

# **Relatório de trabalho de projeto**

## **My Garden**

Trabalho de Projeto da unidade curricular de Tecnologias de Internet  
Engenharia Informática

Ivo Afonso Freire Bispo

José António Portela Areia

Leiria, junho de 2021

**Título do Documento** – Relatório de Projeto | Tecnologias de Internet

**Unidade Curricular** – Tecnologias de Internet

**Curso** – Engenharia Informática

**Instituição de Ensino** – Politécnico de Leiria | Escola Superior de Tecnologia e Gestão

**Ano Letivo** – 2020/2021

**Estudante 1** – Ivo Afonso Freire Bispo | 2200672 | 2200672@my.ipleiria.pt

**Estudante 2** – José António Portela Areia | 2200655 | 2200655@my.ipleiria.pt

**Data de Entrega** – junho de 2021

**Local** – Escola Superior de Tecnologia e Gestão - Leiria

## RESUMO

O presente relatório serve para demonstrar todo o trabalho realizado e o envolvimento que nós, Ivo Afonso Freire Bispo e José António Portela Areia, alunos do primeiro ano de Engenharia Informática tivemos para com o projeto intitulado “*My Garden*” no âmbito da unidade curricular Tecnologias de Internet.

O mesmo, pode ser dividido em quatro grandes frações, que irão ser apresentadas de uma forma exaustiva no decorrer deste mesmo relatório.

É importante sublinhar, que os conhecimentos outrora aprendidos e desenvolvidos na unidade curricular foram utilizados no decorrer do trabalho de projeto, o que têm como significado, um aprofundamento e melhoramento dos mesmos.

**Palavras-chave:** cisco packet tracer, web, laravel, sistemas de redes informáticos.

## ÍNDICE

<b>Resumo .....</b>	<b>II</b>
<b>Índice .....</b>	<b>III</b>
<b>Índice de Figuras .....</b>	<b>IV</b>
<b>Índice de Tabelas .....</b>	<b>V</b>
<b>Glossário .....</b>	<b>VI</b>
<b>Acrónimos .....</b>	<b>VII</b>
<b>Introdução .....</b>	<b>8</b>
<b>1. Arquitetura .....</b>	<b>9</b>
<b>2. Implementação.....</b>	<b>11</b>
2.1. Condições CPT .....	11
2.2. Condições Dashboard .....	14
<b>3. Cenários de teste .....</b>	<b>15</b>
<b>4. Resultados obtidos .....</b>	<b>19</b>
<b>Conclusão .....</b>	<b>20</b>
<b>Referências Bibliográficas .....</b>	<b>21</b>
<b>Anexos.....</b>	<b>22</b>

## ÍNDICE DE FIGURAS

Figura 1 – Exterior da estufa .....	9
Figura 2 – Interior da estufa .....	9
Figura 3 - <i>Core</i> de funcionamento.....	10
Figura 4 - Exemplo do funcionamento de uma comunicação, apenas com CPT .....	10
Figura 5 - Apresentação de uma tabela da base de dados através do <i>phpMyAdmin</i> .....	18

## ÍNDICE DE TABELAS

Tabela 1 - Condições e resultados dos sensores e/ou atuadores nas culturas.....	11
Tabela 2 - Condições e resultados da abertura da estufa.....	12
Tabela 3 - Condições e resultados para a gestão da temperatura .....	12
Tabela 4 - Condições e resultados na gestão da <i>server room</i> .....	13
Tabela 5 - Condições e resultados nas ações da <i>dashboard</i> .....	14

## GLOSSÁRIO

**Web** - Sistema de documentos em hipermídia que são interligados e executados na internet.

**Laravel** – Framework da linguagem de programação PHP.

**Query** – Pedido de informação a base de dados que carece de uma resposta.

## ACRÓNIMOS

**AJAX** - Asynchronous JavaScript And XML.

**CPT** – Cisco Packet Tracer, programa utilizado para a criação de sistemas de redes virtuais.

**JS** – JavaScript, linguagem de programação.

**MVC** – Model, View and Controller.

**PHP** - Acrônimo recursivo para "PHP: Hypertext Preprocessor", originalmente Personal Home Page) é uma linguagem interpretada livre, usada para o desenvolvimento de aplicações presentes no lado do servidor.

**UC** - Unidade Curricular.



## INTRODUÇÃO

Este relatório foi desenvolvido no âmbito do trabalho de projeto da unidade curricular Tecnologias de Internet lecionada no curso Engenharia Informática no seu primeiro ano. Sublinha-se ainda que o mesmo foi trabalho e desenvolvido em toda a sua extensão pelos elementos Ivo Afonso Freire Bispo e José António Portela Areia identificados pelos números 2200672 e 2200655, respetivamente.

O nosso objetivo com este trabalho é revolucionar a indústria da agricultura, para algo que pode ser portátil, autónomo e eficiente. Escolhemos, portanto, desenvolver este tema/projeto com um ideal bastante simples: revolucionar aquilo a que se conhece como “estufa”.

Criámos assim algo que pode ser sustentável/portátil. Contentores podem ser utilizados para a estrutura. Autónomo, todas as plantações são mantidas e desenvolvidas automaticamente sem qualquer “toque” humano; e eficiente, pois é possível criar qualquer plantação mesmo não sendo a estação do ano correta, mas tendo todos os dados necessários como luminosidade, temperatura entre outros, podem ser criadas essas culturas.

Existem hipóteses a testar com este projeto, umas podem ser lunáticos outras podem ser humanísticas. As lunáticas têm como objetivo enviar estes contentores para Marte, enquanto os humanistas é enviar para os países pobres para terem alimentação em qualquer época do ano, por fim pudemos sempre colocar contentores uns em cima de outros criando assim a ideia de um “prédio” poupando em componentes da sala de servidor.

Utilizámos várias técnicas assim como explorámos novas tecnologias, deixando de utilizar PHP na sua forma mais simples, para utilizar a *framework* Laravel, os ficheiros passaram a ser uma Base de Dados e as novas funcionalidades foram o Cisco Packet Tracer, utilizando JavaScript e por fim Python.

## 1. ARQUITETURA

Como introduzido anteriormente a nossa ideia a desenvolver é uma “estufa” automática, para esse desenvolvimento são necessários vários equipamentos, sensores, atuadores, entre outros...

Para o desenvolvimento da parte das plantações vão ser necessários sensores de temperatura, humidade, vento e luminosidade, todos ligados a um Microcontrolador (MCU) e na parte dos atuadores são uma janela, uma lâmpada, rega e LCD, ligados a um Microcontrolador SBC, assim como alguns destes também vão estar ligados ao MCU. No que toca à medição dos valores atmosféricos genéricos apenas são necessários um sensor de humidade e temperatura, ligados a um Microcontrolador.

No que toca à segurança e funcionamento da sala de servidores já são necessários um sistema de entrada utilizando um leitor de cartões ligado a um MCU, uma porta para a sala de servidor com sistema de segurança de reconhecimento facial e dentro da sala de servidor um sistema anti-incêndio. Para a sala de servidores funcionar corretamente são necessários um servidor, *switch*, *accesspoint* e por fim um portátil para realizar testes.

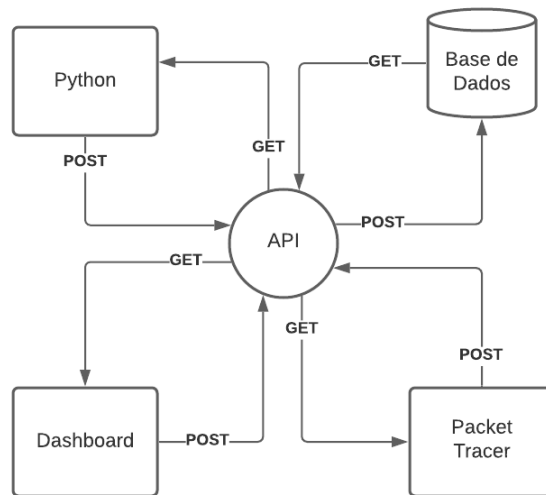
Em software houve a mudança de ficheiro para Base de Dados, de PHP puro para a *framework* Laravel e a utilização de Python, salientamos que todas as comunicações realizadas com a BD são sempre pela API.



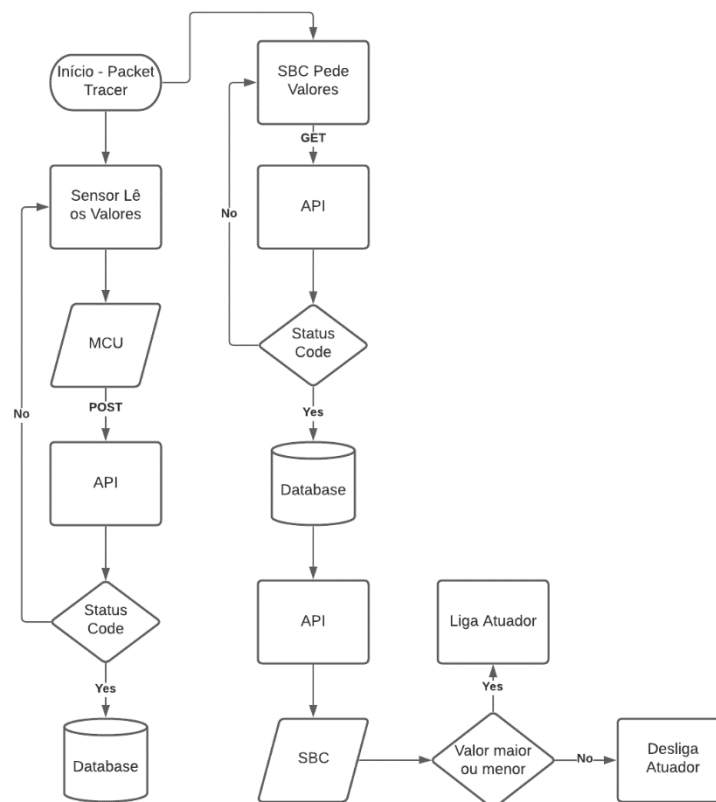
**Figura 1 – Exterior da estufa**



**Figura 2 – Interior da estufa**



**Figura 3 - Core de funcionamento**



**Figura 4 - Exemplo do funcionamento de uma comunicação, apenas com CPT**

## 2. IMPLEMENTAÇÃO

Este capítulo visa a representar todo o ideal que está por de trás do projeto. Para facilitar a sua interpretação irá ser apresentado abaixo algumas tabelas, divididas por secções, que contêm as condições e os consequentes resultados de toda a implementação de sensores e/ou atuadores projeto presente.

### 2.1. Condições CPT

A seguinte tabela representa todas as condições e os consequentes resultados para as secções das culturas. Sublinha-se que, por omissão, todos os resultados abaixo apresentados contêm o seguinte valor “**Escreve no LCD**”, este apenas não é apresentado, visto a quantidade de repetições que se iria proceder.

Condição	Resultado
Se o valor da humidade for menor que 20%	Liga a rega
Se o valor da humidade for maior que 20%	Desliga a rega
Se o valor da luminosidade for menor que 30%	Liga a luz
Se o valor da luminosidade for maior que 30%	Desliga a luz
Se a velocidade do vento for maior que 10 km/h	Fecha a janela
Se a velocidade do vento for menor que 10 km/h	Abre a janela

**Tabela 1 - Condições e resultados dos sensores e/ou atuadores nas culturas**

A tabela abaixo representa a simples lógica que se encontra no processo de abertura da estufa.

<b>Condição</b>	<b>Resultado</b>
Se o código do cartão RFID for igual a “1001”	Abre a porta
Se o código do cartão RFID for igual a “1002”	Abre a porta
Se o código do cartão RFID for diferente dos acima mencionados	Nada acontece
Se a porta for aberta pelo cartão RFID	Envia um pedido POST

**Tabela 2 - Condições e resultados da abertura da estufa**

De seguida irá ser apresentada a lógica de condição e resultado para o funcionamento automático do aquecimento e/ou arrefecimento da estufa através de um termostato e de um ar condicionado.

<b>Condição</b>	<b>Resultado</b>
Se o valor da temperatura for maior que 20°C	A/C aciona o arrefecimento
Se o valor da temperatura for menor que 10°C	A/C aciona o aquecimento
Se a temperatura estiver no intervalo entre 10°C e 20°C	Nada acontece

**Tabela 3 - Condições e resultados para a gestão da temperatura**

Por último será apresentada a tabela que contém as condições e resultados provenientes dos sensores e/ou atuadores da *server room*.

Condição	Resultado
Se for reconhecida uma face pelo script Python	Abre a porta
Se não for reconhecida uma face pelo script Python	Nada acontece
Se se passar pelo sensor de movimento	Liga a luz
Se o valor da temperatura for maior que 20°C	Liga a ventoinha no máximo
Se o valor da temperatura for maior que 15°C	Liga a ventoinha no mínimo
Se o valor da temperatura for menor que 15°C	Nada acontece
Se o sensor de incêndio detetar fogo	Liga o aspersor de água e ativa a sirene
Se a humidade for menor que 40%	Liga o humificador
Se a humidade for maior que 40%	Desliga o humificador

**Tabela 4 - Condições e resultados na gestão da *server room***

## 2.2. Condições Dashboard

Com esta tabela, termina-se assim toda a lógica da implementação dos sensores e atuadores provenientes da aplicação CPT. Falta apenas referir a mudança de valores dos atuadores, que é passível de ser feita através da *dashboard* com auxílio de pedidos assíncronos através da ferramenta AJAX.

Condição	Resultado
Ao clicar no estado da luz, para esta ser desligada, e se o valor da luminosidade for maior que 30%	Desliga a luz
Ao clicar no estado da luz, para esta ser desligada, e se o valor da luminosidade for menor que 30%	Retorna aviso de impossibilidade da ação
Ao clicar no estado da luz para esta ser ligada	Liga a luz
Ao clicar no estado da rega, para esta ser desligada, e se o valor da humidade for maior que 20%	Desliga a rega
Ao clicar no estado da rega, para esta ser desligada, e se o valor da humidade for menor que 20%	Retorna aviso de impossibilidade da ação
Ao clicar no estado da rega para esta ser ligada	Liga a rega
Ao clicar no estado da janela, para esta ser aberta, e se o valor da velocidade do vento for menor que 10 km/h	Abre a janela
Ao clicar no estado da janela, para esta ser aberta, e se o valor da velocidade do vento for maior que 10 km/h	Retorna aviso de impossibilidade da ação
Ao clicar no estado da janela para esta ser fechada	Fecha a janela

**Tabela 5 - Condições e resultados nas ações da *dashboard***

### 3. CENÁRIOS DE TESTE

Este projeto foi realizado com auxílio de uma ferramenta MVC, sendo a mesma Laravel (*framework* de PHP), uma base de dados, em substituição dos ficheiros “.txt”, Python, JavaScript, AJAX e o programa Cisco Packet Tracer.

No Cisco Packet Tracer, o contentor foi separado em quatro áreas distintas, sendo as mesmas, a área das plantações contendo três culturas (alfaces, cenouras e tomates) e a *server room*.

Na entrada do contentor deparamo-nos com um sistema de segurança de leitura de cartões, utilizando um RFID *Reader* e dois cartões RFID *Card* cada um com o seu código de autorização. Ao passarmos o cartão pelo *Reader* o mesmo vai validar se o código do cartão é o correto enviando para o MCU-PT e abrindo assim a porta, ao abrir a porta vai ser realizado um POST para a API com os valores de quem entrou ou saiu, guardando na base de dados essa mesma informação. Na *dashboard* pudemos ir a página da “Histórico de entradas” e dependendo da conta que temos a sessão iniciada pudemos ver as nossas entradas, assim como a fotografia da última pessoa a entrar na *server room*, assunto ao qual será falado posteriormente.

Ao entrarmos, podemos observar as plantações, que neste exemplo têm o mesmo tipo de funcionamento, constituído por: três LCDs, SBC-PT, *Lawn Sprinkler*, *Light*, *Window*, MCU-PT, *Temperature Sensor*, *Humidity Sensor*, *Wind Sensor* e *Photo Sensor* (estas componentes são as mesmas para as outras plantações). O leitor MCU está ligado aos quatro sensores, temperatura e humidade às portas analógicas e o de vento e luminosidade às portas digitais, assim como a três atuadores: janela, lâmpada e rega nas portas digitais. Para a leitura dos sensores de vento e luminosidade foi necessário fazer a alteração do mesmo para permitir a leitura dos seus valores. O SBC está ligado a seis atuadores pelas portas digitais: os três LCDs, a rega, a lâmpada e a janela.

O MCU têm como funcionamento geral receber os valores ou estados dos sensores e atuadores e enviá-los para API. Aprofundando mais a explicação do seu funcionamento, o microcontrolador lê todos os valores dos sensores e atuadores, agrupando-os num *array* do tipo dicionário para de seguida enviá-lo para a API pelo método POST, estes valores vão ser validados pela API através do seu controlador, se algum dos valores for incorreto



irá apresentar uma mensagem de erro ao utilizador, caso contrário irá apresentar uma mensagem de sucesso e vai ser criada uma nova linha na tabela com os valores enviados. Pode-se, conseqüentemente, observar estes valores na *dashboard* da aplicação.

Na *dashboard* é ainda possível controlar o funcionamento dos atuadores do CPT utilizando AJAX, como exemplo podemos utilizar o atuador da rega, ao clicar no mesmo é possível desligar a rega se a mesma se encontrar ligada ou vice-versa, sem a necessidade de fazer *refresh* a página; se formos à base de dados também podemos observar que esse valor foi alterado.

Voltando às plantações temos o SBC-PT que tem como funcionamento principal colocar os atuadores em funcionamento, realizando um pedido GET à API, de seguida separa esses valores que vêm em modo *string* para um array dicionário. Os valores recebidos vão ser avaliados pelo controlador para saber se deve ligar a luz, a rega ou até mesmo abrir a janela. Estes valores são todos apresentados nos três LCDs para quando o utilizador entrar, conseguir ler logo se os atuadores se encontram ligados ou não.

A *server room* contém todos os equipamentos necessários para a ligação com o exterior, como: um Server-PT (IOT), 2960-24TT (switch), AccessPoint-PT e um portátil. Devido à importância desta sala, foi criado todo um sistema de anti-incêndio. O sistema anti-incêndio é controlado por um MCU, que está ligado a quatro equipamentos de entrada digital e uma analógica, sendo as digitais: uma *Fan*, um *Fire Sprink*, um *Fire Monitor* e uma *Siren* e na porta analógica está ligado o *Temperature Sensor*. O sensor de temperatura está a fazer uma comunicação constante com o MCU, enviando os valores e dependendo dos valores lidos o microcontrolador liga a ventoinha. O sistema de incêndio funciona com o mesmo processo, se o sensor de incêndio detetar o início do mesmo, o extintor de incêndio irá se ativar assim como a sirene (todo este funcionamento é realizado dentro de CPT).

Para entrarmos na *server room* encontramos uma porta e para a mesma se abrir é necessário realizar o nosso reconhecimento facial. Este reconhecimento é feito através de Python, em que o mesmo faz a captura do rosto e compara com as imagens das faces válidas, se o rosto da pessoa for igual à das faces válidas o mesmo vai fazer um pedido POST para alterar o estado da porta na base de dados. O MCU que está ligado à porta está à escuta do pedido GET desse valor para abrir a porta, passados cinco segundos o MCU fecha a porta e faz um envio POST para o servidor para indicar que a porta foi fechada, de seguida o Python realiza um pedido GET para saber se a porta foi fechada, apresentando de seguida em mensagem que a porta foi fechada.

Ainda no CPT existem equipamentos que servem para o funcionamento normal dos contentores, como um sistema de ar-condicionado que contém: um sensor de temperatura, um MCU, um termostato e o ar-condicionado, em que o sensor de temperatura vai enviar os valores lidos para o MCU e o mesmo verifica se a temperatura se encontra demasiado baixa ou alta, para de seguida enviar para o termostato e se os mesmo se encontrar debaixo da temperatura escolhida, liga o quente caso contrário liga o frio. Os equipamentos que funcionam com condições colocadas no servidor IoT (MyGarden), são a ventoinha que quando se encontra numa temperatura amena liga no estado LOW e quando está quente liga no HIGH, o umidificador que têm o mesmo funcionamento caso a humidade esteja baixa liga-se e por fim o sistema de entrada na sala de servidor, em que tem um detetor de movimento e um lâmpada que se encontra sempre com fraca luz, mas no caso se movimento acende-se no máximo.

Por fim no CPT existem os sensores gerais que têm como principal funcionalidade fazer envios POST para a API, de como se encontra a temperatura e humidade exterior, estes dados são apresentados na *dashboard*. No funcionamento do CPT foram colocados mais atuadores e sensores do que os previsto assim como funcionalidades explicadas anteriormente.

Na parte de Laravel existem duas rotas principais a *web.php* que contém apenas rotas que retornam *views* sem qualquer informação adiciona, como exemplo as páginas de históricos dos componentes, no entanto todo o funcionamento, pedidos ou informação que aparece nas páginas são todas realizadas através da API, assim respeitando todos os parâmetros e requisitos pedidos.

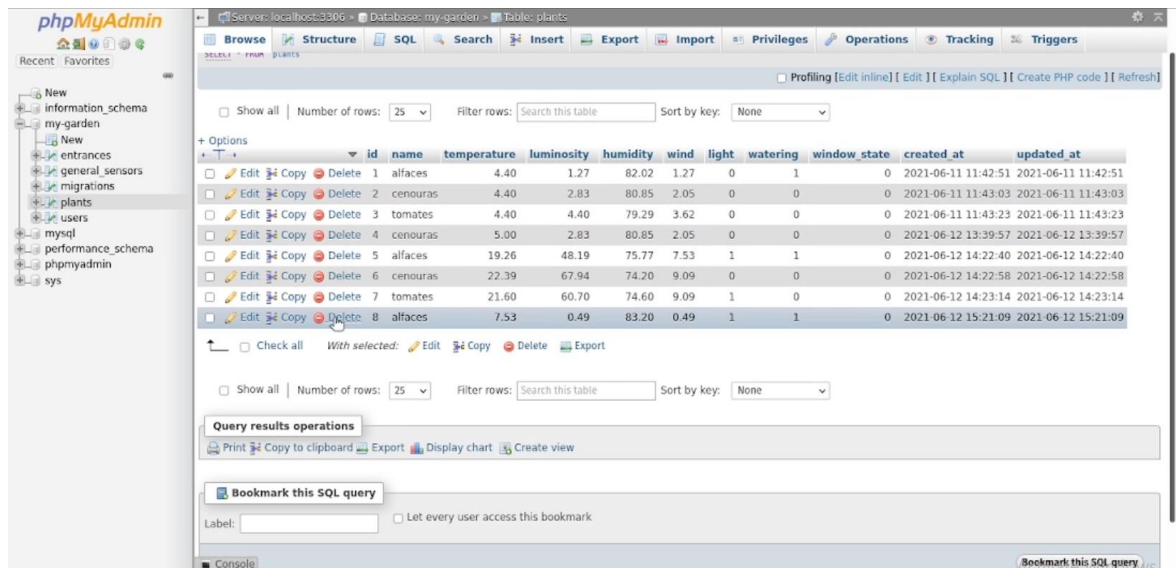
Na página web, no menu lateral encontramos uma seleção de páginas de histórico que podemos observar, como o da temperatura, humidade, sensor específico ou cultura, ao escolhermos um, exemplo o da temperatura, apresenta uma tabela com os valores da BD, estes valores são pedidos à API em AJAX, assim como também um gráfico com as últimas dez entradas, para uma visualização mais gráfica e simples.

Para finalizar todas as funcionalidades desenvolvidas, foi realizado um script em Python que serve para observar, assim como alterar qualquer valor dos sensores na *dashboard*, cortando assim o uso de ir à BD alterar valores ou quando queremos algum valor específico para testar no CPT sem ser os valores medidos pelos sensores, este script faz pedidos GET e POST à API.

Finalizamos assim as funcionalidades desenvolvimentos, mencionando que todas as funcionalidades propostas foram desenvolvidas assim como funcionalidade opcionais

como: o uso de AJAX, uma base de dados, um sistema MVC (Laravel), gráficos para a apresentação dos valores, uma biblioteca externa de python (face\_recognition), modificamos o uso dos sensores e por fim utilizamos mais sensores e atuadores do que os pedidos.

É importante referir num último ponto que é possível ler e visualizar alguns dos códigos acima mencionados nos anexos junto ao presente relatório.



	id	name	temperature	luminosity	humidity	wind	light	watering	window_state	created_at	updated_at
<input type="checkbox"/>	1	alfices	4.40	1.27	82.02	1.27	0	1	0	2021-06-11 11:42:51	2021-06-11 11:42:51
<input type="checkbox"/>	2	cenouras	4.40	2.83	80.85	2.05	0	0	0	2021-06-11 11:43:03	2021-06-11 11:43:03
<input type="checkbox"/>	3	tomates	4.40	4.40	79.29	3.62	0	0	0	2021-06-11 11:43:23	2021-06-11 11:43:23
<input type="checkbox"/>	4	cenouras	5.00	2.83	80.85	2.05	0	0	0	2021-06-12 13:39:57	2021-06-12 13:39:57
<input type="checkbox"/>	5	alfices	19.26	48.19	75.77	7.53	1	1	0	2021-06-12 14:22:40	2021-06-12 14:22:40
<input type="checkbox"/>	6	cenouras	22.39	67.94	74.20	9.09	0	0	0	2021-06-12 14:22:58	2021-06-12 14:22:58
<input type="checkbox"/>	7	tomates	21.60	60.70	74.60	9.09	1	0	0	2021-06-12 14:23:14	2021-06-12 14:23:14
<input type="checkbox"/>	8	alfices	7.53	0.49	83.20	0.49	1	1	0	2021-06-12 15:21:09	2021-06-12 15:21:09

**Figura 5 - Apresentação de uma tabela da base de dados através do *phpMyAdmin***

## 4. RESULTADOS OBTIDOS

Configurámos as variáveis ambiente no CPT, de tal forma que a temperatura ambiente varia entre 5°C às 00:00 subindo até ao máximo de 24°C ao 12:00, a humidade inicia a 80% desce passado 6 horas até 0% voltando a subir para 70% até às 12:00 e assim continuamente de 6 em 6 horas, já a luminosidade inicia a 0 e às 6:00 começa a aumentar até atingir o seu ponto máximo de 100% ao meio-dia a partir daí começa a sua descida.

Neste cenário conseguimos provar o funcionamento do Packet Tracer, pois o MCU enviava os valores para a API de 10 em 10 segundos, que de seguida são guardados pela API na BD, a Dashboard apresenta os valores lidos assim como o Python; de seguida o SBC faz as verificações necessários, como provámos no vídeo do projeto, se fizermos à 00:00, a luminosidade e a temperatura encontram-se com valores baixos enquanto que a humidade já se apresenta com valores altos, devido a isso apenas é acendida a lâmpada para certificar que as culturas continuam a crescer saudáveis, já às 12:00 horas, é quando a luminosidade se encontra em 100% assim como a temperatura se encontra alta, mas a humidade está com níveis reduzidos, devido a isso o SBC, abre a janela e inicia a rega, mas a lâmpada mantém-se desligada. Todo este funcionamento é totalmente automático sem qualquer toque humano para a manter, continuado de 10 em 10 segundos.

Para os restantes equipamentos o processo é exatamente o mesmo, se a temperatura ou humidade lidas pelo MCU, não forem as corretas para o bom funcionamento da plantação, o SBC recebe esses valores escolhe os equipamentos que coloca a funcionar para contrapor essa irregularidade.

## CONCLUSÃO

Sendo o propósito do relatório dar a conhecer todo o trabalho desenvolvido enquanto estagiário na entidade Estudar Portugal, resta-me apenas, no seu término, dar o meu parecer pessoal acerca do mesmo.

Indicado anteriormente e com toda a minha sinceridade colocada nas seguintes palavras, acredito que o estágio realizado não só cumpriu com as minhas expectativas, mas enriqueceu em muito, a minha experiência profissional e o meu conhecimento em diversas áreas, sendo as mesmas, as trabalhadas e referidas anteriormente no presente relatório.

Resta-me apenas, referir, que o conceito de responsabilidade foi claramente incutido pela entidade responsável pelo meu estágio, referindo assim, que todo o trabalho realizado, tinha sobre nós uma carga de responsabilidade, ao qual deixo escrito, uma responsabilidade saudável, que acrescenta em nós um sentido profissional bastante agradável.

Num último ponto, coloco sobre este relatório, e tudo aquilo que ele representa, o maior orgulho e satisfação pessoal.

## REFERÊNCIAS BIBLIOGRÁFICAS

Bootstrap. Framework for building responsive, mobile-first sites. Consulta contínua. Disponível em <https://getbootstrap.com>

Font Awesome. Icon library set and toolkit. Consulta contínua. Disponível em <https://fontawesome.com>

RegExLib. Regular Expression Library. Disponível em <http://regexlib.com>

W3schools. The World's Largest Web Developer Site. Consulta contínua. Disponível em <https://www.w3schools.com>

Laravel. Laravel – The PHP Framework for web artisans. Consulta contínua. Disponível em <https://laravel.com>

## **ANEXOS**

Anexo 1 | Excertos de código desenvolvidos

# **Anexo 1**

Excertos de código desenvolvido



Caminho do ficheiro apresentado:  
**my-garden/routes/web.php**

```
<?php
```

```
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\LoginController;
use App\Http\Controllers\DashboardController;
use App\Http\Controllers\HistoryController;

// use App\Http\Controllers\DataController;
// Uncomment this lines, and run "/data" to create the users.
// Route::get('data', [DataController::class, 'create_users']);

Route::middleware(['prevent.history'])->group(function () {
    Route::middleware(['auth'])->group(function () {
        Route::get('logout', [LoginController::class, 'logout'])->
>name('logout');
        Route::get('/', [DashboardController::class, 'index'])->
>name('dashboard');

        //Rotas para dar update ao valores dos atuadores "Rega", "Luz"
e "Janela"
        Route::put('/plant/update-watering/{plant}',
[DashboardController::class, 'update_watering']);
        Route::put('/plant/update-light/{plant}',
[DashboardController::class, 'update_light']);
        Route::put('/plant/update-window/{plant}',
[DashboardController::class, 'update_window']);

        //Rotas que retornam as views para a apresentação do histórico
dos sensores individuais
        Route::get('/historico/humidade', [HistoryController::class,
'history_hum'])->name('history.hum');
        Route::get('/historico/luminosidade',
[HistoryController::class, 'history_lum'])->name('history.lum');
        Route::get('/historico/temperatura',
[HistoryController::class, 'history_temp'])->name('history.temp');
        Route::get('/historico/vento', [HistoryController::class,
'history_wind'])->name('history.wind');

        //Rotas que retornam as views para a apresentação do histórico
das diferentes culturas
        Route::get('/historico/alfaces', [HistoryController::class,
'history_alfaces'])->name('history.alfaces');
        Route::get('/historico/cenouras', [HistoryController::class,
'history_cenouras'])->name('history.cenouras');
        Route::get('/historico/tomates', [HistoryController::class,
'history_tomates'])->name('history.tomates');

        //Rotas que retornam as views para a apresentação do histórico
dos sensores gerais
        Route::get('/historico/geral/humidade',
[HistoryController::class, 'history_general_hum'])->name('history-
geral.hum');
        Route::get('/historico/geral/temperatura',
[HistoryController::class, 'history_general_temp'])->name('history-
geral.temp');

        //Rota que retorna a view que apresenta o histórico de entrada
do utilizador autenticado
```

```
Route::get('/historico/entradas/{user}',  
[HistoryController::class, 'history_user_entrance'])-  
>name('history.entrance');  
});  
Route::get('login', [LoginController::class, 'index'])-  
>name('login')->middleware('guest');  
Route::post('authenticate', [LoginController::class,  
'authenticate'])->name('authenticate');  
});
```

Caminho do ficheiro apresentado:  
**my-garden/routes/api.php**

```
<?php
```

```
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\PlantsController;
use App\Http\Controllers\HistoryController;
use App\Http\Controllers\GeneralSensorController;
use App\Http\Controllers\EntranceController;

/*
|-----
| Rotas da API
|-----
|
| Aqui encontram-se todas as rotas criadas para o funcionamento
correto da API.
| Cada rota definida aponta para um controlador específico, sendo o
segundo
| parâmetro o nome da função a ser chamada pela mesma.
|
*/

Route::name('api.')->group(function() {
    //Pedido GET para os valores dos sensores individuais
    Route::get('/historico/humidade', [HistoryController::class,
'history_hum_api'])->name('hum');
    Route::get('/historico/luminosidade', [HistoryController::class,
'history_lum_api'])->name('lum');
    Route::get('/historico/temperatura', [HistoryController::class,
'history_temp_api'])->name('temp');
    Route::get('/historico/vento', [HistoryController::class,
'history_wind_api'])->name('wind');

    //Pedido GET para os valores dos sensores gerais
    Route::get('/historico/geral/humidade', [HistoryController::class,
'history_general_hum_api'])->name('hum-geral');
    Route::get('/historico/geral/temperatura',
[HistoryController::class, 'history_general_temp_api'])->name('temp-
geral');

    //Pedido GET para os valores das diferentes culturas
    Route::get('/historico/alfaces', [HistoryController::class,
'history_alfaces_api'])->name('alfaces');
    Route::get('/historico/cenouras', [HistoryController::class,
'history_cenouras_api'])->name('cenouras');
    Route::get('/historico/tomates', [HistoryController::class,
'history_tomates_api'])->name('tomates');
});

Route::post('plants', [PlantsController::class, 'store']);
Route::post('entrance', [EntranceController::class, 'entrance']);
Route::post('general-sensor', [GeneralSensorController::class,
'store']);
Route::post('change-door-server', [EntranceController::class,
'change_door_server']);
```

```
Route::get('cenouras', [PlantsController::class,  
'get_info_cenouras']);  
Route::get('alfaces', [PlantsController::class, 'get_info_alfaces']);  
Route::get('tomates', [PlantsController::class, 'get_info_tomates']);  
Route::get('get-state-door', [EntranceController::class,  
'get_state_door']);
```

Caminho do ficheiro apresentado:  
**my-garden/resources/views/dashboard.blade.php**

```
@extends('layout.master')
<!-- Page Title -->
@section('title', 'Dashboard')
<!-- Page Content -->
@section('content')
<!-- Begin Page Content -->
<div class="container-fluid">
    <!-- Page Heading -->
    <div class="d-sm-flex align-items-center justify-content-between mb-4">
        <h1 class="h4 mb-0 text-gray-800">Informações gerais</h1>
        <a href="#" data-toggle="modal" data-target="#infoModal"
class="btn btn-secondary btn-icon-split btn-sm" title="Informações">
            <span class="icon text-white-50">
                <i class="fas fa-info-circle"></i>
            </span>
            <span class="text">Informações</span>
        </a>
    </div>

    <!-- Content Row -->
    <div class="row">
        <!-- Vegetais -->
        <div class="col-xl-3 col-md-6 mb-4 cards">
            <div class="card border-left-primary shadow h-100 py-2">
                <div class="card-body">
                    <div class="row no-gutters align-items-center">
                        <div class="col mr-2">
                            <div class="text-xs font-weight-bold text-
primary text-uppercase mb-1">Culturas (Quantidade)</div>
                            <div class="h5 mb-0 font-weight-bold text-
gray-800">3</div>
                        </div>
                        <div class="col-auto">
                            <i class="fas fa-seedling fa-2x text-gray-
300"></i>
                        </div>
                    </div>
                </div>
            </div>
        </div>

        <!-- Temperatura -->
        <div class="col-xl-3 col-md-6 mb-4 cards">
            <div class="card border-left-success shadow h-100 py-2">
                <div class="card-body">
                    <div class="row no-gutters align-items-center">
                        <div class="col mr-2">
                            <div class="text-xs font-weight-bold text-
success text-uppercase mb-1">Temperatura (Geral)</div>
                            <div class="h5 mb-0 font-weight-bold text-
gray-800">
                                @isset($general_sensor->temperature)
                                    {{ $general_sensor->
temperature.'°C' }}
                                @else
                                    -
                                @endisset
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

```

        </div>
    </div>
    <div class="col-auto">
        <i class="fas fa-temperature-low fa-2x
text-gray-300"></i>
    </div>
</div>
</div>
</div>
</div>
</div>

<!-- Humidade -->
<div class="col-xl-3 col-md-6 mb-4 cards">
    <div class="card border-left-info shadow h-100 py-2">
        <div class="card-body">
            <div class="row no-gutters align-items-center">
                <div class="col mr-2">
                    <div class="text-xs font-weight-bold text-
success text-uppercase mb-1">Humidade (Geral)</div>
                    <div class="h5 mb-0 font-weight-bold text-
gray-800">
                        @isset($general_sensor->humidity)
                            {{ $general_sensor->humidity. '%' }}
                        @else
                            -
                        @endisset
                    </div>
                </div>
                <div class="col-auto">
                    <i class="fas fa-cloud-rain fa-2x text-
gray-300"></i>
                </div>
            </div>
        </div>
    </div>
    <div class="col-auto">
        <i class="fas fa-cloud-rain fa-2x text-
gray-300"></i>
    </div>
</div>
</div>
</div>
</div>
</div>

<!-- Hora -->
<div class="col-xl-3 col-md-6 mb-4 cards">
    <div class="card border-left-warning shadow h-100 py-2">
        <div class="card-body">
            <div class="row no-gutters align-items-center">
                <div class="col mr-2">
                    <div class="text-xs font-weight-bold text-
warning text-uppercase mb-1">Hora (Registro)</div>
                    <div class="h5 mb-0 font-weight-bold text-
gray-800">
                        @isset($general_sensor->created_at)
                            {{ date_format($general_sensor->
created_at, "H:i") }}
                        @else
                            -
                        @endisset
                    </div>
                </div>
                <div class="col-auto">
                    <i class="fas fa-clock fa-2x text-gray-
300"></i>
                </div>
            </div>
        </div>
    </div>
    <div class="col-auto">
        <i class="fas fa-clock fa-2x text-gray-
300"></i>
    </div>
</div>
</div>
</div>

```

```

        </div>
    </div>
</div>
<div class="d-sm-flex align-items-center justify-content-between mt-3">
    <h1 class="h4 mb-0 text-gray-800">Informações das
culturas</h1>
</div>
<small class="mb-4 text-muted">*Os dados abaixo apresentados
refletem o último registo efectuado às culturas.</small>
<br><br>
<div class="row">
    @foreach ($plants as $plant)
        @isset($plant)
            <div class="col-lg-4 mb-4">
                <div class="card shadow mb-4">
                    <div class="card-header py-3">
                        <h6 class="m-0 font-weight-bold text-
primary text-capitalize">{{ $plant->name }}</h6>
                    </div>
                    <div class="card-body">
                        <div class="row">
                            <span class="col-8 text-left text-
gray-700">Temperatura</span>
                            <span class="col-4 mb-0 font-
weight-bold text-gray-800 text-right">{{ $plant->temperature }}°C</span>
                        </div>
                        <hr class="sidebar-divider my-2">
                        <div class="row">
                            <span class="col-8 text-left text-
gray-700">Luminosidade</span>
                            <span class="col-4 mb-0 font-
weight-bold text-gray-800 text-right">{{ $plant->luminosity }}%</span>
                        </div>
                        <hr class="sidebar-divider my-2">
                        <div class="row">
                            <span class="col-8 text-left text-
gray-700">Humidade no solo</span>
                            <span class="col-4 mb-0 font-
weight-bold text-gray-800 text-right">{{ $plant->humidity }}%</span>
                        </div>
                        <hr class="sidebar-divider my-2">
                        <div class="row">
                            <span class="col-8 text-left text-
gray-700">Vento</span>
                            <span class="col-4 mb-0 font-
weight-bold text-gray-800 text-right">{{ $plant->wind }} km/h</span>
                        </div>
                        <hr class="sidebar-divider my-2">
                        <div class="row">
                            <span class="col-6 text-left text-
gray-700">Rega</span>
                            <span class="col-6 mb-0 font-
weight-bold text-right text-capitalize">
                                <a class="watering-btn
{{{ $plant->watering } ? "text-success" : " text-danger" }}" href="#"
data-planta="{{ $plant->id }}">{{ $plant->watering } ? "Ligada" : "
Desligada" }}</a>
                            </span>
                        </div>
                        <hr class="sidebar-divider my-2">
                    </div>
                </div>
            </isset>
        </foreach>
    </div>

```

```

        <div class="row">
            <span class="col-6 text-left text-gray-700">Luz</span>
            <span class="col-6 mb-0 font-weight-bold text-right text-capitalize">
                <a class="light-btn {{$plant->light) ? "text-success" : " text-danger"}}" href="#" data-planta="{{$plant->id}}">{{$plant->light) ? "Ligada" : "Desligada"}}</a>
            </span>
        </div>
        <hr class="sidebar-divider my-2">
        <div class="row">
            <span class="col-6 text-left text-gray-700">Janela</span>
            <span class="col-6 mb-0 font-weight-bold text-right text-capitalize">
                <a class="window-btn {{$plant->window_state) ? "text-success" : " text-danger"}}" href="#" data-planta="{{$plant->id}}">{{$plant->window_state) ? "Aberta" : "Fechada"}}</a>
            </span>
        </div>
        <hr class="sidebar-divider my-2">
        <div class="row">
            <span class="col-4 text-left text-gray-700">Atualização</span>
            <span class="col-8 mb-0 font-weight-bold text-gray-800 text-right">{{date_format($plant->created_at, "d/m/Y H:i")}} </span>
        </div>
    </div>
</div>
@endisset
@endforeach
</div>
<!-- End of container-fluid -->

<!-- Modal for more information -->
<div class="modal fade" id="infoModal" tabindex="-1" role="dialog" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header pl-4 pb-1 pt-4">
                <h5 class="modal-title text-gray-800 font-weight-bold">Para que serve?</h5>
                <button type="button" class="close" data-dismiss="modal" aria-label="Close" id="close-button">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body text-gray-800 pl-4 pr-5">
                Na <i>dashboard</i>, pode encontrar todos os dados gerais acerca das estufas My Garden, bem como outros dados importantes para as culturas que se criam.
            </div>
            <div class="modal-footer mt-3">
                <a data-dismiss="modal" class="mr-4 font-weight-bold id="close-option">Fechar</a>
            </div>
        </div>
    </div>
</div>

```



```

        <button type="button" data-dismiss="modal" class="btn
btn-success font-weight-bold mr-2">Entendido!</button>
    </div>
</div>
</div>
</div>
<!-- End of Modal for more information -->
@section('scripts')
<script>
$(document).ready(function() {
    // AJAX Request para a atualização do estado da rega
    $(".watering-btn").click(function(event) {
        event.preventDefault();

        var button = $(event.relatedTarget);
        var anchor = $(this);
        var info = this.text;

        $.ajaxSetup({
            headers: {
                'X-CSRF-TOKEN': "{{csrf_token()}}"
            }
        });

        $.ajax({
            type: "put",
            url: "/plant/update-watering/"+anchor.data('planta'),
            context: this,
            success: function(data) {
                if (info === "Ligada") {
                    anchor.text("Desligada").removeClass("text-
success").addClass("text-danger");
                } else {
                    anchor.text("Ligada").removeClass("text-
danger").addClass("text-success");
                }
            },
            error: function() {
                alert("Não é possível desligar a rega, porque o valor
da humidade está abaixo de 20%.");
            }
        });
    });

    // AJAX Request para a atualização do estado da luz
    $(".light-btn").click(function(event) {
        event.preventDefault();

        var button = $(event.relatedTarget);
        var anchor = $(this);
        var info = this.text;

        $.ajaxSetup({
            headers: {
                'X-CSRF-TOKEN': "{{csrf_token()}}"
            }
        });

        $.ajax({
            type: "put",
            url: "/plant/update-light/"+anchor.data('planta'),

```

```

        context: this,
        success: function(data) {
            if (info === "Ligada") {
                anchor.text("Desligada").removeClass("text-
success").addClass("text-danger");
            } else {
                anchor.text("Ligada").removeClass("text-
danger").addClass("text-success");
            }
        },
        error: function() {
            alert("Não é possível desligar a luz, porque o valor
da luminosidade está abaixo de 30%.");
        }
    });
});

```

```

// AJAX Request para a atualização do estado da janela
$(".window-btn").click(function(event) {
    event.preventDefault();

```

```

    var button = $(event.relatedTarget);
    var anchor = $(this);
    var info = this.text;

```

```

    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': "{{csrf_token()}}"
        }
    });

```

```

    $.ajax({
        type: "put",
        url: "/plant/update-window/"+anchor.data('planta'),
        context: this,
        success: function(data) {
            if (info === "Aberta") {
                anchor.text("Fechada").removeClass("text-
success").addClass("text-danger");
            } else {
                anchor.text("Aberta").removeClass("text-
danger").addClass("text-success");
            }
        },
        error: function() {
            alert("Não é possível abrir a janela, porque a
velocidade do vento está acima de 10 km/h.");
        }
    });
});

```

```

});
</script>
@endsection
@endsection

```

Caminho do ficheiro apresentado:  
**my-garden/app/Http/Controllers/DashboardController.php**

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Models\Plant;
```

```
use App\Models\GeneralSensor;
```

```
use Illuminate\Http\Request;
```

```
class DashboardController extends Controller
```

```
{
    public function index()
    {
        $general_sensor = GeneralSensor::orderBy("created_at",
"DESC")->take(1)->first();
        $cenoura = Plant::where("name", "cenouras")->
orderBy("created_at", "DESC")->take(1)->first();
        $alface = Plant::where("name", "alfaces")->
orderBy("created_at", "DESC")->take(1)->first();
        $tomate = Plant::where("name", "tomates")->
orderBy("created_at", "DESC")->take(1)->first();
        $plants = array($cenoura, $alface, $tomate);
        return view('dashboard', compact('plants', 'general_sensor'));
    }
}
```

```
    public function update_watering(Plant $plant)
    {
        if ($plant->watering) {
            if ($plant->humidity < 20) {
                return response()->json("NOK", 403);
            } else {
                Plant::where('id', $plant->id)->update(['watering' =>
0]);
            }
        } else {
            Plant::where('id', $plant->id)->update(['watering' => 1]);
        }
        return response()->json("Watering updated.", 200);
    }
}
```

```
    public function update_light(Plant $plant)
    {
        if ($plant->light) {
            if ($plant->luminosity < 30) {
                return response()->json("NOK", 403);
            } else {
                Plant::where('id', $plant->id)->update(['light' =>
0]);
            }
        } else {
            Plant::where('id', $plant->id)->update(['light' => 1]);
        }
        return response()->json("Light state updated.", 200);
    }
}
```

```
    public function update_window(Plant $plant)
    {
        if (!$plant->window_state) {
```

```
        if ($plant->wind < 10) {
            Plant::where('id', $plant->id)->update(['window_state'
=> 1]);
        }else {
            return response()->json("NOK", 403);
        }
    }else {
        Plant::where('id', $plant->id)->update(['window_state' =>
0]);
    }
    return response()->json("Window state updated.", 200);
}
```

Caminho do ficheiro apresentado:  
**my-garden/app/Http/Controllers/PlantsController.php**

```
<?php

namespace App\Http\Controllers;

use App\Models\Plant;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class PlantsController extends Controller
{
    public function store(Request $request)
    {
        if ($request->has('name')) {
            $request->merge(array('name' => strtolower($request->name)));
        }

        $validator = Validator::make($request->all(), [
            'name' => 'required|alpha|in:alfaces,tomates,cenouras',
            'temperature' => 'required|numeric|between:-100,100',
            'luminosity' => 'required|numeric|between:-100,100',
            'humidity' => 'required|numeric|between:-100,100',
            'wind' => 'required|numeric|between:-100,100',
            'light' => 'required|boolean',
            'watering' => 'required|boolean',
            'window_state' => 'required|boolean'
        ],
        $messages = [
            'name.in' => 'The selected name is invalid. Valid options are \'alfaces\', \'cenouras\' and \'tomates\'.'
        ]
    );

    if ($validator->fails()) {
        return response()->json($validator->errors()->all(), 400);
    } else {
        $plant = Plant::create($request->all());
    }

    return response()->json("Registo POST guardado com sucesso!", 201);
}

    public function get_info_tomates()
    {
        $tomate = Plant::where("name", "tomates")->orderBy("created_at", "DESC")->take(1)->first();
        return response()->json($tomate, 200);
    }

    public function get_info_cenouras()
    {
        $cenoura = Plant::where("name", "cenouras")->orderBy("created_at", "DESC")->take(1)->first();
        return response()->json($cenoura, 200);
    }
}
```

```
public function get_info_alfaces()
{
    $alface = Plant::where("name", "alfaces")-
>orderBy("created_at", "DESC")->take(1)->first();
    return response()->json($alface, 200);
}
```

Caminho do ficheiro apresentado:  
**my-garden/public/py/recognition.py**

```
import face_recognition
import cv2 as cv
import requests
import time

# Função que envia os dados acerca do reconhecimento facial para a API
def send_to_api(array):
    r = requests.post('http://my-garden.test/api/change-door-server',
data=array)
    if r.status_code == 200:
        print(r.json())
    else:
        print("ERRO: Não foi possível realizar o pedido")

# Função que envia os dados acerca do reconhecimento facial para a API
def get_door_state():
    r = requests.get('http://my-garden.test/api/get-state-door')
    if r.status_code == 200:
        return r.json()
    else:
        print("ERRO: Não foi possível realizar o pedido")

# Função que é responsável pela captura da imagem do utilizador do
script
def capture_image():
    cap = cv.VideoCapture(0)
    while True:
        _, img = cap.read()
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        cv.imshow('Webcam', img)
        # A foto só é tirada se o utilizador pressionar ESC(27) na
janela da webcam
        k = cv.waitKey(30) & 0xff
        if(k == 27):
            break
    cap.release()
    cv.destroyAllWindows()
    return img

try:
    print("***Início do programa***\n")
    print("Caso pretenda terminar o programa, pressione CTRL+C.")
    print("Clique ESC para fazer uma captura da sua cara.")

    # Chamada da função para tirar uma foto sendo a mesma guardada na
pasta do script com o nome "face.jpg"
    image = capture_image()
    cv.imwrite('face.jpg', image)

    # Carregamento das imagens dos utilizadores conhecidos, bem como a
imagem tirada anteriormente, para arrays
    ivo_image = face_recognition.load_image_file("face_ivo.jpg")
    jose_image = face_recognition.load_image_file("face_jose.jpg")
    unknown_image = face_recognition.load_image_file("face.jpg")

    try:
        print("A fazer o reconhecimento facial...")
```

```

        # Obter um encoding para cada face na imagem
        # Como sabemos que apenas irá haver uma face em cada imagem
obtemos o primeiro valor do array
        jose_face_encoding =
face_recognition.face_encodings(jose_image)[0]
        ivo_face_encoding =
face_recognition.face_encodings(ivo_image)[0]
        unknown_face_encoding =
face_recognition.face_encodings(unknown_image)[0]
    except IndexError:
        print("Não foi possível localizar nenhuma cara na imagem.
Abortar...")
        quit()

    known_faces = [
        jose_face_encoding,
        ivo_face_encoding
    ]

    # O resultado é um array de True/False que diz se a face
"desconhecida" é
    # reconhecida por uma das faces de "known_faces"
    results = face_recognition.compare_faces(known_faces,
unknown_face_encoding)

    # Estrutura de decisão, caso seja reconhecida a face
"desconhecida" chama a função "send_to_api"
    # para o envio dos dados necessários para a abertura da porta do
server room
    if results[0] == True:
        array = {'name': 'José Areia', 'value': '1'}
        send_to_api(array)
    elif results[1] == True:
        array = {'name': 'Ivo Bispo', 'value': '1'}
        send_to_api(array)
    else:
        print("O sistemas não conseguiu-o reconhecer a sua cara.")

    time.sleep(7)

    door_state = get_door_state()
    if door_state == False:
        print("Aviso! A porta foi fechada.")

except KeyboardInterrupt:
    print("\nPrograma terminado pelo utilizador!")
finally:
    print("\n***Fim do programa***")

```



Caminho do ficheiro apresentado:  
**my-garden/public/py/sensors.py**

```
import requests
import sys

# Função que envia os dados em modo POST para a API
def send_to_api(array):
    print(array)
    r = requests.post('http://my-garden.test/api/plants', data=array)
    if r.status_code == 201:
        print(r.json())
    else:
        invalid_pedido()

# Função que faz o GET dos dados da API
def get_cenouras():
    r = requests.get('http://my-garden.test/api/cenouras')
    if r.status_code == 200:
        return r.json()
    else:
        invalid_pedido()

# Função que faz o GET dos dados da API
def get_alfaces():
    r = requests.get('http://my-garden.test/api/alfaces')
    if r.status_code == 200:
        return r.json()
    else:
        invalid_pedido()

# Função que faz o GET dos dados da API
def get_tomates():
    r = requests.get('http://my-garden.test/api/tomates')
    if r.status_code == 200:
        return r.json()
    else:
        invalid_pedido()

# Caso a opção do menu seja inválida apresenta mensagem
def invalid():
    print("OPÇÃO INVÁLIDA!")

# Caso o POST não tenha sido bem realizado
def invalid_pedido():
    print("ERRO: Não foi possível realizar o pedido")

try :
    print("Caso pretenda terminar o programa, pressione CTRL+C.")
    while True:
        # cria o dicionário menu plantas
        menu_plants = {"1": "Cenouras",
                       "2": "Alfaces",
                       "3": "Tomates" }

        # aprenseta o menu plantas
        print("\n**MENU PLANTAÇÕES**\n")
        for key in sorted(menu_plants.keys()):
            print(key+": " + menu_plants[key])
```

```

# pede a opção
ans = int(input("Opção: "))

if ans == 1:
    # Recebe os values das cenouras
    cenouras_values = get_cenouras()

    # Cria o dicionário menu sensores
    menu_sensor_cenoura = {"1": "Temperatura Atual: " +
str(cenouras_values['temperature']) + " °C",
        "2": "Luminosidade Atual: " +
str(cenouras_values['luminosity']) + " %",
        "3": "Humidade Atual: " +
str(cenouras_values['humidity']) + " %",
        "4": "Vento Atual: " + str(cenouras_values['wind']) + "
km/h" }

    # Aprenseta o menu cenouras
    print("\n**MENU CENOURAS**\n")
    for key in sorted(menu_sensor_cenoura.keys()):
        print(key+"": " + menu_sensor_cenoura[key])

    # Pede a opção
    ans_cenouras = int(input("Opção: "))

    if ans_cenouras == 1:
        ans_cenouras_temp = int(input("Temperatura: "))
        # Altera a variável Temperatura para a inserida
        cenouras_values['temperature'] = ans_cenouras_temp

    elif ans_cenouras == 2:
        ans_cenouras_lum = int(input("Luminosidade: "))
        # Altera a variável Luminosidade para a inserida
        cenouras_values['luminosity'] = ans_cenouras_lum

    elif ans_cenouras == 3:
        ans_cenouras_hum = int(input("Humidade: "))
        # Altera a variável Humidade para a inserida
        cenouras_values['humidity'] = ans_cenouras_hum

    elif ans_cenouras == 4:
        ans_cenouras_wind = int(input("Vento: "))
        # Altera a variável Vento para a inserida
        cenouras_values['wind'] = ans_cenouras_wind

    else:
        # Opção inválida

menu_sensor_cenoura.get(ans_cenouras, [None, invalid])[1]()

    # Cria e Envia o dicionário para ser enviado para a API,
    com os novos valores
    values = {'name': 'cenouras',
        'temperature': cenouras_values['temperature'],
        'luminosity': cenouras_values['luminosity'],
        "humidity": cenouras_values['humidity'],
        'wind': cenouras_values['wind'],
        'light': cenouras_values['light'],
        'watering': cenouras_values['watering'],
        'window_state': cenouras_values['window_state']}
    send_to_api(values)

```

```

elif ans == 2:
    # Recebe os valores das alfaces
    alfaces_values = get_alfaces()

    # Cria o dicionário menu sensores
    menu_sensor_alfaces = {"1": "Temperatura Atual: " +
        str(alfaces_values['temperature']) + " °C",
        "2": "Luminosidade Atual: " +
        str(alfaces_values['luminosity']) + " %",
        "3": "Humidade Atual: " +
        str(alfaces_values['humidity']) + " %",
        "4": "Vento Atual: " + str(alfaces_values['wind']) + "
km/h" }

    # Aprenseta o menu alfaces
    print("\n**MENU ALFACES**\n")
    for key in sorted(menu_sensor_alfaces.keys()):
        print(key+":" + menu_sensor_alfaces[key])

    # Pede a opção
    ans_alfaces = int(input("Opção: "))

    if ans_alfaces == 1:
        ans_alfaces_temp = int(input("Temperatura: "))
        # Altera a variável Temperatura para a inserida
        alfaces_values['temperature'] = ans_alfaces_temp

    elif ans_alfaces == 2:
        ans_alfaces_lum = int(input("Luminosidade: "))
        # Altera a variável Luminosidade para a inserida
        alfaces_values['luminosity'] = ans_alfaces_lum

    elif ans_alfaces == 3:
        ans_alfaces_hum = int(input("Humidade: "))
        # Altera a variável Humidade para a inserida
        alfaces_values['humidity'] = ans_alfaces_hum

    elif ans_alfaces == 4:
        ans_alfaces_wind = int(input("Vento: "))
        # Altera a variável Vento para a inserida
        alfaces_values['wind'] = ans_alfaces_wind

    else:
        # Opção inválida

menu_sensor_alfaces.get(ans_alfaces, [None, invalid])[1]()

    # Cria e Envia o dicionário para ser enviado para a API,
    com os novos valores
    values = {'name': 'alfaces',
        'temperature': alfaces_values['temperature'],
        'luminosity': alfaces_values['luminosity'],
        'humidity': alfaces_values['humidity'],
        'wind': alfaces_values['wind'],
        'light': alfaces_values['light'],
        'watering': alfaces_values['watering'],
        'window_state': alfaces_values['window_state']}
    send_to_api(values)

elif ans == 3:

```

```

# Recebe os valores das tomates
tomates_values = get_tomates()

# Cria o dicionário menu sensores
menu_sensor_tomates = {"1": "Temperatura Atual: " +
str(tomates_values['temperature']) + " °C",
"2": "Luminosidade Atual: " +
str(tomates_values['luminosity']) + " %",
"3": "Humidade Atual: " +
str(tomates_values['humidity']) + " %",
"4": "Vento Atual: " + str(tomates_values['wind']) + "
km/h" }

# Aprenseta o menu tomates
print("\n**MENU TOMATES**\n")
for key in sorted(menu_sensor_tomates.keys()):
    print(key+": " + menu_sensor_tomates[key])

# Pede a opção
ans_tomates = int(input("Opção: "))

if ans_tomates == 1:
    ans_tomates_temp = int(input("Temperatura: "))
    # Altera a variável Temperatura para a inserida
    tomates_values['temperature'] = ans_tomates_temp

elif ans_tomates == 2:
    ans_tomates_lum = int(input("Luminosidade: "))
    # Altera a variável Luminosidade para a inserida
    tomates_values['luminosity'] = ans_tomates_lum

elif ans_tomates == 3:
    ans_tomates_hum = int(input("Humidade: "))
    # Altera a variável Humidade para a inserida
    tomates_values['humidity'] = ans_tomates_hum

elif ans_tomates == 4:
    ans_tomates_wind = int(input("Vento: "))
    # Altera a variável Vento para a inserida
    tomates_values['wind'] = ans_tomates_wind

else:
    # Opção inválida

menu_sensor_tomates.get(ans_tomates, [None, invalid])[1]()

# Cria e Envia o dicionário para ser enviado para a API,
com os novos valores
values = {'name': 'tomates',
'temperature': tomates_values['temperature'],
'luminosity': tomates_values['luminosity'],
"humidity": tomates_values['humidity'],
'wind': tomates_values['wind'],
'light': tomates_values['light'],
'watering': tomates_values['watering'],
>window_state': tomates_values ['window_state']}
send_to_api(values)

else:
    # opção inválida
    menu_plants.get(ans, [None, invalid])[1]()

```

```
except KeyboardInterrupt: # caso haja interrupção de teclado CTRL+C
    print( "Programa terminado pelo utilizador")

except : # caso haja um erro qualquer
    print( "Ocorreu um erro:", sys.exc_info() )

finally : # executa sempre, independentemente se ocorreu exception
    print( "Fim do programa")
```

Caminho do ficheiro apresentado:

**CPT – SBC-PT - atuador-microcontroller-garden-alfaces**

```
//URL da API que ira buscar os valores dos sensores e atuadores
var url = "http://my-garden.test/api/alfaces";

var REGA = 0;
var JANELA = 1;
var LAMPADA = 2;
var LCD1 = 3;
var LCD2 = 4;
var LCD3 = 5;

function setup() {
  pinMode(REGA, OUTPUT);
  pinMode(JANELA, OUTPUT);
  pinMode(LAMPADA, OUTPUT);
  pinMode(LCD1, OUTPUT);
  pinMode(LCD2, OUTPUT);
  pinMode(LCD3, OUTPUT);
}

function loop() {
  RealHttpClient.get(url, function(status, data){
    //Transforma os valores em dicionario
    var valores = JSON.parse(data);
    if(status == 201){
      //Liga a rega consoante o valor da humidade ou o valor
      bool da dashboard
      if (valores.humidity < 20 || valores.watering) {
        customWrite(REGA, "1");
        customWrite(LCD2, "REGA: ON");
      } else {
        customWrite(REGA, "0");
        customWrite(LCD2, "REGA: OFF");
      }

      //Liga a luz consoante o valor da luminosidade ou o valor
      bool da dashboard
      if (valores.luminosity < 30 || valores.light) {
        customWrite(LAMPADA, "2");
        customWrite(LCD1, "LAMPADA: ON");
      } else {
        customWrite(LAMPADA, "0");
        customWrite(LCD1, "LAMPADA: OFF");
      }

      //Abre a janela consoante o valor do vento ou o valor bool
      da dashboard
      if (valores.wind > 10 || !valores.window_state) {
        customWrite(JANELA, "0");
        customWrite(LCD3, "JANELA: OFF");
      } else {
        customWrite(JANELA, "1");
        customWrite(LCD3, "JANELA: ON");
      }
    } else {
      Serial.println("Erro na leitura de dados! (" + status +
        ")");
    }
  })
}
```

```
});  
  
//Espera 5 segundos para uma nova leitura de valores  
delay(5000);  
}
```

Caminho do ficheiro apresentado:

## CPT – MCU-PT - sensor-microcontroller-garden-alfaces

//URL da API que ira receber os valores dos sensores e atuadores das  
alfaces

```
var url = "http://my-garden.test/api/plants";
```

```
function setup() {
  pinMode(A0, INPUT); //Sensor de temperatura
  pinMode(A1, INPUT); //Sensor de humidade
  pinMode(0, INPUT); //Sensor de vento
  pinMode(1, INPUT); //Sensor de luminosidade
  pinMode(2, INPUT); //Atuador rega
  pinMode(3, INPUT); //Atuador luz
  pinMode(4, INPUT); //Atuador janela
}

function loop() {
  //Leitura dos valores dos sensores e atuadores
  var temperatura = (((analogRead(A0)/1023)*200)-100).toFixed(2);
  var humidade = (map(analogRead(A1), 0, 1023, 0, 100) +
0.5).toFixed(2);
  var vento = (((digitalRead(0)/1023)*200)-100).toFixed(2);
  var luminosidade = (((digitalRead(1)/1023)*200)-100).toFixed(2);
  var rega = customRead(2);
  var luz = customRead(3);
  var janela = customRead(4);

  //A API espera receber um bool, logo o valor 2 tera que ser
  convertido para 1
  if(luz == 2) {
    luz = 1;
  }

  //Array que guarda os valores a serem enviados para a API
  var valores = {
    'name': 'alfaces',
    'temperature': temperatura,
    'luminosity' : luminosidade,
    'humidity': humidade,
    'wind': vento,
    'light': luz,
    'watering': rega,
    'window_state': janela
  };

  //Envio dos dados guardados no array anterior para a API
  RealHTTPClient.post(url, valores, function(status, data){
    Serial.println(data + " - " + status);
  });

  //Espera 10(?) segundos para o envio de outros valores
  delay(10000);
}
```



Caminho do ficheiro apresentado:  
**CPT – Wind Sensor (Sensor modificado)**

```
var ENVIRONMENT_NAME = "Wind Speed";
var MIN = -100;
var MAX = 100;
var value;

function loop() {
  value = Environment.get(ENVIRONMENT_NAME);
  if (value < MIN)
    value = MIN;
  else if (value > MAX)
    value = MAX;
  setDeviceProperty(getName(), "level", value);
  value = Math.floor(map(value, MIN, MAX, 0, 255));

  analogWrite(A0, value);
  analogWrite(0, value);
  delay(1000);
}
```