figura 22.

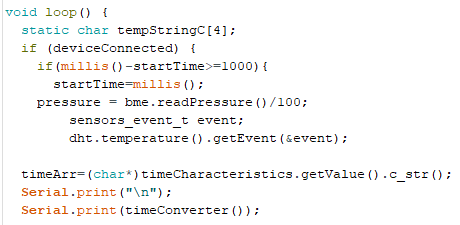


Figura 22 - Leitura do valor temporal.

Tendo obtido o mencionado valor são realizadas um conjunto de operações de modo a converter o tempo em segundos numa *timestamp* com o formato: dia da semana, ano, mês, dia e hh(horas): mm(minutos): ss(segundos), como mostra a figura 23.

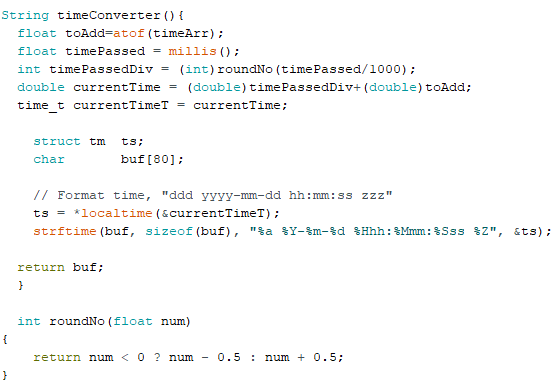


Figura 23 - Funções de conversão da data para o formato necessário.

É de notar que optamos por calcular o tempo atual adicionando o tempo em segundos desde a inicialização do Arduíno, visto nos termos deparado com faltas de fiabilidade relativamente a obtermos um valor temporal novo, do cliente, a cada segundo (tempos de execução e paragens no momento da abertura de *sockets*, podiam levar ao aumento do tempo de forma não ideal).

## 3.4. Servidor *Web*

Para esta etapa, o servidor *Web* que estamos a apresentar é uma versão mais atualizada da que foi apresentada na fase B.

Uma modificação visual importante foi a substituição do submenu *Feeds* porum novo intitulado *Historical Data*, cuja função é a de adicionar uma dimensão gráfica à apresentação dos dados obtidos, com gráficos que mostram a alteração da temperatura, humidade e pressão atmosférica ao longo do tempo. Outra modificação visual importante é a adição do controlo, por parte de um utilizador com capacidades de administração, dos *gateways*, podendo estes assim fazer alterações ao estado de cada *gateway.*

Como já vimos anteriormente, também adicionamos ao nosso *Servidor Web* um protocolo de *login*, sendo essa a primeira página onde cada utilizador vai parar.

Caso o utilizador não seja administrador, este apenas terá acesso a uma página com uma tabela, que apresenta, em tempo real, o ID da amostra, o ID do *gateway* de onde origina a amostra, tal como a temperatura, humidade e pressão atmosférica recolhida pelos sensores, juntamente com o *timestamp*, como mostra a figura 25.



Figura 24 - Página com todas as amostras.

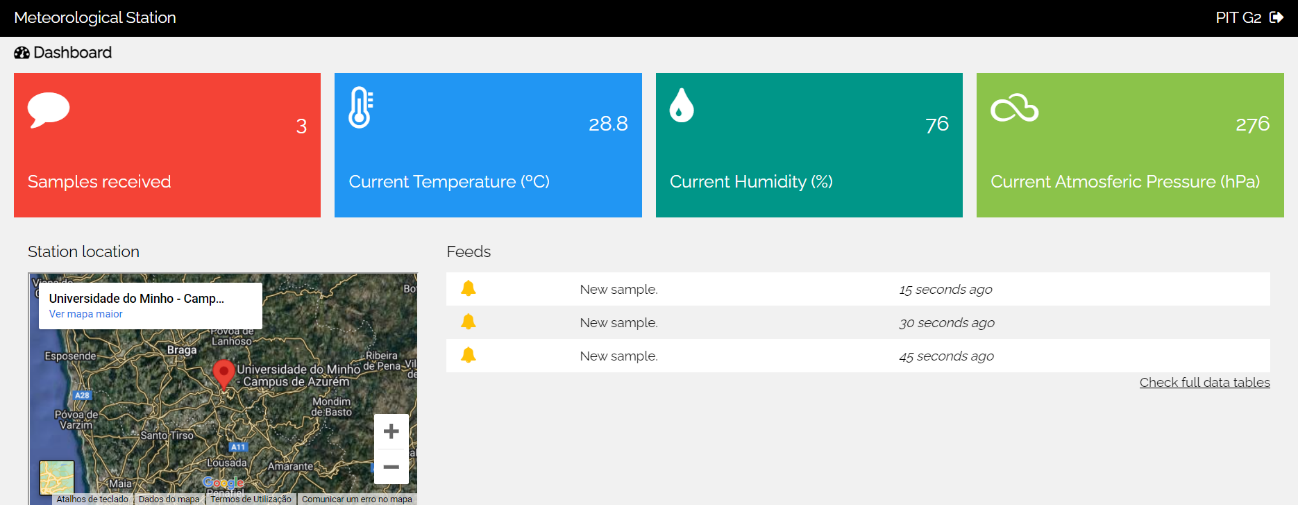


Figura 25 - Página inicial do servidor web.

# Análise de resultados e testes efetuados

De forma a retificar o funcionamento e o comportamento do Sistema Central e do protocolo de comunicação foram retiradas várias amostras, que evidenciam o funcionamento dos protocolos e mecanismos utilizados.

A figura 27 ilustra um conjunto de 10 amostras enviadas pelo Gateway para o Sistema Central, sendo armazenadas na base de dados pelo mesmo.

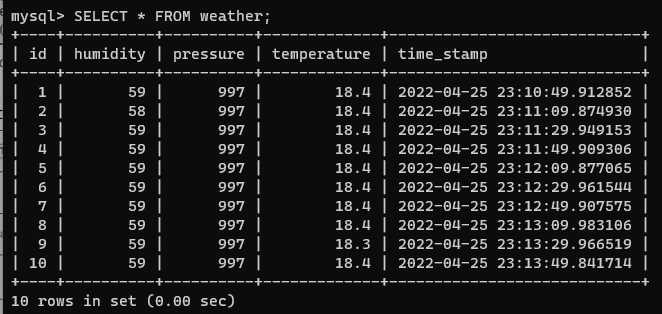


Figura 26 - Amostras armazenadas na base de dados.

A figura 28 ilustra o conjunto de 10 amostras referido, colocadas no serviço *web*.

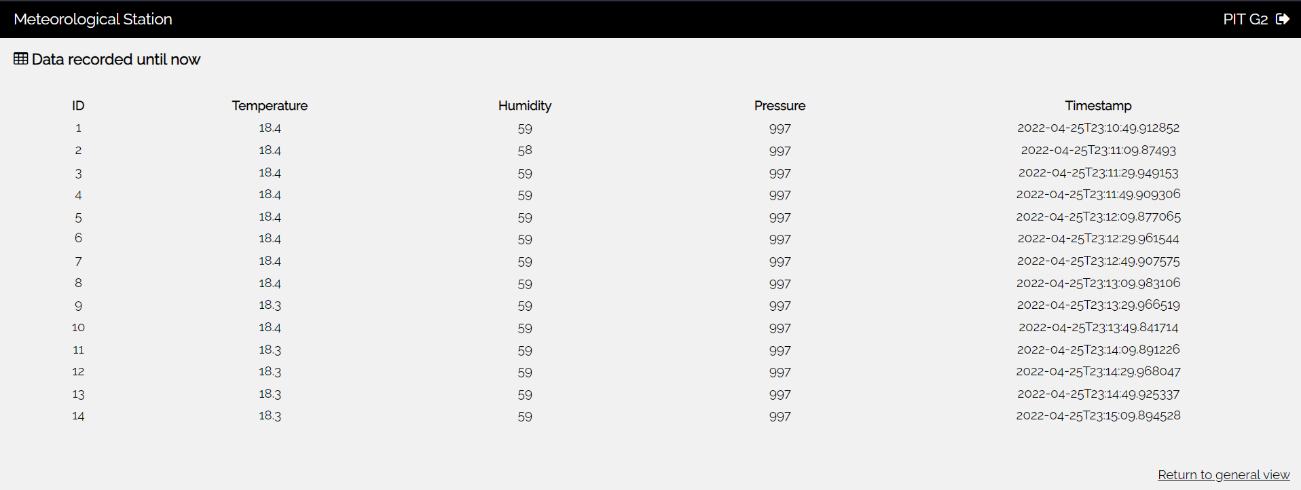


Figura 27 - Amostras impressas no serviço web.

Para se estabelecer um fio de ligação entre a base de dados e o serviço web, o grupo construiu um *script* que se encontra apresentado na figura 29.

Uma imagem com texto

Descrição gerada automaticamente

Figura 28 - Script que efetua o parsing das amostras.

Relativamente ao domínio dos sistemas simulados, o grupo definiu um conjunto de amostras *hardcoded* e armazenou-as na base de dados, com o intuito de testar as funcionalidades do repositório e a ligação com o serviço *web*.

A figura 30 ilustra os valores *hardcoded* definidos pelo grupo.

Uma imagem com texto

Descrição gerada automaticamente

Figura 29 - Definição das amostras hardcoded.

A figura 31 ilustra as amostras armazenadas na base de dados, que por sua vez encontram-se impressas no serviço *web*.

Uma imagem com texto

Descrição gerada automaticamente

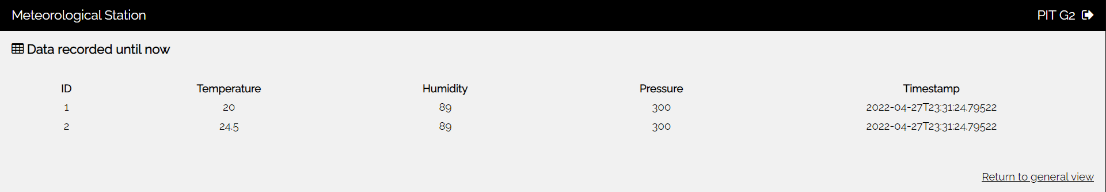


Figura 30 - Armazenamento e impressão das amostras hardcoded.

# Conclusão

Concluída esta fase, na opinião do grupo, conseguimos atingir todos os objetivos propostos. Esta fase contribuiu para o conhecimento e estudo da implementação de *Sockets*, comunicação sobre TCP/IP e sobre o desenho de Servidores *Web*, tal como comunicação bilateral utilizando o protocolo BLE.

O grupo pretende que, com o trabalho desenvolvido ao longo deste fase intermédia, ter boas fundações para completar a etapa final deste projeto com sucesso.

## 5.1. Contribuição de cada aluno

Catarina Neves

* Conceção dos algoritmos;
* Tratamento de dados, tabelas e imagens a estes relacionados;
* Transmissão de *timestamps,* sua codificação e descodificação;
* Programação e implementação da aplicação dos Sistema Sensor e *gateway*;
* Desenvolvimento do relatório;

Eduardo Cardoso

* Conceção dos algoritmos;
* Transmissão de *timestamps,* sua codificação e descodificação;
* Programação e implementação da aplicação dos Sistema Sensor e *gateway*;
* Desenvolvimento e implementação das *sockets* e comunicação TCP/IP;
* Desenvolvimento do relatório;

José Gomes

* Conceção dos algoritmos;
* Tradução das classes em diagramas;
* Implementação do Sistema Central;
* Desenvolvimento e implementação das *sockets* e comunicação TCP/IP;
* Desenvolvimento do relatório;

Luís Oliveira

* Conceção dos algoritmos;
* Desenvolvimento do *Frontend* e *Backend* do Servidor *Web*;
* Implementação do Sistema Central;
* Desenho de esquemas e gráficos;
* Desenvolvimento do relatório.

# Lista de referências

|  |  |
| --- | --- |
| [1] | SWA, “ESP32 Websocket: Control ESP32 Pins [PlatformIO],” 4 2021. [Online]. Available: https://www.survivingwithandroid.com/esp32-websocket-control-esp32-pins-platformio/. |
| [2] | donsky, “ESP32 Robot Car Using Websockets,” julho 2021. [Online]. Available: https://www.donskytech.com/esp32-robot-car-using-websockets/. |
| [3] | VMware, “Spring Security,” 2022. [Online]. Available: https://spring.io/projects/spring-security?msclkid=03ed647cc53c11ec824b0986c6465ca1. |
| [4] | NTP Pool Project, “Europe — europe.pool.ntp.org,” [Online]. Available: https://www.pool.ntp.org/zone/europe. |
| [5] | R. Santos, “Random Nerd Tutorials,” 10 2020. [Online]. Available: https://randomnerdtutorials.com/esp32-websocket-server-arduino/. |
| [6] | W3Schools, “W3Schools W3.CSS,” [Online]. Available: https://www.w3schools.com/w3css/. |