



Instituto Tecnológico de Estudios Superiores de Monterrey
Campus Querétaro

Documentación - Go Life!

Manuel Villalpando Linares

A01352033

Ian Joab Padrón Corona

A01708940

Francisco Couttolenc Ortiz

A01754959

Fecha de entrega: 29 de noviembre de 2022
Implementación de internet de las cosas (Gpo 501)

Introducción al proyecto	3
- Sobre Go life!	3
- Misión	3
- Visión	3
- Valores	3
- Propósito	3
Creación de base de datos en MySQL	5
Configuración del NodeMCU	6
Alambrado del NodeMCU	8
Código del Proyecto	8
Código de enlace PostData -> MySQL	11
Conexión y Arranque del Servidor/Código	12
Página Web	13
Conexión MySQL -> Página web mediante php	16

Introducción al proyecto

“No hay que ser un experto”

- Sobre Go life!

Somos una organización que se preocupa por las plantas, procurando su bienestar y se apasiona por adentrarse aún más a esas personas que muestran interés por tener plantas, es por eso que nosotros decidimos desarrollar un método de ayuda eficiente en la manera de cultivar plantas, en el cual ni siquiera se requiere conocimiento previo acerca de su humedad, temperatura o ubicación ideal.

- Misión

Buscamos apoyar principalmente la ODS 15, Vida de ecosistemas terrestres, de modo que las personas puedan comprar plantas teniendo la certeza de que habrá una aplicación web de monitoreo en vivo de sus plantas para tener noción en tiempo real de si es que les hace falta algo y poder atenderlo lo más antes posible.

- Visión

Queremos un mundo lleno de plantas, en el cual no se necesite ser un experto en botánica como dice nuestro lema, para poder comenzar a tener tu propio jardín o espacio zen dentro de tu hogar.

- Valores

Somos guiados principalmente por el amor a las plantas y tenemos noción de la importancia que tiene la ecología no solo en el medio ambiente sino la manera en la que influye dentro de nuestras vidas incluso psicológicamente, creando espacios seguros donde las personas se sientan en paz.

- Propósito

Promover que la gente tenga plantas, sin importar que no tenga la información exacta acerca de las necesidades que requiere plantar algún cultivo como chiles por ejemplo.

Creación de base de datos en MySQL

Primero que nada, para la creación de nuestra aplicación web, se debe tener definida una base de datos, para esto utilizaremos la aplicación de XAMPP, para poder hacer que nuestra computadora funcione como un servidor local al cual estaremos subiendo la información. Posteriormente iniciaremos el phpMyAdmin con el usuario y contraseñas predeterminadas de la aplicación, en este caso el usuario es "root" y la contraseña es un espacio en blanco " " (Si es que lo llegase a pedir).

Ya dentro del localhost, posterior al razonamiento de cómo es que irá estructurada, (¿Con qué llaves principales o foráneas cuenta? ¿Cuáles son sus tablas? ¿Cuáles son sus columnas?) pasaremos a crear la base de datos con ayuda del lenguaje SQL.

```
CREATE DATABASE maceta;
```

```
USE maceta;
```

```
CREATE TABLE planta(  
    id INT(5) NOT NULL,  
    date DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP,  
    Nombre VARCHAR(15) NOT NULL  
    Temperatura DECIMAL(3, 2) NOT NULL,  
    Humedad DECIMAL(3, 2) NOT NULL,  
    Ubicacion VARCHAR(6),  
  
    PRIMARY KEY (id, Nombre)  
);
```

```
CREATE TABLE medidas(  
    Conteo INT AUTO_INCREMENT,  
    Dia DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP,  
    Hora TIME NOT NULL  
    NombreP VARCHAR(15) NULL,  
    Temp_actual DECIMAL(3, 2) NOT NULL,  
    Humed_actual DECIMAL(3, 2) NOT NULL,  
  
    PRIMARY KEY (Conteo),  
    CONSTRAINT nombre FOREIGN KEY (NombreP) REFERENCES planta(Nombre)  
);
```

Al principio únicamente le indicamos al mySQL que queremos crear una nueva base de datos, en este caso con el nombre de maceta, por lo cual, seguido de eso se tiene que indicar que estaremos trabajando con esa base de datos

Configuración del NodeMCU

Para la realización de este proyecto, ocupamos el CHIP-INTEGRADO: ESP8266. Con este chip, podremos recopilar datos de temperatura y humedad (requeridos para el proyecto) y enviarlos a una base de datos.

Característica	ESP8266
Procesador	Tensilica LX106 32 bit a 80 MHz (hasta 160 MHz)
Memoria RAM	80 kB (40 kB disponibles)
Memoria Flash	Hasta 4 MB
ROM	No
Alimentación	3.0 a 3.6 V
Rango de temperaturas	-40°C a 125°C
Consumo de corriente	80 mA (promedio). 225 mA máximo
Consumo en modo sueño profundo	20 uA (RTC + memoria RTC)
Coprocesador de bajo consumo	No
WiFi	802.11 b/g/n (hasta +20 dBm) WEP, WPA
Soft-AP	Sí

Además, utilizamos un sensor de humedad DHT-11 de tres terminales, con las siguientes características:

MODELO	DHT11
Alimentación	de 3,5 V a 5 V
Consumo	2,5 mA
Señal de salida	Digital
Temperatura	
Rango	de 0°C a 50°C
Precisión	a 25°C \pm 2°C
Resolución	0.1°C
Humedad	
Rango	de 20% RH a 90% RH
Precisión	entre 0°C y 50°C \pm 5% RH
Resolución	1% RH

Consideraciones

Los pasos que se mencionan a continuación servirán únicamente para el Sistema Operativo Windows, por conveniencia del equipo para la realización de este proyecto.

Para poder extender el proyecto desde una red LAN, hasta un servidor Público, hicimos uso de un servicio 'playit.gg' que nos permite abrir los puertos de la computadora en LAN, para que puedan ser accedidos a través de internet, como si se tratara de un servidor funcional.

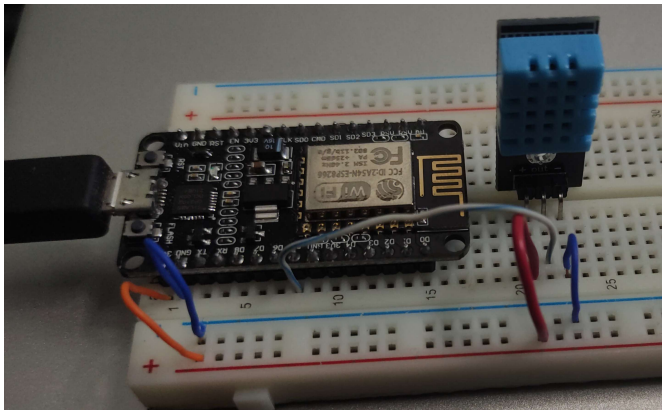
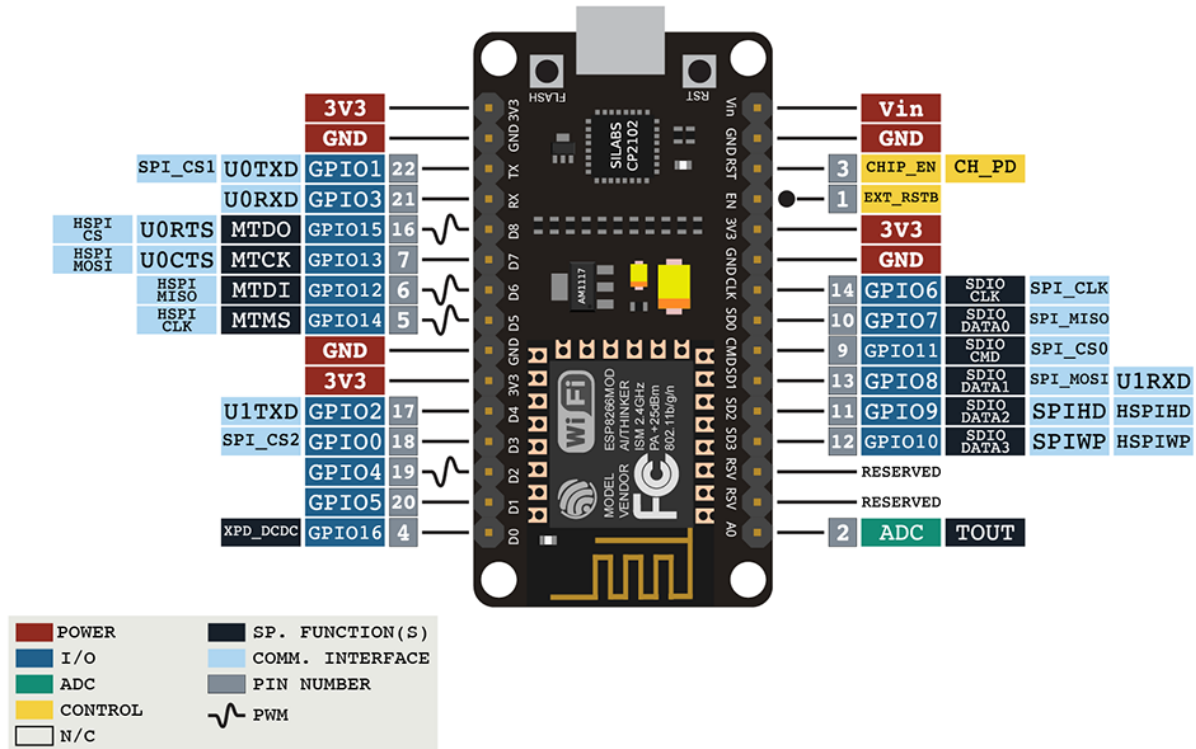
Pasos

1. Descargar el Software de Arduino IDE.
 - a. Para descargar en Windows para versiones anteriores a W10:
<https://www.arduino.cc/en/software>
 - b. Para W10 y posteriores, puede ser desde la tienda de Windows:
<https://apps.microsoft.com/store/detail/arduino-ide/9NBLGGH4RSD8?hl=es-mx&gl=mx>
2. Instalar los Drivers del NodeMCU. Si bien, en W10, al conectar el Chip a la computadora, se instala automáticamente, es mejor utilizar el archivo instalador de drivers ofrecido por: 'Silicon Labs' los creadores del micro controlador integrado al ESP8266: <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>
3. Agregar la tarjeta 'NodeMCU' al IDE de Arduino
 - a. Para ello, podemos encontrar el apartado 'Additional Boards Manager' en File>>Preferences>>Settings
 - b. Utilizamos la siguiente URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json
 - c. Ahora que el IDE ya cargó las librerías requeridas, vamos a instalarlas en Tools>>Board>>BoardManager y buscamos 'ESP8266'. Instalamos la última versión (Actualmente la 3.0.2)
4. Agregar la librería 'DHT sensor library for ESPx'
 - a. Abrimos el gestor de Librerías con Ctrl+Shift+I
 - b. Buscar la librería e instalar la última versión (autor beeg_ee_tokyo)
5. Configurar las características específicas del NodeMCU en Arduino. Al menos para la versión que utilizamos (V2) se utilizan las siguientes características:

WiFi101 / WiFinINA Firmware Updater	
Placa: "NodeMCU 1.0 (ESP-12E Module)"	>
Built-in Led: "2"	>
Upload Speed: "921600"	>
CPU Frequency: "80 MHz"	>
Flash Size: "4MB (FS:3MB OTA;~512KB)"	>
Debug port: "Disabled"	>
Debug Level: "Ninguno"	>
lwIP Variant: "v2 Lower Memory"	>
VTables: "Flash"	>
C++ Exceptions: "Disabled (new aborts on oom)"	>
Stack Protection: "Disabled"	>
Erase Flash: "Only Sketch"	>
SSL Support: "All SSL ciphers (most compatible)"	>
MMU: "32KB cache + 32KB IRAM (balanced)"	>
Non-32-Bit Access: "Use pgm_read macros for IRAM/PROGMEM"	>
Puerto	>

Alambrado del NodeMCU

No se entrará en profundidad al respecto. Sin embargo, es necesario verificar que el PIN de lectura de datos sea el correspondiente al establecido en el Código de Arduino, de acuerdo al diagrama específico del NodeMCU. Para el caso del utilizado por nuestro equipo es:



Código del Proyecto

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include "DHTesp.h"

```

```

#define HOST "iot.niubycraft2.playit.gg:6286" //HOST URL

#define WIFI_SSID "" // WIFI SSID
#define WIFI_PASSWORD "" // WIFI password here

#define DHTpin 14

// Variables which will be uploaded to server

float val1 = 0;
float val2 = 0;
String sendval, sendval2, postData;

DHTesp dht; // Initialize DHT Sensor

void setup(){
  Serial.begin(115200); // Information transfer rate from Arduino Code to NodeMCU
  Serial.println("Communication Started \n\n");
  delay(1000);

  dht.setup(DHTpin, DHTesp::DHT11); // GPIO14

  pinMode(LED_BUILTIN, OUTPUT); // Initialize NodeMCU LED

  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.print("Connecting to: ");
  Serial.print(WIFI_SSID);

  // Try to connect with WiFi
  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(500);
  }

  Serial.println();
  Serial.print("\nSuccess!");
  Serial.print("\nIP Address is: ");
  Serial.println(WiFi.localIP()); // Print local IP address

  delay(30);
}

void loop() {

```



```

HTTPClient http;                                // Http object of class HTTPClient
WiFiClient wclient;                             // Wifi-Client object of class HTTPClient

val1 = dht.getTemperature();                    // Gets the values of the temperature
val2 = dht.getHumidity();                       // Gets the values of the humidity

//Check if values are NaN or invalid
if(isnan(val1) || isnan(val2)){
  Serial.println("Error reading values from DHT Sensor");
  Serial.println("Sending 1's to Database");
  sendval = String(1);
  sendval2 = String(1);
}
else{
  // Convert float variables to string
  sendval = String(val1);
  sendval2 = String(val2);
}

postData = "sendval=" + sendval + "&sendval2=" + sendval2;

// Connect to host where dbwrite is located (PHP File to Write into SQL Database)
http.begin(wclient, "http://iot.niubycraft2.playit.gg:6286/dbwrite.php");

// Specify content-type header
http.addHeader("Content-Type", "application/x-www-form-urlencoded");

// Send POST request to php file and store server response code in variable named
httpCode

int httpCode = http.POST(postData);
Serial.println("Values to send are: Temperature = " + sendval + " && Humidity = "+sendval2 );

// Connection established with SQL Database
if (httpCode == 200){
  String webpage = http.getString();
  Serial.println(webpage + "\n");
  Serial.println("Values uploaded successfully.");
}

// Connection failed with SQL Database
else {
  Serial.println("Failed to upload values.Error code: ");
  Serial.println(httpCode);
  http.end();
  return;
}

```

```

delay(3000);
digitalWrite(LED_BUILTIN, LOW);
delay(3000);
digitalWrite(LED_BUILTIN, HIGH);
}

```

En este código básicamente se inicializan todas las variables necesarias para poder hacer distintas cosas:

- Leer los datos del sensor
- Crear un Objeto de tipo WiFiClient, con el cuál el NodeMCU podrá comunicarse a internet y enviar datos
- Crear un Objeto de tipo HTTPClient, para poder enviar datos via php a la base de datos
- Crear un Query (postData) con la información de temperatura y humedad que se enviarán a un php dentro del servidor, para que cargue los datos a la base de datos

Código de enlace PostData -> MySQL

```

<?php

// host = localhost because database hosted on the same server where PHP files are hosted
$host = "localhost";
$dbname = " "; // Database name
$username = " "; // Database username
$password = " "; // Database password

// Establish connection to MySQL database
$conn = new mysqli($host, $username, $password, $dbname);

// Check if connection established successfully
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

else { echo "Connected to mysql database. "; }

// Get date and time variables
date_default_timezone_set('America/Mexico_City'); // For other timezones, refer:-
https://www.php.net/manual/en/timezones.asia.php
$d = date("Y-m-d");
$t = date("H:i:s");

```

```
// If values send by NodeMCU are not empty then insert into MySQL database table
```

```
if(!empty($_POST['sendval']) && !empty($_POST['sendval2'])) {  
    $val = $_POST['sendval'];  
    $val2 = $_POST['sendval2'];  
    $sql = "INSERT INTO nodemcu_table(val, val2, Date, Time) VALUES  
('".$val."','".$val2."','".$d."','".$t."')";  
  
    if ($conn->query($sql) === TRUE) {  
        echo "Values inserted in MySQL database table.";  
    }  
    else {  
        echo "Error: " . $sql . "<br>" . $conn->error;  
    }  
}
```

```
// Close MySQL connection
```

```
$conn->close();
```

```
?>
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

No somos expertos en php, pero logramos entender que con este código es posible crear un Query que lea el postData que se carga desde el Código de Arduino, y este se manda como una instrucción de tipo sql a la base de datos que se encuentra en el mismo lugar que este archivo dbwrite.php (dentro del HTDocs del servidor)

Conexión y Arranque del Servidor/Código

1. Por motivos de bloqueo de puertos de nuestra Institución, al menos para la realización de este proyecto dentro de sus instalaciones, requerimos inicializar una red Hotspot para la comunicación NodeMCU -> Internet
2. Encendido de XAMPP>>Apache/MySQL
3. Encendido del Cliente de playit.gg
4. Compilar el Código en Arduino

Si todo sale bien, el servidor ya debería estar recibiendo datos desde el NodeMCU, independientemente de que el NodeMCU esté conectado a la misma red o el Código de Arduino se ejecute en el mismo Ordenador que el Servidor.

Página Web

1- Para poder realizar la página web primero necesitamos lo que va a ser la parte principal, por lo que creamos un archivo index.html en donde será nuestra página inicial, empezamos con la clase = Hero dentro del Header que está dentro de la sección Body, dentro de la clase = Hero ingresamos una "clase = nav container" que va a ser el contenido de la parte superior de la página en donde encontraremos el logo.

Después creamos la sección `` el cual es un elemento que representa una lista de ítems en el cual se encuentra, inicio, acerca de y comparador. Dentro de la sección `` para poder representar los ítems los colocamos entre la sección `` el cual representa un elemento de lista.

Realizamos un sección `<section class= "hero_container container">` en el cual implementamos un título con `<h1 class= "hero_title"> Aprende a cuidar las plantas del hogar </h1>` , al igual que implementamos un párrafo de la manera siguiente e implementamos una clase dentro de ella para que a la hora de realizar el código en css sea más apropiado y más fácil de implementar `<p class= "hero_paragraph"> Elige de una vez ser un experto en el cuidado de las plantas del hogar</p>`. Después creamos un hipervínculo con la ayuda de `<a>` en el cual implementamos un `href=`, ya que funciona de la siguiente manera, si el elemento a tiene un atributo `href` este representará un hipervínculo, realizamos una clase dentro del elemento a `class="cta"` el cual va a funcionar como un botón, por lo que quedaría de la siguiente manera: `Aprende ahora`, colocamos el signo `"#"` para que no dirija a una liga, sino que esté ligada al botón que colocamos.

////////////////////////////////////

```

1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Plantas</title>
8      <link rel="shortcut icon" href="./images/favicon.png.png" type="image/x-icon">
9      <link rel="stylesheet" href="/css/estilos.css">
10 </head>
11 <body>
12
13     <header class="hero">
14         <nav class="nav container">
15
16             <div class="nav_logo">
17                 <h2 class="nav_title">Plantas Herbáceas</h2>
18
19             </div>
20
21             <ul class="nav_link nav_link--menu">
22
23                 <li class="nav_items">
24                     <a href="#" class="nav_links">Inicio </a>
25
26                 </li>
27                 <li class="nav_items">
28                     <a href="#" class="nav_links">Acerca de </a>
29
30                 </li>
31                 <li class="nav_items">
32                     <a class="Comparador" href="comparador.html" class="nav_links">Comparador </a>
33
34                 </li>
35
36                 
37

```

////////////////////////////////////

////////////////////////////////////

Una vez teniendo esto en cuenta creamos una `<section></section>` dentro de la `section` principal el cual le agregamos una “class” y la denominamos “`testimony_body` `testimony_body-show`” y le agregamos `data-id="1"`, este para diferenciar el orden de cómo van a ir pasando las imágenes y la información que se va a tener en `testimony`. Dentro del segundo `<section></section>` ingresamos un `<div></div>` el cual contendrá la clase = “`testimony_texts`” y dentro del `div` contendrá el elemento `<h2></h2>` que llevara una clase denominada “`subtitle`” en el cual vendrá el nombre del estudiante, posteriormente en la misma línea de código abriremos un elemento `` con la clase = “`testimony_course`” el cual contendrá la leyenda que se le quiera ingresar.

Ingresamos un párrafo con el elemento `<p></p>` el cual contendrá la clase = “`testimony texts`” esto con la finalidad de que a la hora de pasarlo a editarlo a `css` sea más práctico y fácil de editar debido a que se encuentra todos los párrafos de la sección `slide` dentro de una misma clase, por lo que a la hora de editarlo en `css` se editaran todos los párrafos que están dentro de esa sección `slide`, lo mismo pasa para cada una de las clases que están determinadas en cierta parte del código, para que sea más rápido y sencillo de editar cualquier aspecto de la página mediante el uso de `css`.

Repetimos el mismo procedimiento 2 veces más para que sea en total 3 de estas secciones, lo único que modificaremos será el “id” cambiándolo por 2 o por 3 para diferenciar la posición en la que estarán los slides y cambiamos la leyenda de la clase = “subtitle” junto con la leyenda del elemento span, ambos se cambian, por lo que nos quedaría un slide capaz de poder moverse de izquierda a derecha y en orden para cerrar con la otra imagen de la flecha pero del lado contrario.

////////////////////////////////////

```

117 <section class="testimony">
118   <div class="testimony_container container">
119     
120     <section class="testimony_body testimony_body--show" data-id="1">
121       <div class="testimony_texts">
122         <h2 class="subtitle"> Francisco Couttolenc Ortiz, <span class="testimony_course"> Desarrollo web</span></h2>
123         <p class="testimony_review"> Soy estudiante del Tecnológico de Monterrey Campus Querétaro, estudio la carrera de<br>
124           Ingeniería en Tecnología Computacional (ITC) y tengo 21 años.
125       </p>
126     </div>
127
128     <figure class="testimony_picture">
129       
130
131     </figure>
132
133   </section>
134
135   <section class="testimony_body" data-id="2">
136     <div class="testimony_texts">
137       <h2 class="subtitle"> Manuel Villalpando Linares, <span class="testimony_course"> Base de datos</span></h2>
138       <p class="testimony_review"> Soy estudiante del Tecnológico de Monterrey Campus Querétaro, estudio la carrera de<br>
139         Ingeniería en Tecnología Computacional (ITC) y tengo 19 años.
140     </p>
141
142     </div>
143
144     <figure class="testimony_picture">
145       
146
147     </figure>
148
149   </section>
150
151   <section class="testimony_body " data-id="3">
152     <div class="testimony_texts">
153       <h2 class="subtitle"> Ian Joab Padrón Corona, <span class="testimony_course"> Node MCU</span></h2>
154       <p class="testimony_review"> Soy estudiante del Tecnológico de Monterrey Campus Querétaro, estudio la carrera de<br>
155         Ingeniería en Tecnología Computacional (ITC) y tengo 19 años.
156     </p>
157
158     </div>
159
160     <figure class="testimony_picture">
161       
162
163     </figure>
164
165   </section>
166
167   
168 </div>
169 </section>

```

4- Pasamos a la siguiente `<section></section>` el cual estará dentro de nuestro main y de nuestro `<body></body>`, en el elemento section determinaremos la siguiente clase = “clases” el cual será el de nuestras plantas, dentro de `<section></section>` colocamos un elemento `<div></div>` el cual tendrá la clase = “clases_container container” el cual se desarrollara dentro de este container toda la información de las plantas. Ingresamos otro `<div></div>` dentro del div anterior y creamos una clase llamada = “clases_texts” el cual

contendrá el elemento “a” el cual representa un hipervínculo colocando de manera siguiente el elemento “name” y con una leyenda la que se quiera usar para poder direccionar al punto que se quiere alcanzar dentro de la página.

Implementamos un elemento `<h1></h1>` el cual es una sección del heading al cual le atribuimos la clase `= "subtitle"` en el cual se le ingresara la leyenda que quieras utilizar para hacer diferencia acerca del tema o en este caso el tipo de plantas que se quiere abordar en esta sección, después creamos un elemento `<h2></h2>` al cual también ingresamos la clase `= "subtitle"` el cual contendrá la leyenda del nombre de la planta. Ingresamos un elemento `<p></p>` el cual representa un párrafo en el cual determinaremos la clase `= "clases_paragraph"` en el cual contendrá información necesaria para que los usuarios puedan saber más acerca de esta planta.

Colocamos un elemento `<figure>` el cual será determinado con la clase = “clases_picture” el cual dentro del elemento “figure” ingresamos el elemento `` el cual representa una imagen el cual colocaremos un “src” que es un vínculo que va a estar vinculado con nuestra carpeta images y este vínculo lo atribuimos a la clase = “clases_img”.

Repetimos este proceso 10 veces más para que sean un total de 11 plantas las que se desplegaran en la página web, claramente cambiando la leyenda del tipo de plantas a la que pertenecen, cambiando la leyenda de que planta es y cambiando la información de cada planta. Al igual que cambiar las imágenes.

Cada 4 plantas colocaremos un un vínculo con el elemento <a> mediante un href = con un # de inicio y de siguiente el texto que escribimos tal cual en la sección de la página para que direcciona hacia esa sección y se le integrara también un botón mediante la clase = cta” y una leyenda que a la hora de presionar ahí por ejemplo “inicio” te direcciona al inicio de la página. Esto con la finalidad de que los usuarios puedan moverse de una manera rápida y sencilla a través de los tipos de plantas que hay que son Anuales, Bianuales y Perennes para que no tengan que estar visitando las 11 plantas que se presentan en la página web.

[illegible]


```

3  <!DOCTYPE html>
4  <html lang="es">
5
6  <head>
7      <meta charset="UTF-8">
8      <meta http-equiv="X-UA-Compatible" content="IE=edge">
9      <meta name="viewport" content="width=device-width, initial-scale=1.0">
10     <title>Comparador</title>
11     <script src="https://cdn.jsdelivr.net/npm/chart.js@latest/dist/Chart.min.js"> </script>
12     <link rel="stylesheet" href="/css/comparador.css">
13
14
15
16
17 </head>
18
19 <body>
20
21     <header class="hero">
22         <nav class="nav container">
23
24             <div class="nav_logo">
25                 <h2 class="nav_title">Comparador</h2>
26
27             </div>
28
29             <ul class="nav_link nav_link--menu">
30
31                 <li class="nav_items">
32                     <a href="index.html" class="nav_links">Inicio </a>
33
34                 </li>
35
36
37
38                 
39

```

```

39 |         |
40 |     </ul>
41 |
42 |     <div class="nav_menu">
43 |         
44 |     </div>
45 |
46 |
47 |
48 |
49 | </nav>
50 |
51 | <section class="hero_container container">
52 |     <h1 class="hero_title"> Comparacion de los diferentes tipos de plantas con la base de datos</h1>
53 |     <p class="hero_paragraph"> Aprende a diferenciar las temperaturas de las plantas .
54 |
55 |     </p>
56 |
57 |
58 | </section>
59 |
60 |
61 |
62 | </header>

```

////////////////////////////////////

Dentro de la sección que se encuentra dentro del main, colocaremos un elemento `<div></div>` con la clase = “about_main” el cual dentro de esta tendrá un elemento `<article></article>` el cual representa una composición completa o autónoma en un documento, página, aplicación o sitio, el cual implementaremos la clase = “about_icons” y dentro de este `<article></article>` ingresamos el elemento `` con el vínculo (src) que va a ir direccionada a la carpeta “images” esta imagen será nuestro icono que representa “dato” que esta de color azul y la denominaremos con la clase = “about_icon”, despues ingresaremos un elemento `<h3></h3>` el cual tendra una clase denominada = “about_title” y la leyenda “Anuales” la cual es el tipo de planta o la clasificacion de esos tipos de plantas e ingresaemos un boton con el hipervínculo href con el elemento “a” para que se diriga a esa sección de la página.

////////////////////////////////////

```

63 <comparador.html> <html> <body> <main> <section.container.about> <div.about_main> <article.about_icons
64 <main>
65 <section class="container about">
66 <a name="¿Que aprenderas en esta pagina web?"> </a>
67 <h2 class="subtitle">¿Que aprenderas en esta seccion? </h2>
68 <p class="about_paragraph">Aprenderas a diferenciar las temperaturas de los diferentes tipos de plantas,
69 mediante la base de datos <br> en donde se compararan
70 la temperatura ideal y a la que se encuentra.
71
72
73 </p>
74
75 <div class="about_main">
76 <article class="about_icons">
77 
78 <h3 class="about_title">Anuales </h3>
79 <a href="#Tipos de plantas anuales" class="cta"> Temperatura de plantas anuales</a>
80
81
82
83
84
85 </article>
86
87 <article class="about_icons">
88 
89 <h3 class="about_title">Bianuales </h3>
90 <a href="#Tipos de plantas bianuales" class="cta"> Temperatura de plantas bianuales </a>
91
92
93
94
95
96 </article>
97
98 <article class="about_icons">
99 

```



```

    ],
    options: {
      scales: {
        yAxes: [{
          ticks: {
            beginAtZero: true
          }
        }]
      }
    }
  }
}


```

////////////////////////////////////


9- Creamos una carpeta llamada css, en donde irán los archivos comparador.css y estilos.css, los cuales son los 2 css de nuestras 2 paginas que tenemos. Dentro de estos archivos nosotros podemos configurar el aspecto de nuestra página.

estilos.css (index.html):


```
////////////////////////////////////  
css > # estilos.css > %$ .hero::before  
1  @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600;700&display=swap');  
2  
3  :root{  
4    --padding-container: 100px 0;  
5    --color-title: #001a49;  
6  }  
7  
8  body{  
9    font-family: 'Poppins', sans-serif;  
10 }  
11  
12  
13 .container{  
14   width: 90%;  
15   max-width: 1200px;  
16   margin: 0 auto;  
17   overflow: hidden;  
18   padding: var(--padding-container)  
19 }  
20  
21 .hero{  
22   width: 100%;  
23   height: 100vh;  
24   min-height: 600px;  
25   max-height: 800px;  
26   position: relative;  
27   display: grid;  
28   grid-template-rows: 100px 1fr;  
29   color: #fff;  
30  
31 }  
32
```


css > # estilos.css >  .hero:before

```
32
33 .hero::before{
34   content:"";
35   position:absolute;
36   top:0;
37   left:0;
38   width: 100%;
39   height: 100%;
40   background-image: linear-gradient(180deg, □ #0000008c 0%, □ #0000008c 100%), url(../images/image\ 1.png);
41   background-size: cover;
42   clip-path: polygon(0 0, 100% 0, 100% 80%, 50% 100%, 0 80%);
43   z-index: -1;
44
45
46 }
47
48 /* INICIA NAV */
49
50 .nav{
51   --padding-container:0;
52   height: 100%;
53   display: flex;
54   align-items: center;
55
56 }
57
58 .nav_title{
59   font-weight: 300;
60 }
61
62 /* MENU DE NAVEGACION */
63
64
```


css > # estilos.css >  .hero:before

```
65 .nav_link{
66   margin-left: auto;
67   padding: 0;
68   display: grid;
69   grid-auto-flow: column;
70   grid-auto-columns: max-content;
71   gap: 2em;
72 }
73
74 /* PARA QUITAR EL ESTILO DE LISTAS DE NAV LINK MENU ( INICIO, ACERCA DE, CONTACTO)*/
75
76 .nav_items{
77   list-style: none;
78 }
79
80
81 .nav_links{
82   color: □ #fff;
83   text-decoration: none;
84
85 }
86
87 /* PARA DESAPARECER EL ITEM MENU*/
88
89 .nav_menu{
90   margin-left: auto;
91   cursor: pointer;
92   display: none;
93 }
94
95 .nav_img{
96   display: block;
97   width: 30px;
98 }
```


```
css > # estilos.css >  .hero::before
100  /* PARA DESAPARECER EL ITEM CLOSE*/
101
102  .nav_close{
103  |    display: var(--show, none);
104  |  }
105
106  /* TERMINA NAV */
107
108  /* HERO_CONTAINER*/
109
110  .hero_container{
111  |    max-width: 800px;
112  |    --padding-container:0;
113  |    display: grid;
114  |    grid-auto-rows: max-content;
115  |    align-content: center;
116  |    gap: 1em;
117  |    padding-bottom: 100px;
118  |    text-align: center;
119  |  }
120
121  .hero_title{
122  |    font-size: 3rem;
123  |  }
124
125  .hero_paragraphh{
126  |    margin-bottom: 20px;
127  |  }
128
129
```

css > # estilos.css >  .hero::before


```
130  /* BOTON INICIO*/
131
132  .cta{
133      display:inline-block;
134      background-color: #2091F9;
135      justify-self: center;
136      color: #fff;
137      text-decoration: none;
138      padding: 13px 30px;
139      border-radius: 32px;
140  }
141
142
143  /* ABOUT */
144
145  .about{
146      text-align: center;
147  }
148
149  .subtitle{
150      color: var(--color-title);
151      font-size: 2rem;
152      margin-bottom: 25px;
153  }
154
155  .about_paragraph{
156      line-height: 1.7;
157  }
158
```



css > # estilos.css >  .hero::before


```
159 .about_main{
160     padding-top: 80px;
161     display: grid;
162     width: 90%;
163     margin: 0 auto;
164     gap: 1em;
165     overflow: hidden;
166     grid-template-columns: repeat(auto-fit, minmax(260px, auto));
167 }
168
169 .about_icons{
170     display: grid;
171     gap: 1em;
172     justify-items: center;
173     width: 260px;
174     overflow: hidden;
175     margin: 0 auto;
176 }
177
178
179 .about_icon{
180     width: 40px;
181 }
182
183 /* TESTIMONY */
184
185 .testimony{
186     background-color:  #e5e5f7;
187 }
188
189
190 .testimony_container{
191     display: grid;
192     grid-template-columns: 50px 1fr 50px;
193     align-items: center;
194 }
```

css > # estilos.css >  .hero::before

```
196 .testimony_body{
197     display: grid;
198     grid-template-columns: 1fr max-content;
199     justify-content: space-between ;
200     align-items: center;
201     gap: 2em;
202     grid-column: 2/3;
203     grid-row: 1/2;
204     opacity: 0;
205     pointer-events: none;
206
207 }
208
209 .testimony_body--show{
210     pointer-events: unset;
211     opacity: 1;
212     transition: opacity 1.5s ease-in-out;
213 }
214
215 .testimony_img{
216     width: 250px;
217     height: 250px;
218     border-radius: 50%;
219     object-fit: cover;
220     object-position: 50% 30%;
221 }
222
223 .testimony_texts{
224     max-width: 700px;
225 }
```

css > # estilos.css >  .hero::before

```
227 .testimony_course{
228     background-color:  royalblue;
229     color:  #fff;
230     display: inline-block;
231     padding: 5px;
232
233
234 }
235
236 .testimony_arrow{
237     width: 90%;
238     cursor: pointer;
239 }
240
241 /* CLASES* (PLANTA GIRASOL) */
242
243
244
245 .clases_container{
246     display: grid;
247     grid-template-columns: 1fr 1fr;
248     gap: 1em;
249     align-items: center;
250 }
251
252 .clases_picture{
253     max-width: 500px;
254 }
255
256 .clases_paragraph{
257     line-height: 1.7;
258     margin-bottom: 15px;
259 }
260
```

```
css > # estilos.css >  .hero::before
260
261 .clases_img{
262     width: 100%;
263     display: block;
264 }
265
266 /* CLASES 2 (CALÉNDULA)*/
267
268 .clases2{
269     background-color: #e5e5f7;
270     background-image: radial-gradient(#444cf7 0.5px, transparent 0.5px), radial-gradient(#444cf7 0.5px, #e5e5f7 0.5px);
271     background-size: 20px 20px;
272     background-position: 0 0,10px 10px;
273     overflow: hidden;
274 }
275
276 .clases_container2{
277     display: grid;
278     grid-template-columns: 1fr 1fr;
279     gap: 1em;
280     align-items: center;
281 }
282
283
284 .clases_picture2{
285     max-width: 500px;
286 }
287
288 .clases_paragraph2{
289     line-height: 1.7;
290     margin-bottom: 15px;
291 }
292
293 .clases_img2{
294     width: 100%;
295     display: block;
296 }
```

////////////////////////////////////

Comparador.css (comparador.html):

```
css > # comparador.css > ...
1  @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600;700&display=swap');
2
3  :root{
4      --padding-container: 100px 0;
5      --color-title: #001a49;
6  }
7
8  body{
9      font-family: 'Poppins', sans-serif;
10
11
12 }
13
14 .container{
15     width: 90%;
16     max-width: 1200px;
17     margin: 0 auto;
18     overflow: hidden;
19     padding: var(--padding-container)
20 }
21
22 .hero{
23     width: 100%;
24     height: 100vh;
25     min-height: 600px;
26     max-height: 800px;
27     position: relative;
28     display: grid;
29     grid-template-rows: 100px 1fr;
30     color: #fff;
31
32 }
```



```
css > # comparador.css > ...
34 .hero::before{
35     content:"";
36     position:absolute;
37     top:0;
38     left:0;
39     width: 100%;
40     height: 100%;
41     background-image: linear-gradient(180deg, □ #0000008c 0%, □ #0000008c 100%), url(../images/growtika-developer-marketing-agency-ZfVyuV8l7wU-unspl
42     background-size: cover;
43     clip-path: polygon(0 0, 100% 0, 100% 80%, 50% 100%, 0 80%);
44     z-index: -1;
45
46
47 }
48
49 /* INICIA NAV */
50
51 .nav{
52     --padding-container:0;
53     height: 100%;
54     display: flex;
55     align-items: center;
56
57 }
58
59 .nav_title{
60     font-weight: 300;
61 }
62
63 /* MENU DE NAVEGACION */
64
65
```

```
66 .nav_link{
67     margin-left: auto;
68     padding: 0;
69     display: grid;
70     grid-auto-flow: column;
71     grid-auto-columns: max-content;
72     gap: 2em;
73 }
74
75 /* PARA QUITAR EL ESTILO DE LISTAS DE NAV LINK MENU ( INICIO, ACERCA DE, CONTACT
76
77 .nav_items{
78     list-style: none;
79 }
80
81
82 .nav_links{
83     color: #fff;
84     text-decoration: none;
85 }
86
87
88 /* PARA DESAPARECER EL ITEM MENU*/
89
90 .nav_menu{
91     margin-left: auto;
92     cursor: pointer;
93     display: none;
94 }
95
96 .nav_img{
97     display: block;
98     width: 30px;
99 }
100
```

```
101  /* PARA DESAPARECER EL ITEM CLOSE*/
102
103  .nav_close{
104  |    display: var(--show, none);
105  |  }
106
107  /* TERMINA NAV */
108
109  /* HERO_CONTAINER*/
110
111  .hero_container{
112  |    max-width: 800px;
113  |    --padding-container:0;
114  |    display: grid;
115  |    grid-auto-rows: max-content;
116  |    align-content: center;
117  |    gap: 1em;
118  |    padding-bottom: 100px;
119  |    text-align: center;
120  |  }
121
122  .hero_title{
123  |    font-size: 3rem;
124  |  }
125
126  .hero_paragraph{
127  |    margin-bottom: 20px;
128  |  }
129
130
131  /* BOTON INICIO*/
132
```

```
133 .cta{
134     display:inline-block;
135     background-color: #2091F9;
136     justify-self: center;
137     color: #fff;
138     text-decoration: none;
139     padding: 13px 30px;
140     border-radius: 32px;
141 }
142
143
144 /* ABOUT */
145
146 .about{
147     text-align: center;
148 }
149
150 .subtitle{
151     color: var(--color-title);
152     font-size: 2rem;
153     margin-bottom: 25px;
154 }
155
156 .about_paragraph{
157     line-height: 1.7;
158 }
159
160 .about_main{
161     padding-top: 80px;
162     display: grid;
163     width: 90%;
164     margin: 0 auto;
165     gap: 1em;
166     overflow: hidden;
167     grid-template-columns: repeat(auto-fit, minmax(260px, auto));
168 }
```

```
170 .about_icons{
171     display: grid;
172     gap: 1em;
173     justify-items: center;
174     width: 260px;
175     overflow: hidden;
176     margin: 0 auto;
177
178 }
179
180 .about_icon{
181     width: 40px;
182 }
183
184 /* MAIN*/
185
186 .body_main{
187     font-family: Verdana, Geneva, Tahoma, sans-serif;
188     height: 100vh;
189     display: flex;
190     align-items: center;
191     justify-content: center;
192 }
193
194 h1{
195     margin-bottom: 2rem;
196     text-align: center;
197 }
198
199 .content{
200     width: 40%;
201     margin: 0 auto;
202 }
203
```

```
205 canvas{
206     max-width: 100%;
207 }
208
209
210
```

////////////////////////////////////

Conexión MySQL -> Página web mediante php

1. Para poder enlazar la base de datos a nuestra página web se necesita un método, en este caso utilizaremos php, ya que nos permite utilizar las variables como cualquier otra variable de programación dentro del código, dándonos una amplia posibilidad de acciones con la misma, desde comparar hasta utilizarla para suma o restar algo.
2. Para poder enlazar ambas cosas con php primero que nada se necesita iniciar el php dentro de la página que queremos que enlace de nuestro html, por lo que tendremos que agregar el siguiente código en la parte superior de la página en la que estaremos trabajando con nuestra base de datos:

////////////////////////////////////

<?php

```
$host = 'iot.niubycraft2.playit.gg';
$db = 'microsoft2';
$puerto = '6286';
$usuario = 'root';
$password = "";
```

```
$link = mysqli_connect($host,$usuario,$password,$db);
```

?

////////////////////

En el código se tiene hasta afuera del php, “<?php” “?” con el código php en medio de estos para abrir y cerrar indicando desde donde comienza hasta donde acaba. Después inicializamos variables de host, db, puerto, etcétera, esto para que el php

En la última línea antes de cerrar el código se define la variable de link, uniendo las demás variables dentro de la función `mysqli_connect()`, la cual es una función predefinida que le permite a php inicializar con todos esos valores que ingresamos anteriormente.

- Ejemplo del texto anterior:

```
$query = "SELECT * FROM planta";
$querytempreal = "SELECT *
FROM medidas
ORDER by Conteo DESC
LIMIT 1";

$resultado = mysqli_query($link,$query);
$tempreal = mysqli_query($link,$querytempreal);

while($fila = mysqli_fetch_array($tempreal)) {

    $dia = $fila['dia'];
    $nombre = $fila['NombreP'];
    $temperatura = $fila['Temp_actual'];
    $shumedad = $fila['Humedad_actual'];
```

```

        if (abs($tempreal-$resultado) > 5) {
            $tipo = 'fuera';
            echo 'Fuera';
        }
        echo
        '<tr><td>'.$dia.'</td><td>'.$nombre.'</td><td>'.$temperatura.'</td><td>'.$humedad.'<
        /td></tr>';
    }
    mysqli_close($link);
?>
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

Creación de gráficas con actualización en tiempo real a partir de la base de datos

1. Para crear la tabla de la base de datos es primordial recopilar los datos con los que se estará trabajando y meterlos en listas ordenadas por nombre, esto para un mejor seguimiento de los datos o en caso de que sea necesario modificar alguno.
2. Esto se hace asignando los datos de las columnas a cada una de las variables con un ciclo para que vaya pasando por todas y cada una de ellas y pase si es que existe un campo extra.
3. Además de eso, se genera un query para el sql, en el que indique que se deben seleccionar únicamente un determinado número de registros, los cuales se representarán como una gráfica de barras en nuestra base de datos.
4. Posteriormente con ayuda de php y json, se realiza la selección e impresión de las gráficas a partir de la base de datos en la sección que deseemos de nuestra página web.
 - a. Nota: Las gráficas se añaden de manera progresiva y se puede editar el número de registros que se muestra en el "DESC LIMIT".

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
<?php

// Check if connection established successfully
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
} else {
    echo "Finding connection... Please wait...<br>";
    sleep(1);
    echo "Connected to database. <br>";
}
$all = "SELECT * FROM medidas ORDER BY Conteo DESC LIMIT 25";
$resultall = mysqli_query($conn, $all);
$chart_data = "";

```



```

while ($row = mysqli_fetch_array($resultall)) {
    $productname[] = $row['Conteo'];
    $sales[] = $row['Temp_actual'];
    $goldemexico[] = $row['Humed_actual'];
    $horagol[] = $row['Hora'];
}

// Select values from MySQL database table

$sql = "SELECT * FROM medidas ORDER BY Conteo DESC LIMIT 1";
// "SELECT Conteo, Dia, Hora, Temp_actual, Humed_actual FROM medidas";

$result = $conn->query($sql);

echo "<center>";

if ($result->num_rows > 0) {

    // Output data of each row
    while ($row = $result->fetch_assoc()) {

        echo "<strong> # Reg.:</strong> " . $row["Conteo"] . " &nbsp;
<strong>Dia:</strong> " . $row["Dia"] . " &nbsp;
<strong>Hora:</strong> " . $row["Hora"] . " &nbsp; <strong>Temp en °C:</strong> " .
$row["Temp_actual"] . " &nbsp;
<strong>Humedad en %:</strong>" . $row["Humed_actual"] . "<p>";
    }
} else {
    echo "0 results";
}

echo "</center>";

$conn->close();

?>

<section>
    <div class="row">
        <div class="col-md-8 offset-md-2">
            <div class="card">
                <div class="card-header bg">
                    <h1>Temperaturas </h1>
                </div>
                <div class="card-body">
                    <canvas id="chartjs_bar"></canvas>
                </div>
            </div>
        </div>
    </div>
</section>

```

```

        </div>
    </div>
</div>
<script src="assets/js/jquery.min.js"></script>
<script src="assets/bootstrap/js/bootstrap.min.js"></script>
<script
src="//cdnjs.cloudflare.com/ajax/libs/Chart.js/2.4.0/Chart.min.js"></script>
<script type="text/javascript">
    var ctx = document.getElementById("chartjs_bar").getContext('2d');
    var myChart = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: <?php echo json_encode($productname); ?>,
            datasets: [{
                backgroundColor: [
                    "#ffd322",
                    "#5945fd",
                    "#25d5f2",
                    "#2ec551",
                    "#ff344e",
                ],
                data: <?php echo json_encode($sales); ?>
            }]
        },
        options: {
            legend: {
                display: true,
                position: 'bottom',
                labels: {

                    fontColor: '#71748d',
                    fontFamily: 'Circular Std Book',
                    fontSize: 0,
                }
            },
        }
    });
</script>
</section>
<br>
</body>

<body>
<section>
<div class="row">
<div class="col-md-8 offset-md-2">
<div class="card2">
<div class="card-header bg">

```

```

        <h1>Humedad por tiempo</h1>
    </div>
    <div class="card-body">
        <canvas id="chartjs_bar2"></canvas>
    </div>
</div>
</div>
</div>
<script src="assets/js/jquery.min.js"></script>
<script src="assets/bootstrap/js/bootstrap.min.js"></script>
<script
src="//cdnjs.cloudflare.com/ajax/libs/Chart.js/2.4.0/Chart.min.js"></script>
<script type="text/javascript">
    var ctx = document.getElementById("chartjs_bar2").getContext('2d');
    var myChart = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: <?php echo json_encode($horagol); ?>,
            datasets: [{
                backgroundColor: [
                    "#fd322",
                    "#5945fd",
                    "#25d5f2",
                    "#2ec551",
                    "#ff344e",
                ],
                data: <?php echo json_encode($goldemexico); ?>
            }]
        },
        options: {
            legend: {
                display: true,
                position: 'bottom',
                labels: {
                    fontColor: '#71748d',
                    fontFamily: 'Circular Std Book',
                    fontSize: 0,
                }
            },
        }
    });
</script>
</section>

</body>

</html>

```

////////////////////////////////////