

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL  
FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN  
LABORATORIO DE DISEÑO DE SISTEMAS DIGITALES (EYAG1007)  
PARALELO 103**

**AVANCE N°3 - PROYECTO**

**Docente:** Nathaly Simuy Sánchez Chan

**Integrantes:**

- Juan Pablo Cadena Aguilar
- Steven Isaac Santillán Padilla

El reporte debe ser tipo guía de laboratorio indicando paso a paso lo que han realizado hasta el momento. El avance corresponde al funcionamiento de los procesadores (aunque sea por separado). Donde deben presentar la ejecución de los códigos en los procesadores utilizados y la documentación técnica al 70%.

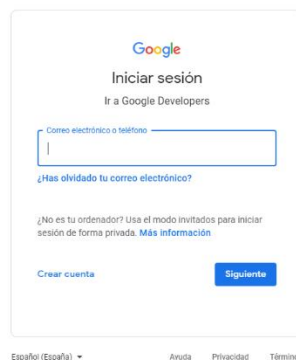
## Parte necesaria para el envío de datos desde los sensores:

### Firestore:

Se ingresa en un explorador web la siguiente dirección: <https://firebase.google.com/> y se mostrará una página como la adjunta.



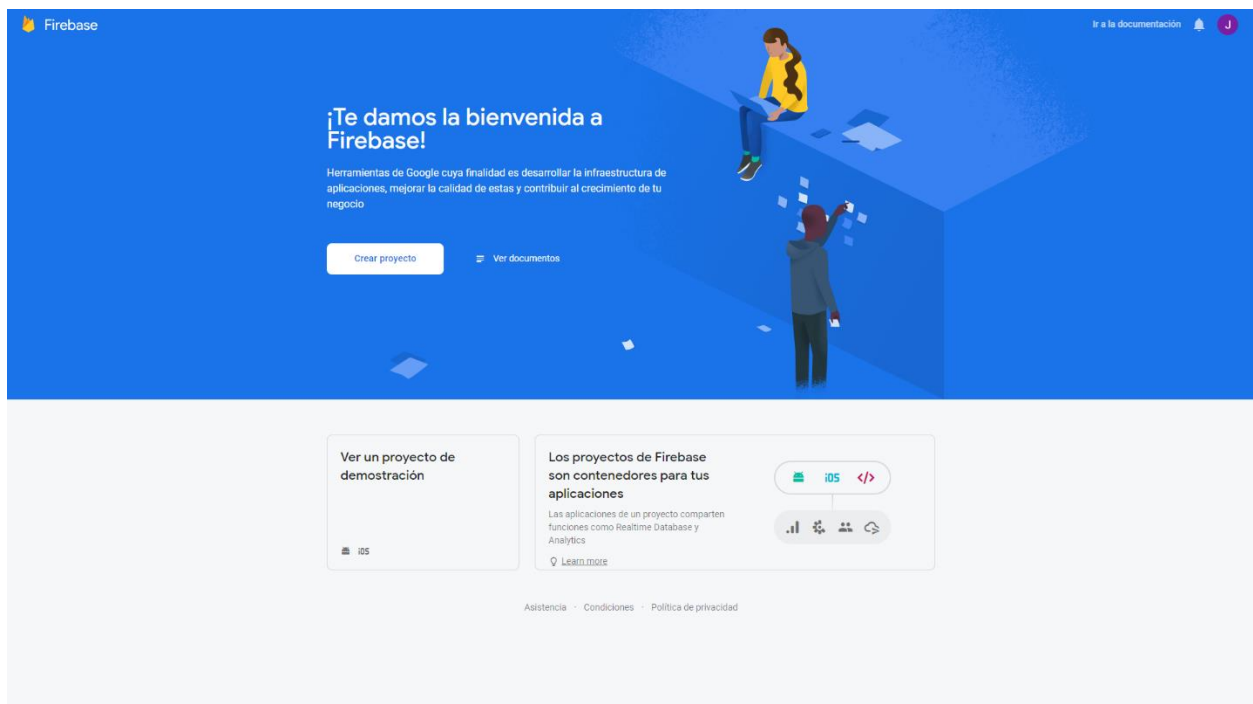
Se dirige al apartado superior donde dice “Acceder” donde podrá iniciar sesión con una cuenta con dominio de Google, como Gmail o Fiecc Espol.



Trás un exitoso inicio de sesión, se dirige nuevamente al apartado superior, pero elegirá “Ir a la consola” como en la imagen.



Luego se cargará una nueva pantalla de bienvenida.



A continuación, seleccionará el botón “Crear proyecto” y será direccionado a un conjunto de pasos para la configuración inicial.

X Crear proyecto(paso 1 de 3)

Empieza por ponerle un nombre al proyecto

Introduce el nombre de tu proyecto

my-awesome-project-id

☐ Acepto los [términos de Firebase](#).

Continuar

An illustration on the right side of the screen shows two people, a man and a woman, standing at a desk. The man is holding a smartphone and looking at it, while the woman is sitting at the desk, typing on a laptop. There is a blue cup on the desk next to the laptop. The background is a solid blue color.

Después de escribir el nombre y aceptar los términos de Firebase mediante la activación de su respectiva casilla, presione el botón continuar

X Crear proyecto(paso 1 de 3)

Empieza por ponerle un nombre al proyecto

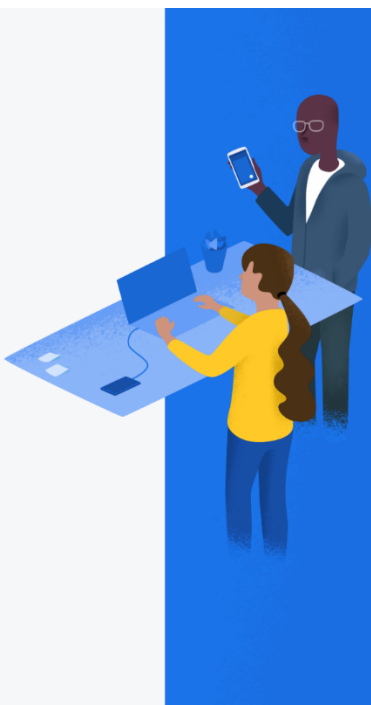
Nombre del proyecto

DisenoSistemasDigitales

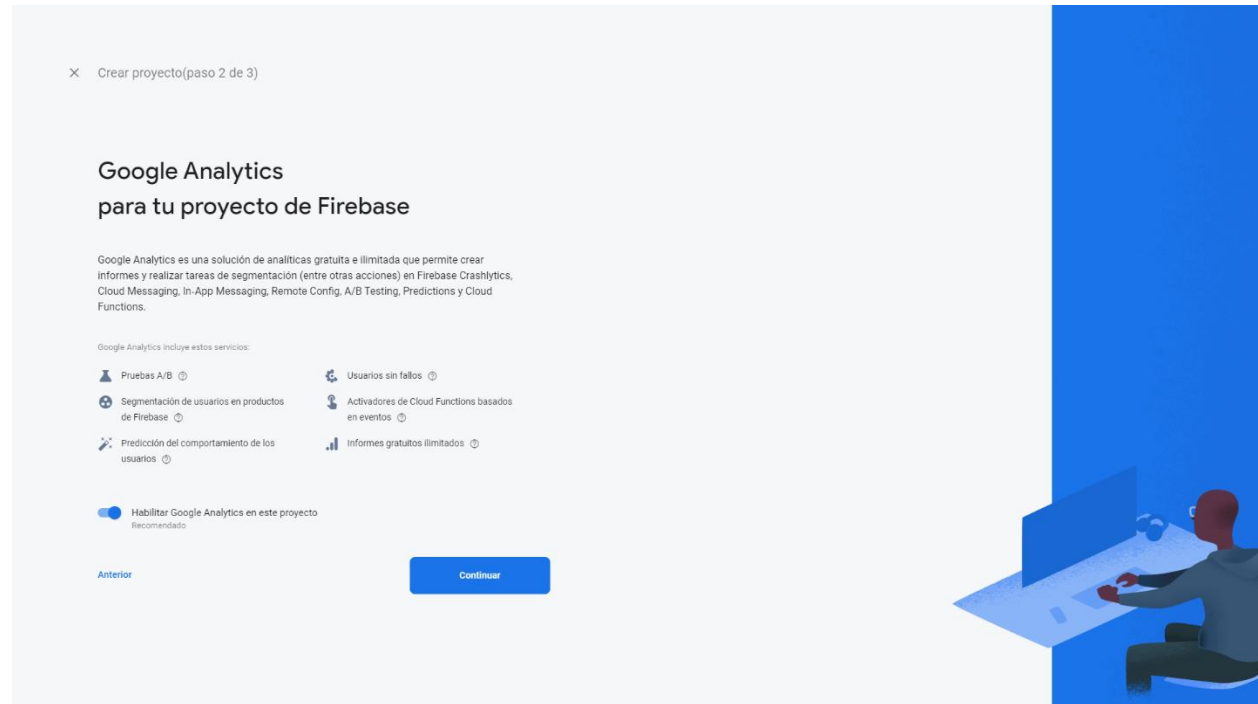
/disenosistemasdigitales

☒ Acepto los [términos de Firebase](#).

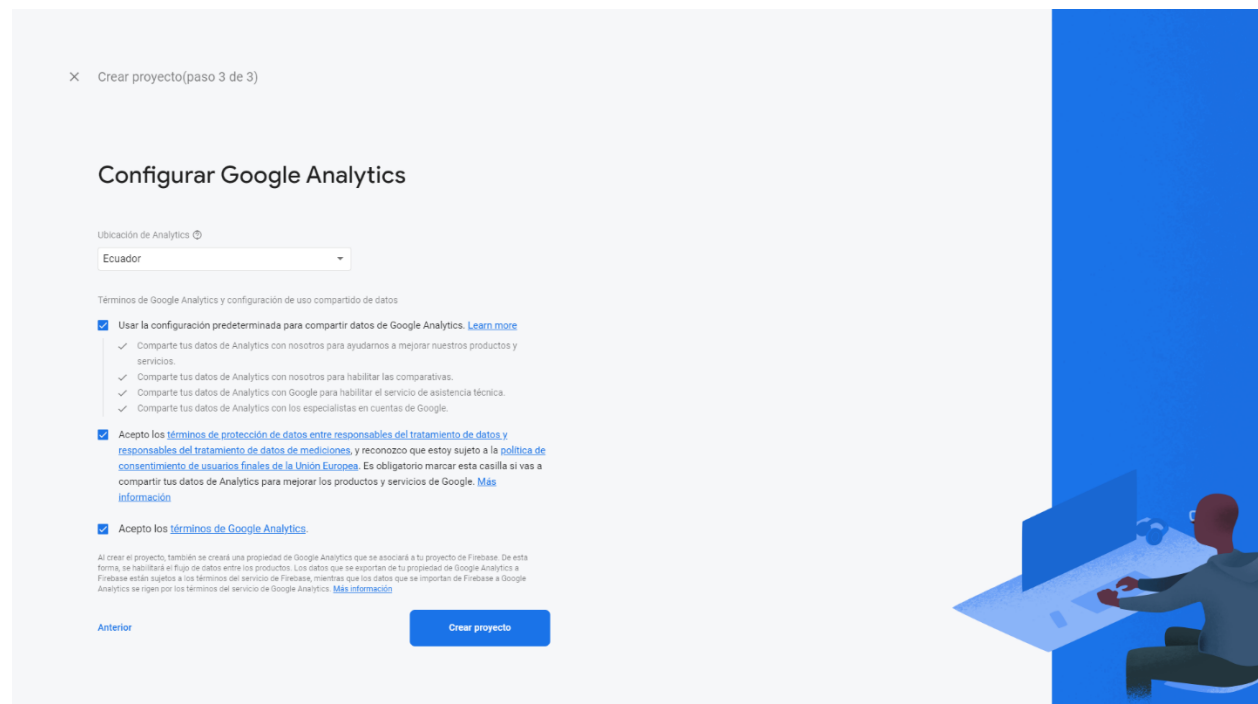
Continuar

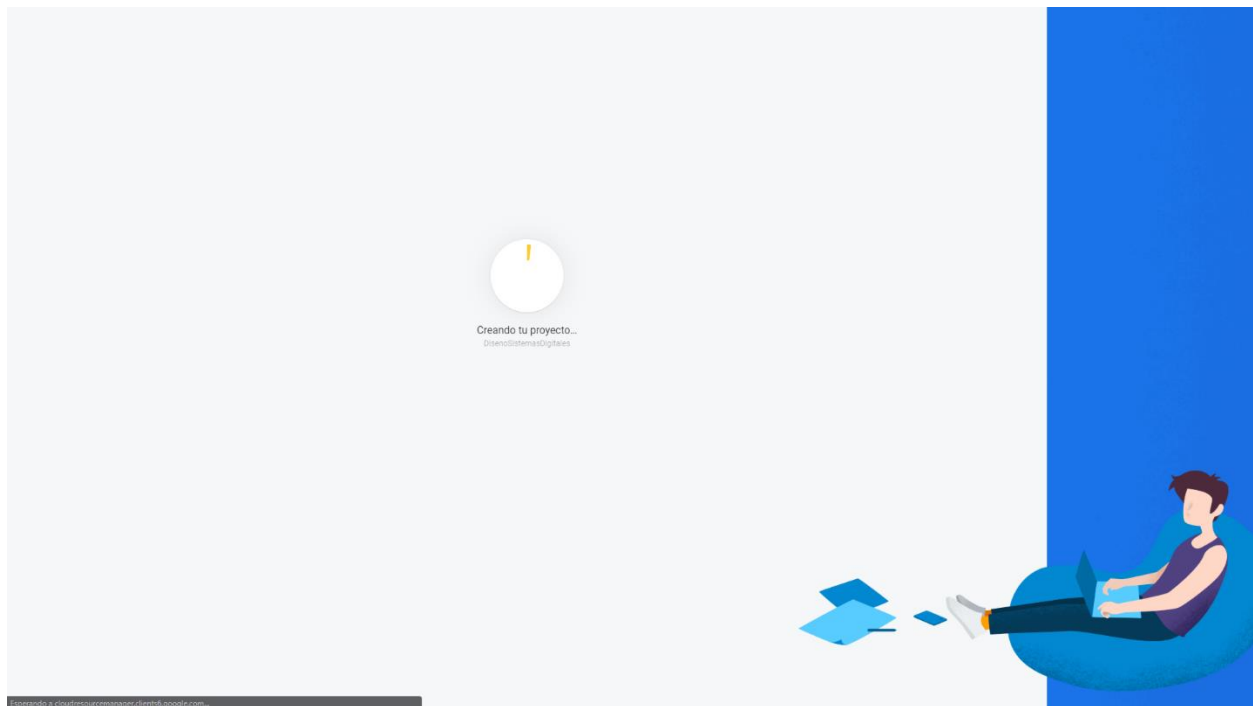
An illustration on the right side of the screen shows two people, a man and a woman, standing at a desk. The man is holding a smartphone and looking at it, while the woman is sitting at the desk, typing on a laptop. There is a blue cup on the desk next to the laptop. The background is a solid blue color.

***Nota: El nombre es el identificador único global utilizado en la URL, no se puede cambiar y solo puede contener letras, números, espacios y estos caracteres: -!'"***  
El siguiente paso solo consiste en presionar el botón “Continuar” y queda a disposición si se desea activar o no Google Analytics.

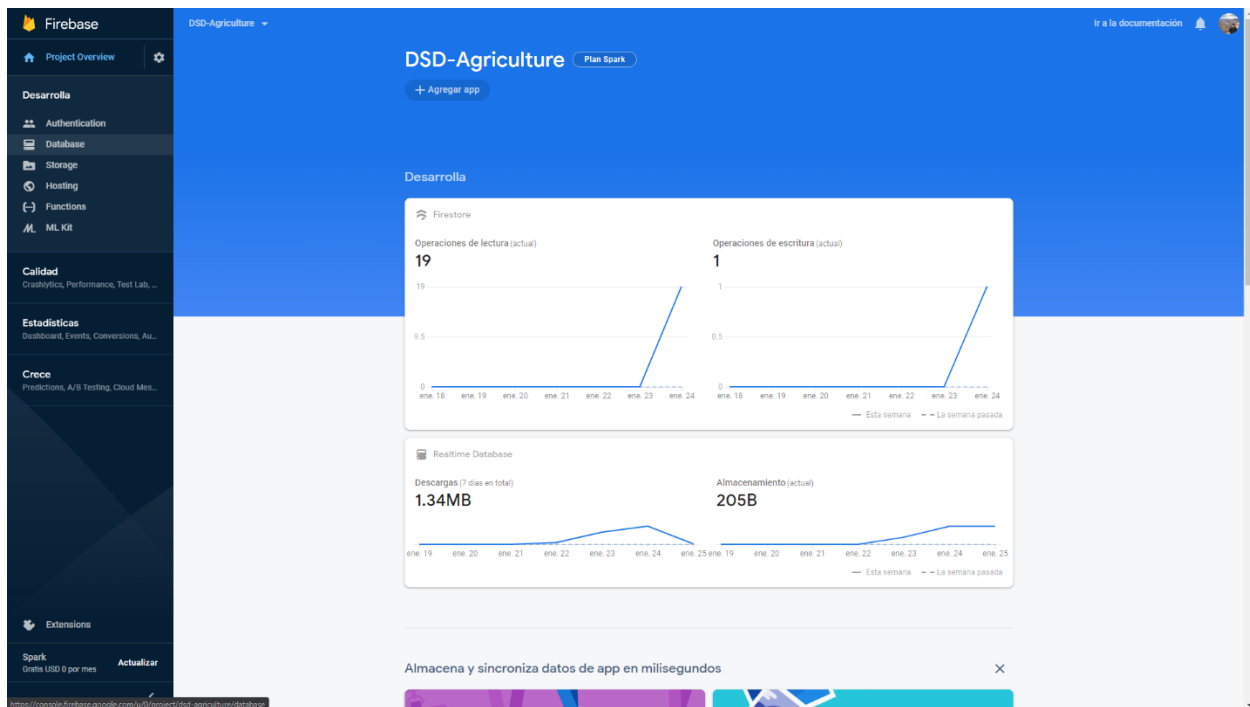


Se aceptan términos de privacidad y finalmente se crea el proyecto.

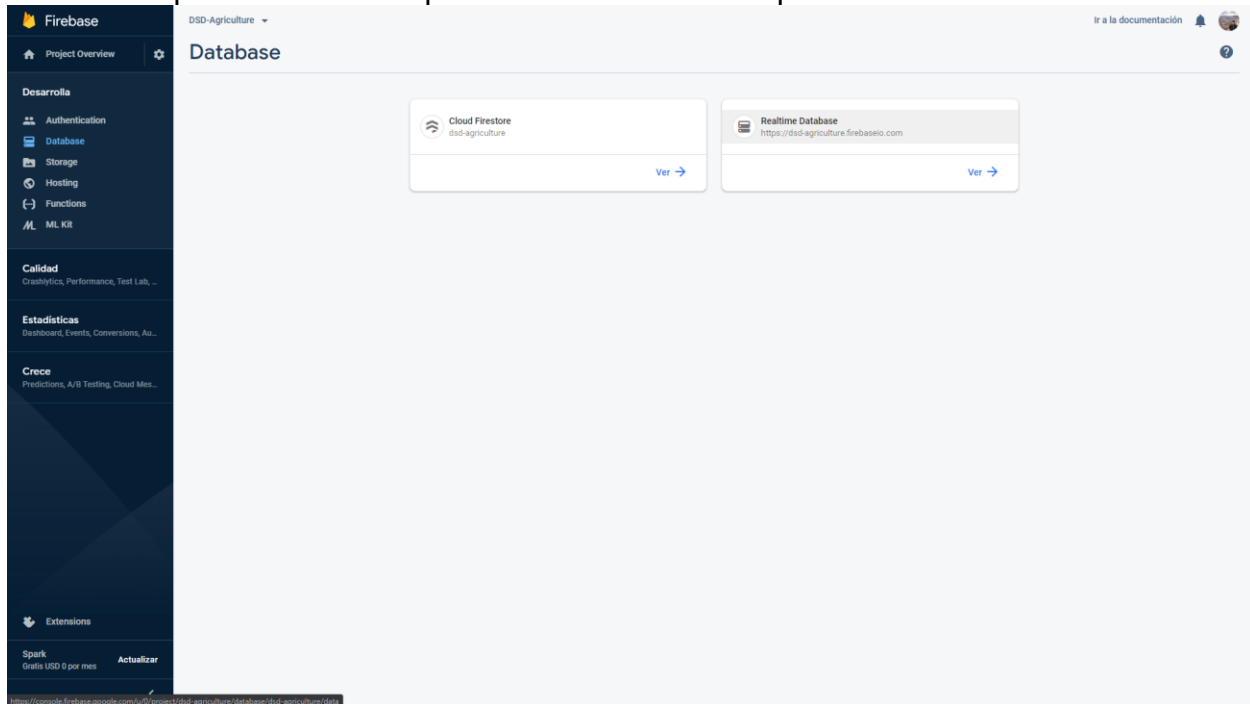




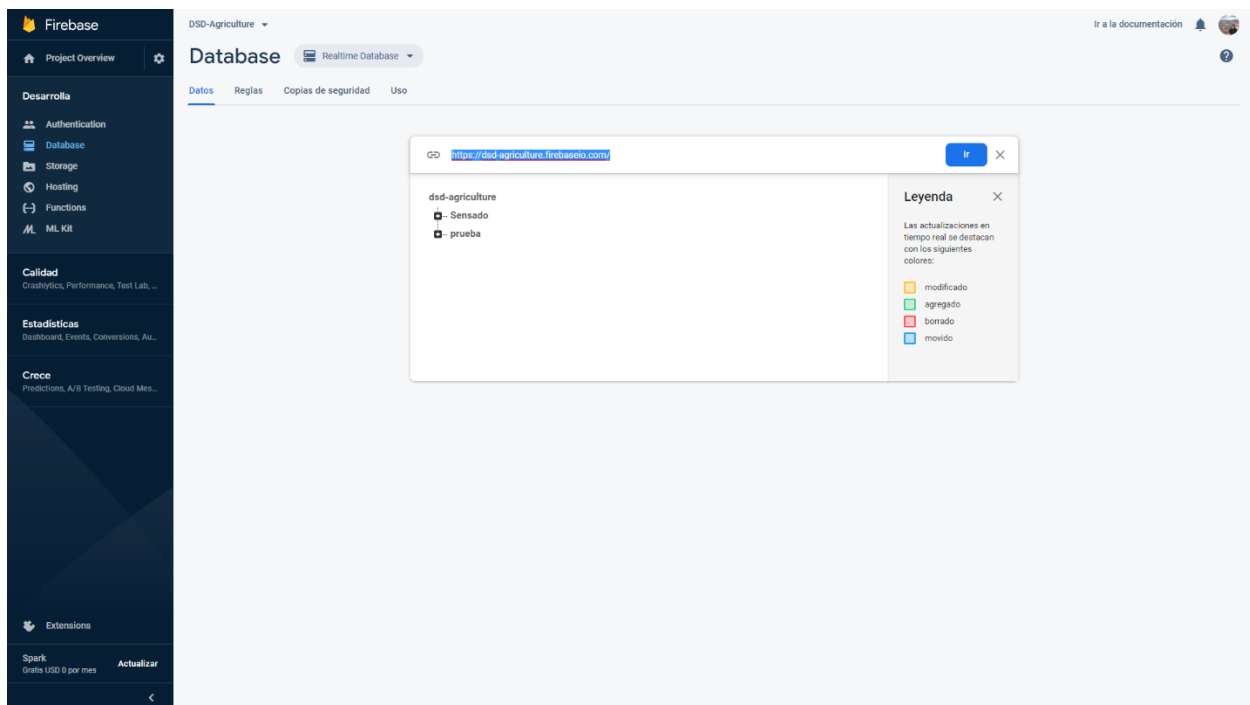
Con el proyecto generado se apreciará el potencial de la plataforma de desarrollo de aplicaciones web y móviles de Google.



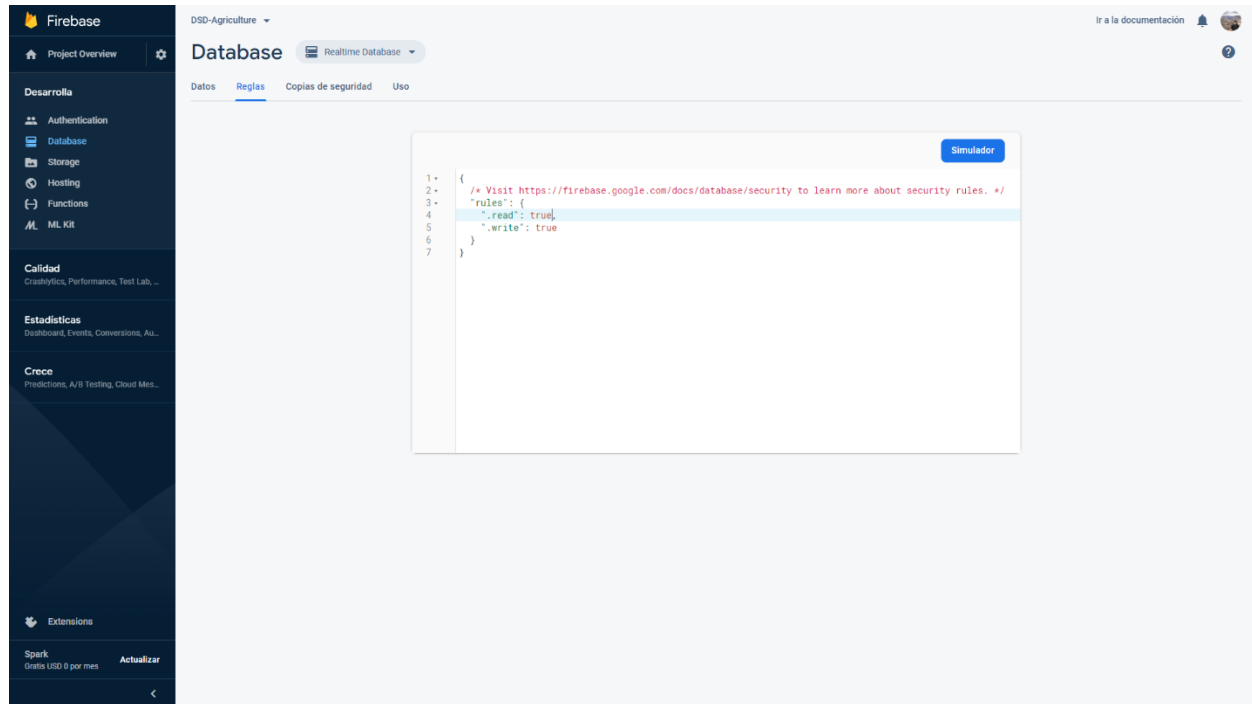
Se dirige al panel izquierdo en el apartado “Desarrolla” y se elige “Database” para luego mostrar el tipo de bases a disposición: Nube o en tiempo real.



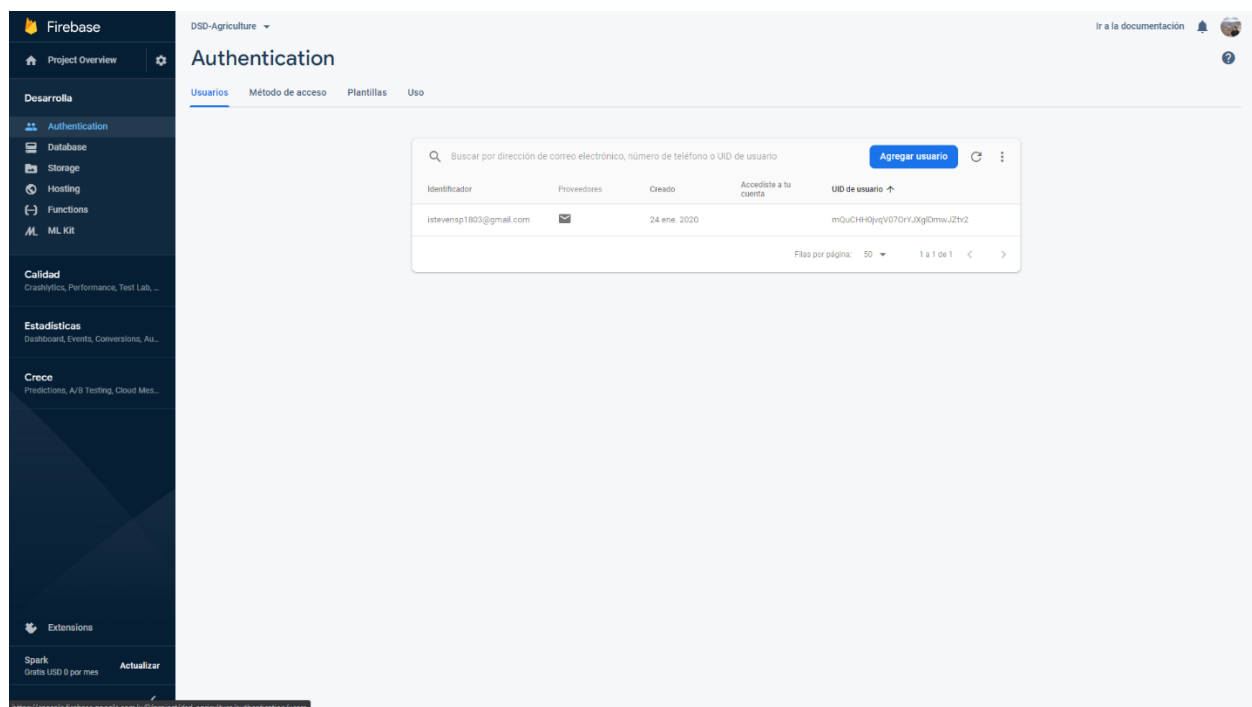
Luego de elegir la última, seleccionamos la dirección que se encuentra en un cuadro central del explorador, la cual será nuestro anfitrión para acceder a la base de datos.



Después se elige del menú de la base, la opción de “Reglas”, donde se deberá cambiar el valor de las claves “read” y “write” de ‘false’ a ‘true’ para así poder trabajar sobre la plataforma.

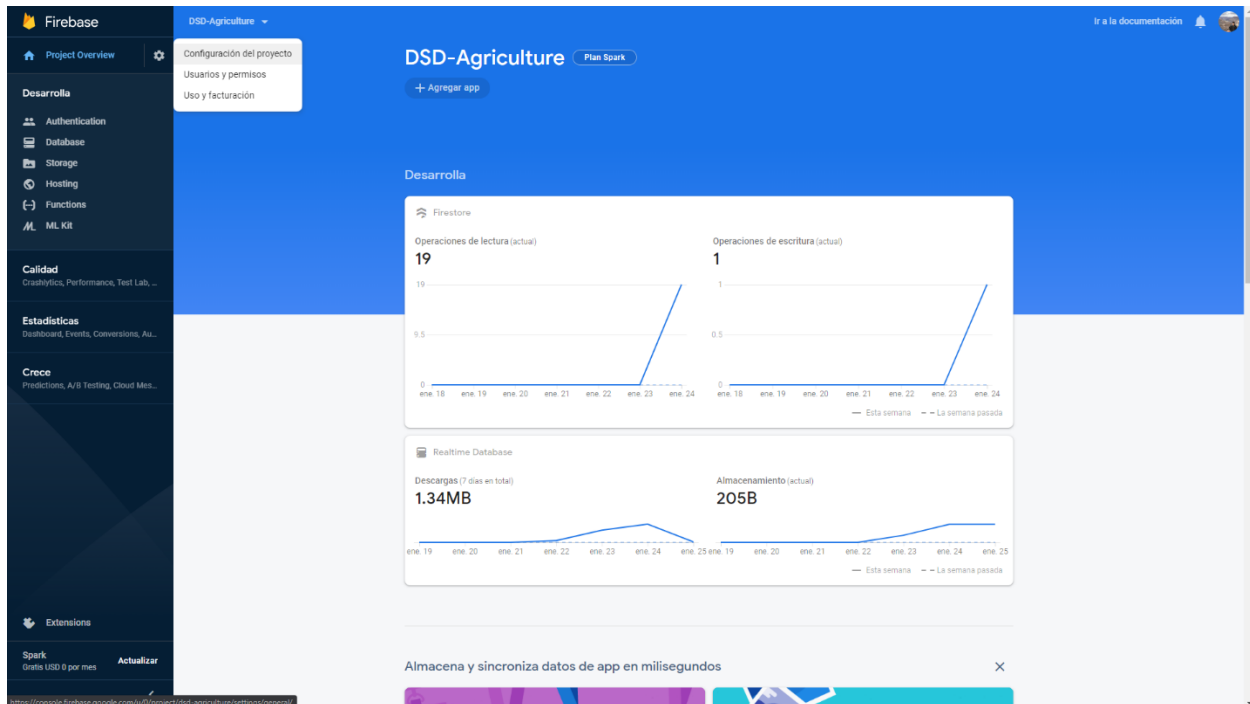


Ahora se debe acceder a la sección de Autenticación en el menú lateral y se podrá agregar usuarios con una respectiva clave.

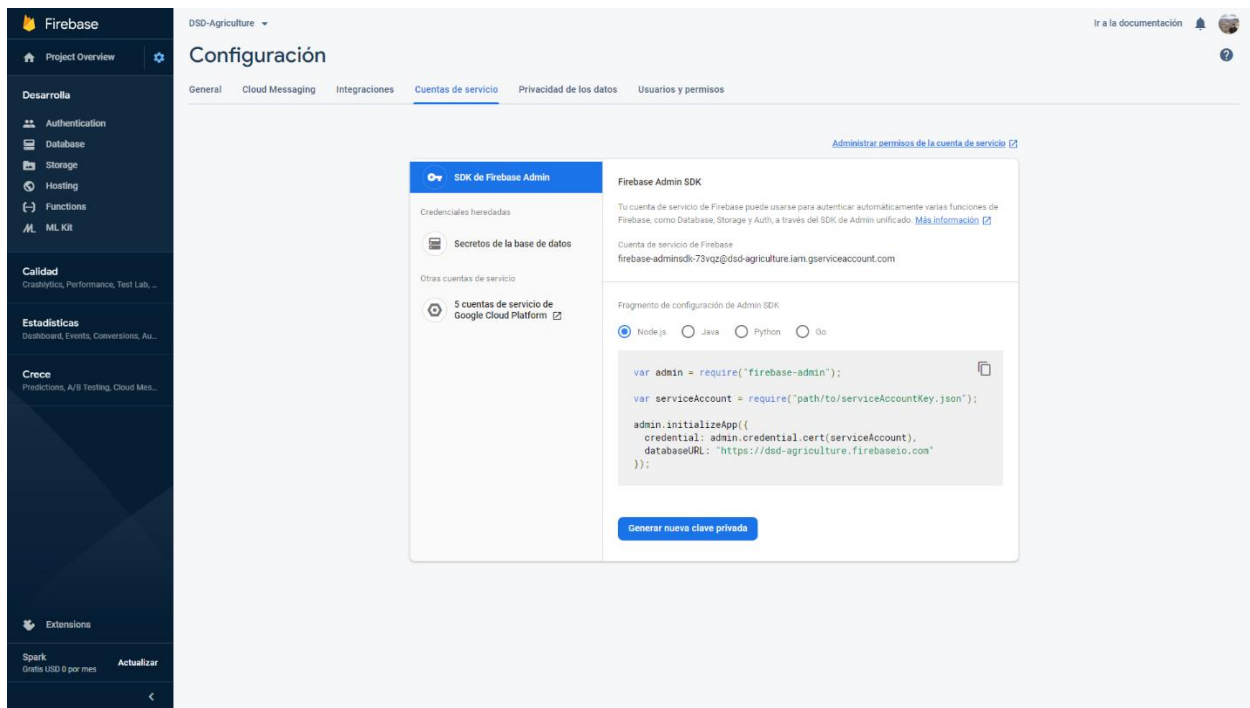




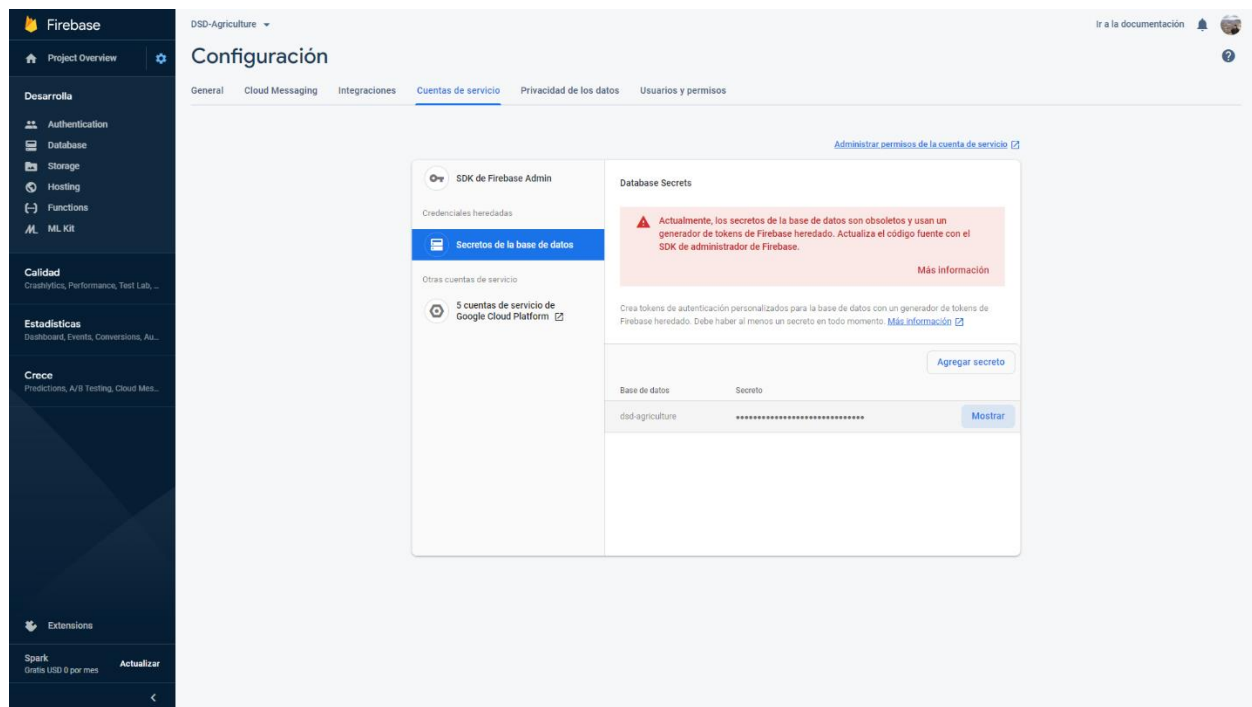
Para poder avanzar en el desarrollo del aplicativo, se debe ir a la “Configuración del proyecto” tras seleccionar la tuerca junto a la “Descripción del Proyecto” en el menú lateral.



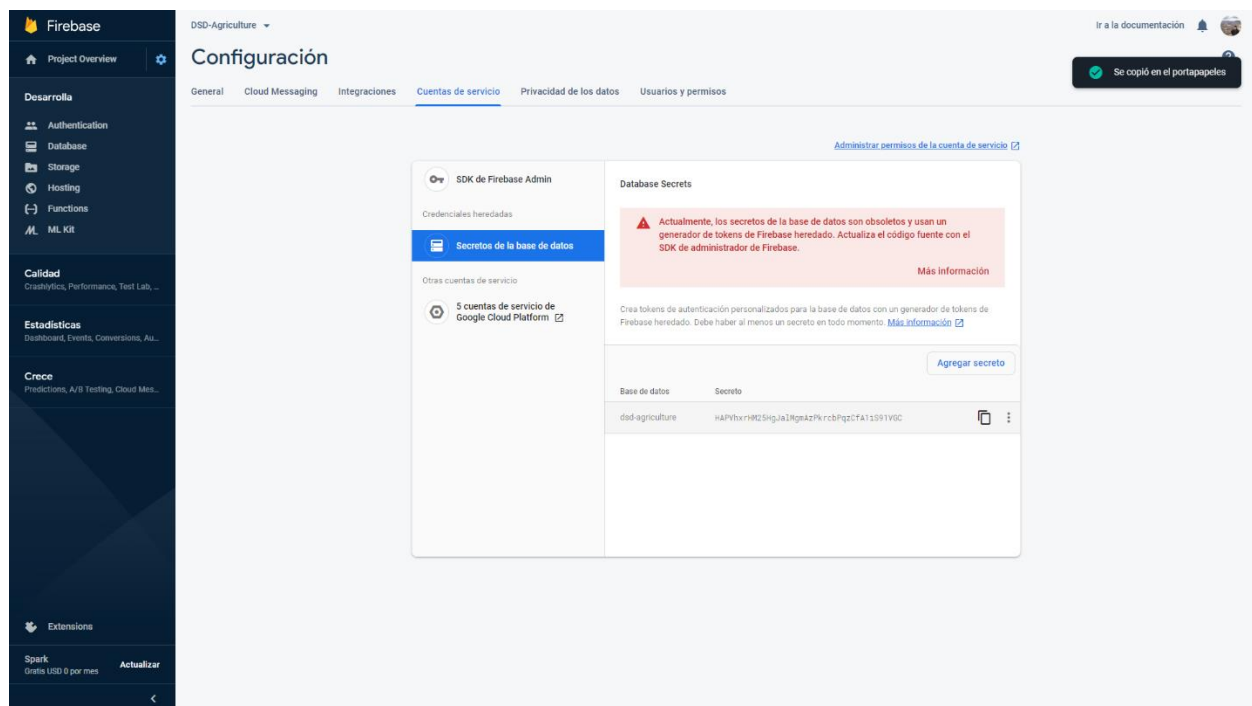
En la nueva ventana, ir a la opción “Cuentas de servicio”.



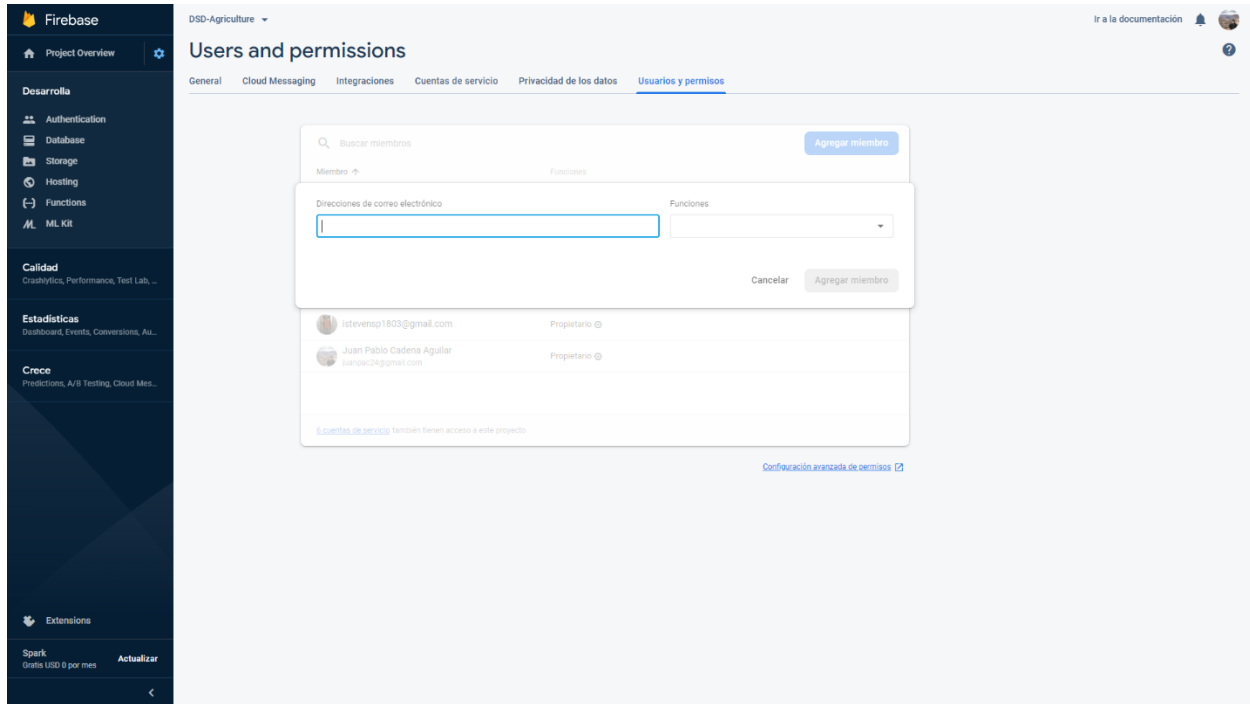
Dirigirse al apartado “Secretos de la base de datos” en el nuevo cuadro central y seleccionar el botón “Mostrar”.



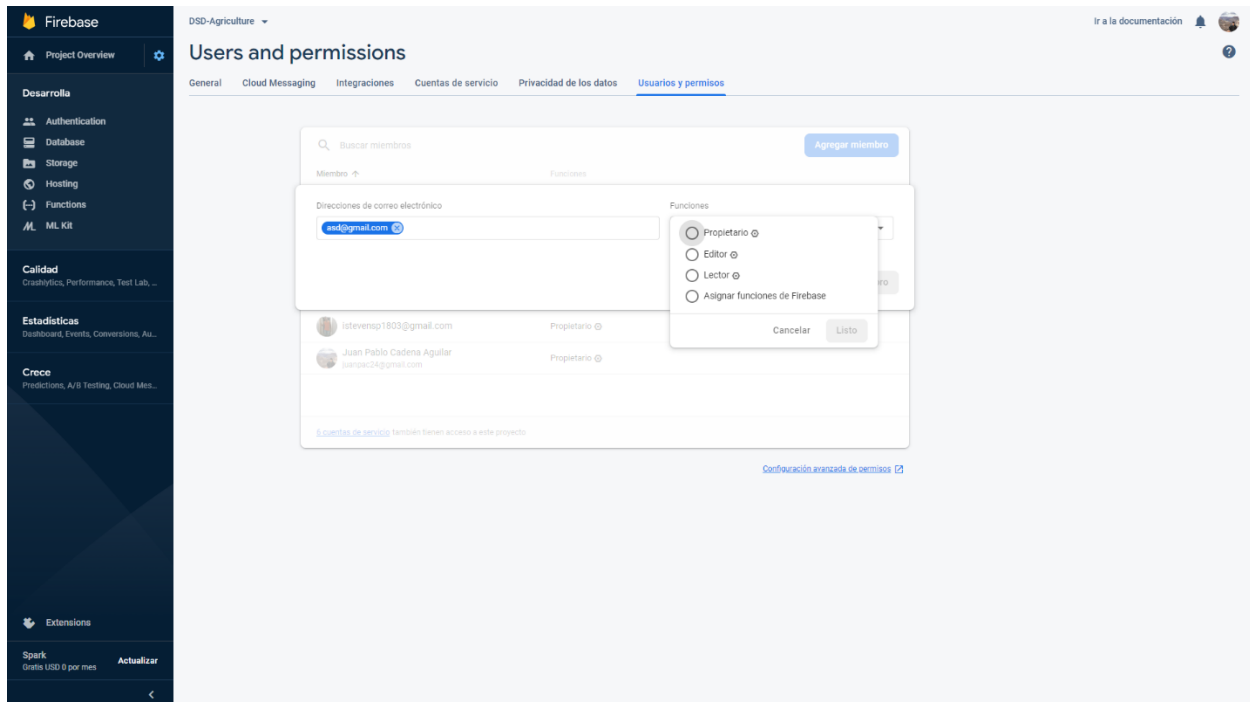
Copiar la ficha “Secreto”, la cual será la autenticación desde el NodeMCU hacia Firebase.



Finalmente seleccionar el submenú “Usuarios y permisos” para poder agregar al compañero de proyecto utilizando el botón “Agregar miembro” en el nuevo cuadro central.



Aquí ingresará el correo y podrá elegir los permisos a dicho usuario.

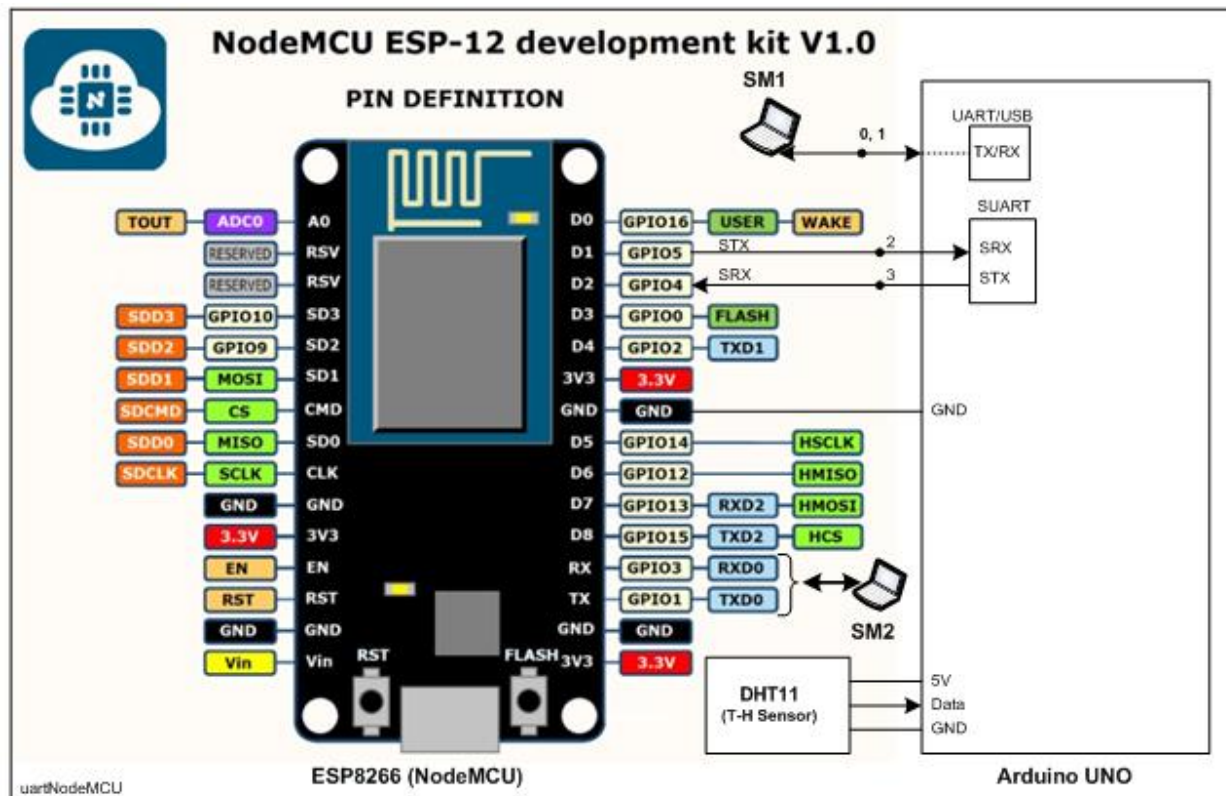


### Prueba con los dispositivos en tiempo real con tiempos de envío de 5 segundos.

Se utilizó VNC Connect, aplicativo multiplataforma con licencia general pública (GPL). Se ejecutó VNC Server en una Raspberry Pi 4 B de 4GB de RAM corriendo sobre un sistema de licencias de software gratuita y de código abierto: Raspbian Buster 4.19 donde se conectó el Arduino Uno vía USB 3.0 y mediante el terminal gráfico serial de código abierto CuteCom se obtuvieron las lecturas de las 5 variables mediante 4 sensores.

La placa NodeMCU que cuenta con un procesador ESP8266 fue conectada vía USB 3.0 a una computadora portátil corriendo con Windows 10-64 bits y las lecturas del Serial fueron apreciadas mediante el monitor del IDE de Arduino y mediante VNC Viewer, se observó de manera remota los datos del ESP8266.

A su vez, dichos dispositivos fueron conectados entre sí como se aprecia en la siguiente gráfica:



*Nota: Imagen referencial en la sección del Arduino UNO, pues se conectaron 3 sensores más.*

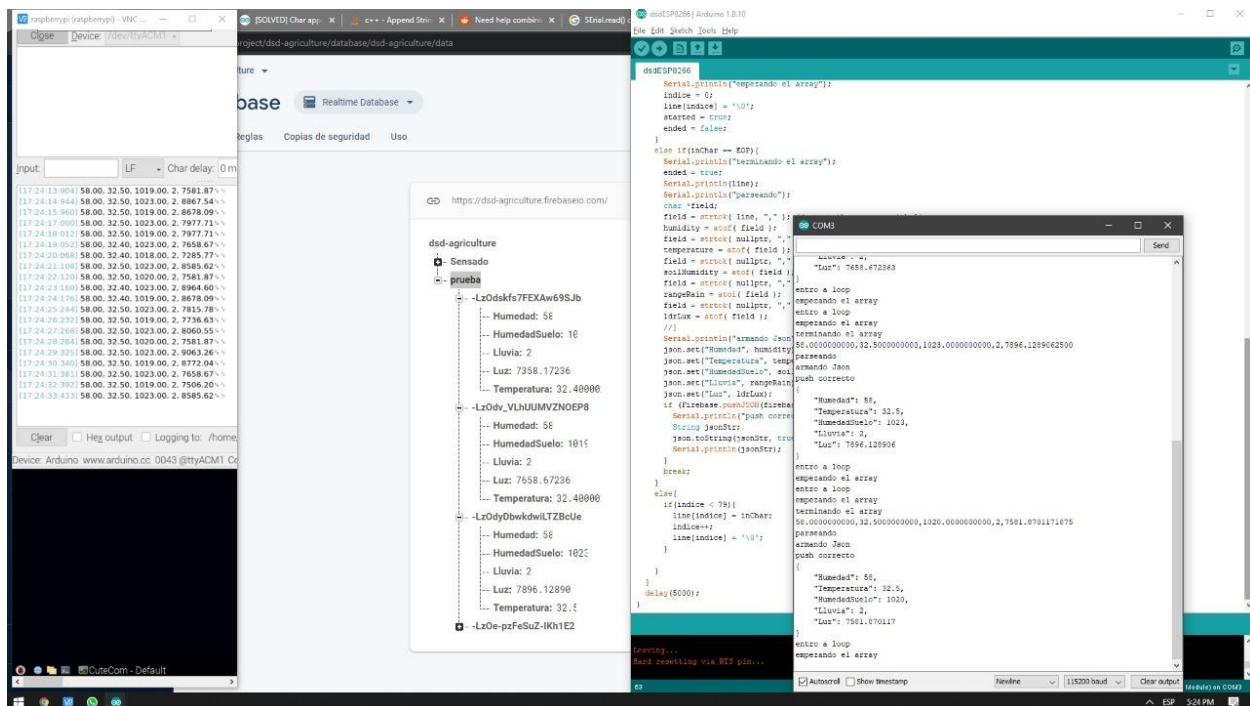


Ilustración 2: Datos enviados en tiempo real.

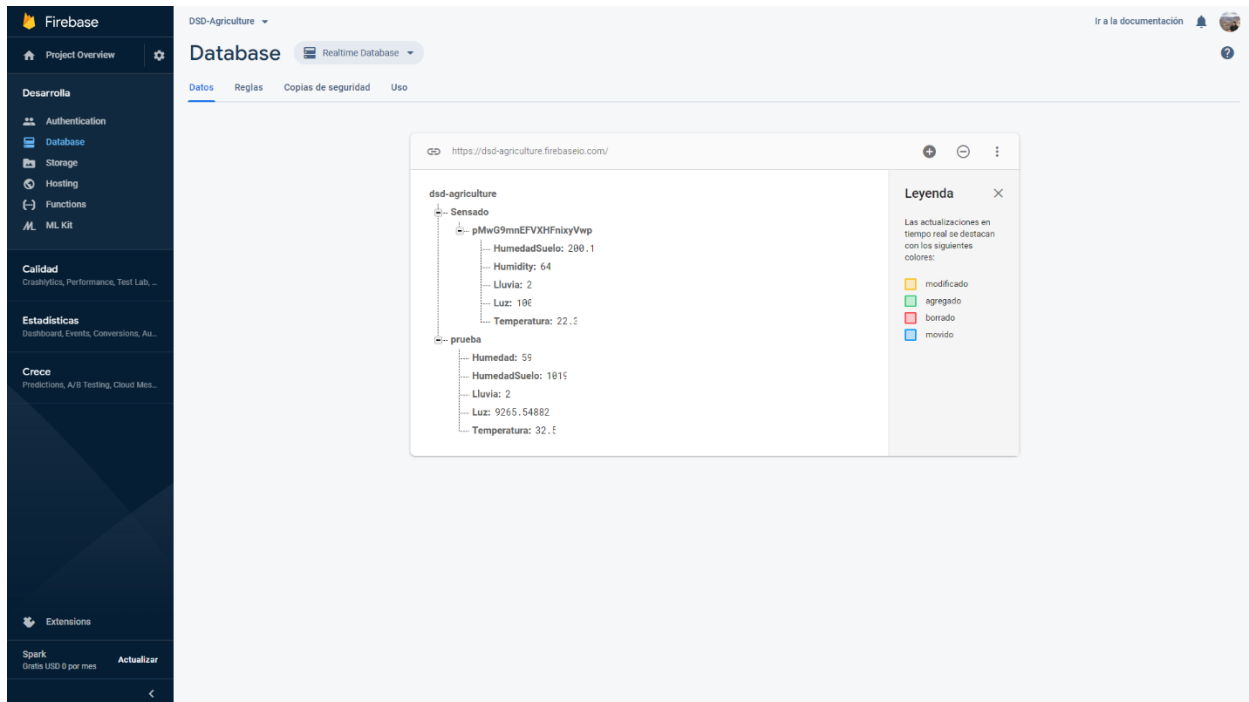


Ilustración 1: Base de datos.

## Código cargado en microcontrolador ATmega328P del Arduino Uno:

```
//Inclusion de librerias y definicion de constantes
#include "DHT.h"
#include <SoftwareSerial.h>
#define DHTPIN 7
#define DHTTYPE DHT11
#define LDR_PIN A2
#define MAX_ADC_READING 1023
#define ADC_REF_VOLTAGE 5.0
#define REF_RESISTANCE 5000
#define LUX_CALC_SCALAR 12518931
#define LUX_CALC_EXPONENT -1.405

//variables globales y uso de librerias
const int sensorMin = 0;
const int sensorMax = 1024;
int fc37 = A0; //rain sensor analog pin
int fc28 = A3; //soil humidity digital pin
DHT dht(DHTPIN, DHTTYPE);
SoftwareSerial SUART(2, 3); //SRX=Dpin2, STX=Dpin3

//configuracion de librerias a tasas de transmision y modos de pines
void setup() {
  Serial.begin(115200);
  SUART.begin(115200);
  dht.begin();
  pinMode(fc28, INPUT);
}

//ciclo infinito
void loop() {
  //variables y sus lecturas respectivas
  float resistorVoltage, ldrVoltage, ldrResistance, ldrLux;
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  float soilHumidity = analogRead(fc28);
  float rainVoltage = analogRead(fc37);
  int ldrRawData = analogRead(LDR_PIN);
  int rangeRain = map(rainVoltage, sensorMin, sensorMax, 0, 3);

  // Conversion ADC con voltaje referencial
  resistorVoltage = (float)ldrRawData / MAX_ADC_READING * ADC_REF_VOLTAGE;

  // voltaje del ldr basado en resta de voltajes voltage across the LDR is the 5V supply minus the
  5k resistor voltage
  ldrVoltage = ADC_REF_VOLTAGE - resistorVoltage;
```

```

// resistencia que el LDR tendria para dicho voltaje
ldrResistance = ldrVoltage/resistorVoltage * REF_RESISTANCE;

// calculo de la luminancia
ldrLux = LUX_CALC_SCALAR * pow(ldrResistance, LUX_CALC_EXPONENT);

//imprimiendo por el puerto serial
Serial.print(humidity);
Serial.print(", ");
Serial.print(temperature);
Serial.print(", ");
Serial.print(soilHumidity);
Serial.print(", ");
Serial.print(rangeRain);
Serial.print(", ");
Serial.print(ldrLux);
Serial.println();

//enviando por SUART al NodeMCU
SUART.print('<');
SUART.print(humidity, DEC);
SUART.print(',');
SUART.print(temperature, DEC);
SUART.print(',');
SUART.print(soilHumidity, DEC);
SUART.print(',');
SUART.print(rangeRain, DEC);
SUART.print(',');
SUART.print(ldrLux, DEC);
SUART.print('>');
SUART.println();

delay(27000);
}

```

Done compiling.

Sketch uses 8016 bytes (24%) of program storage space. Maximum is 32256 bytes.  
Global variables use 338 bytes (16%) of dynamic memory, leaving 1710 bytes for local variables. Maximum is 2048 bytes.

## Código cargado en microcontrolador ATmega328P del Arduino Uno:

```
//Inclusion de librerias y definicion de constantes
#include <ArduinoJson.h>
#include "FirebaseESP8266.h"
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#define SOP '<'
#define EOP '>'
#define FIREBASE_HOST "dsd-agriculture.firebaseio.com" //sitio para alojar la base de
datos en tiempo real
#define FIREBASE_AUTH "HAPVhxrHM25HgJalMgmAzPkrCbPqzCfA1iS91VGC" //ficha
secreta de autentificacion generada de manera heredada por Firebase
#define WIFI_SSID "Free.Wifi" //secuencia identificadora de la red inalambrica
#define WIFI_PASSWORD "cafecito24" //clave de dicha red

//Uso de librerias y variables globales
SoftwareSerial SUART(4, 5); //SRX=Dpin-D2; STX-Dpin-D1
FirebaseData firebaseData;
FirebaseJson json;
bool started = false;
bool ended = false;
char line[80];
byte indice;
const String path = "prueba"; //referencia a la llave de Firebase
int rangeRain=0;
double humidity=0.0, temperature=0.0, soilHumidity=0.0, ldrLux=0.0;

//configuracion de librerias a tasas de transmision altas e inicio de comunicacion
inalambrica con la red y la plataforma de Google
void setup() {
  Serial.begin(115200);
  SUART.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("conectando");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("Conectado a: ");
  Serial.println(WiFi.localIP());
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}

//ciclo infinito
void loop() {
  // Leyendo la informacion por serial entre ambas placas
  while(SUART.available() > 0){
    char inChar = SUART.read();
    if(inChar == SOP){ //validando el inicio del arreglo de la forma: <data>
```



```

    Serial.println("inicio del arreglo");
    indice = 0;
    line[indice] = '\0';
    started = true;
    ended = false;
}
else if(inChar == EOP){ //validando el fin del arreglo
    Serial.println("fin del arreglo");
    ended = true;
    Serial.println(line);
    Serial.println("extrayendo valores a nuevas variables locales");
    char *field;
    field = strtok( line, "," ); // separando el arreglo por comas
    humidity = atof( field ); //transformando lo extraido a tipo flotante
    field = strtok( nullptr, "," ); // redireccionando el resto del arreglo
    temperature = atof( field );
    field = strtok( nullptr, "," );
    soilHumidity = atof( field );
    field = strtok( nullptr, "," );
    rangeRain = atoi( field );
    field = strtok( nullptr, "," );
    ldrLux = atof( field );

    Serial.println("armando estructura Json");
    json.set("Humedad", humidity); //estableciendo las claves y valores de la coleccion
    Json
    json.set("Temperatura", temperature);
    json.set("HumedadSuelo", soilHumidity);
    json.set("Lluvia", rangeRain);
    json.set("Luz", ldrLux);
    if (Firebase.setJSON(firebaseData, path, json)){ //estableciendo la estructura en la
base
        Serial.println("push correcto");
        String jsonStr;
        json.toString(jsonStr, true); //obteniendo el objeto JSON a manera de cadena
        Serial.println(jsonStr);
    }
    break;
}
else{
    if(indice < 79){ //validacion de numero de caracteres en el arreglo del Serial
        line[indice] = inChar;
        indice++;
        line[indice] = '\0';
    }
}
}
}
delay(27000);
}

```

Done compiling.

Executable segment sizes:

IRAM : 454212 - code in flash (default or ICACHE\_FLASH\_ATTR)

IRAM : 28864 / 32768 - code in IRAM (ICACHE\_RAM\_ATTR, ISRs...)

DATA : 1320 ) - initialized variables (global, static) in RAM/HEAP

RODATA : 1616 ) / 81920 - constants (global, static) in RAM/HEAP

BSS : 27296 ) - zeroed variables (global, static) in RAM/HEAP

Sketch uses 486012 bytes (46%) of program storage space. Maximum is 1044464 bytes.

Global variables use 30232 bytes (36%) of dynamic memory, leaving 51688 bytes for local variables. Maximum is 81920 bytes.

## Parte necesaria para la visualización de datos en el servidor:

### 1. Instalación PYTHON

```
sudo apt-get install Python
```

```
pip install uwsgi
```

```
pip install nginxpy
```

En caso de tener problemas con el internet:

1. Verificar que la FPGA se encuentre con la hora y fecha actual.
2. Ejecutar los comandos:

```
sudo apt-get install aptitude
```

```
sudo aptitude update
```

### 2. Instalación NGINX

Antes de empezar con la instalación actualizaremos la lista de los paquetes y los paquetes:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Primero se instalará el paquete nginx:

```
sudo apt-get install nginx
```

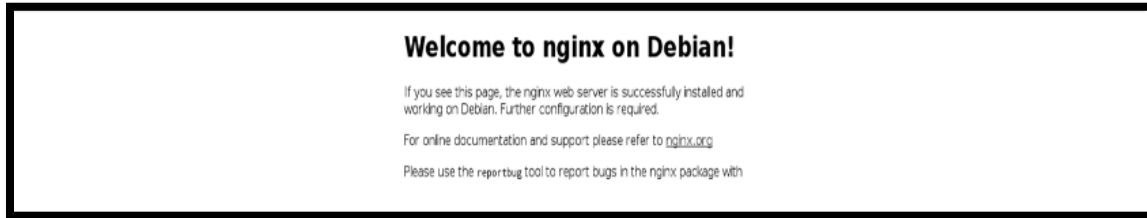
y se iniciará el servidor con:

```
sudo /etc/init.d/nginx start
```

Prueba del servidor web Por defecto, NGINX pone un archivo HTML de prueba en el directorio web. Esta página por defecto se publica cuando navegamos a <http://localhost/> en la FPGA o <http://192.168.0.107> (dirección IP configurada en nuestra FPGA).

Si no supiéramos esa IP para averiguar dicha IP escribiríamos en la línea de comandos:

```
hostname -I
```



### 3. Instalación PHP en NGINX

Instalado Nginx, ahora continuamos con la instalación de PHP:

```
sudo apt-get install php-fpm php-mysql
```

1. Usaré el editor nano para editarlo.

```
sudo nano /etc/nginx/sites-enabled/default
```

2. Buscar la siguiente línea:

```
index index.html index.htm;
```

3. Añadimos index.php después de index , quedando de la siguiente manera:

```
index index.php index.html index.htm;
```

4. Avanzar hacia abajo del documento hasta encontrar una sección con el siguiente contenido:

```
# pass the PHP scripts to FastCGI server
```

```
#location ~ \.php$ {
```

```
location ~ \.php$ {
```

5. Editar el archivo borrando el carácter # en las siguientes líneas

```
location ~ \.php$ {
```

```
    include snippets/fastcgi-php.conf;
```

```
    fastcgi_pass unix:/var/run/php7.3-fpm.sock;
```

```
}
```

Quedando de la siguiente manera:

```
location ~ \.php$ {
    include snippets/fastcgi-php.conf;
#
#       # With php-fpm (or other unix sockets):
fastcgi_pass unix:/run/php/php7.3-fpm.sock;
#       # With php-cgi (or other tcp sockets):
fastcgi_pass 127.0.0.1:9000;
}
```

6. Guardamos los cambios realizados y salimos del editor.

7. Ejecutamos la línea de comandos:

```
sudo /etc/init.d/nginx reload
```

### 3.1 Prueba del PHP

Para comprobar que todo funciona correctamente, creamos un archivo de prueba llamado **prueba.php**:

```
cd /usr/share/nginx/www sudo nano prueba.php
```

En el que incluimos el siguiente código:

```
<?php
```

```
echo phpinfo();
```

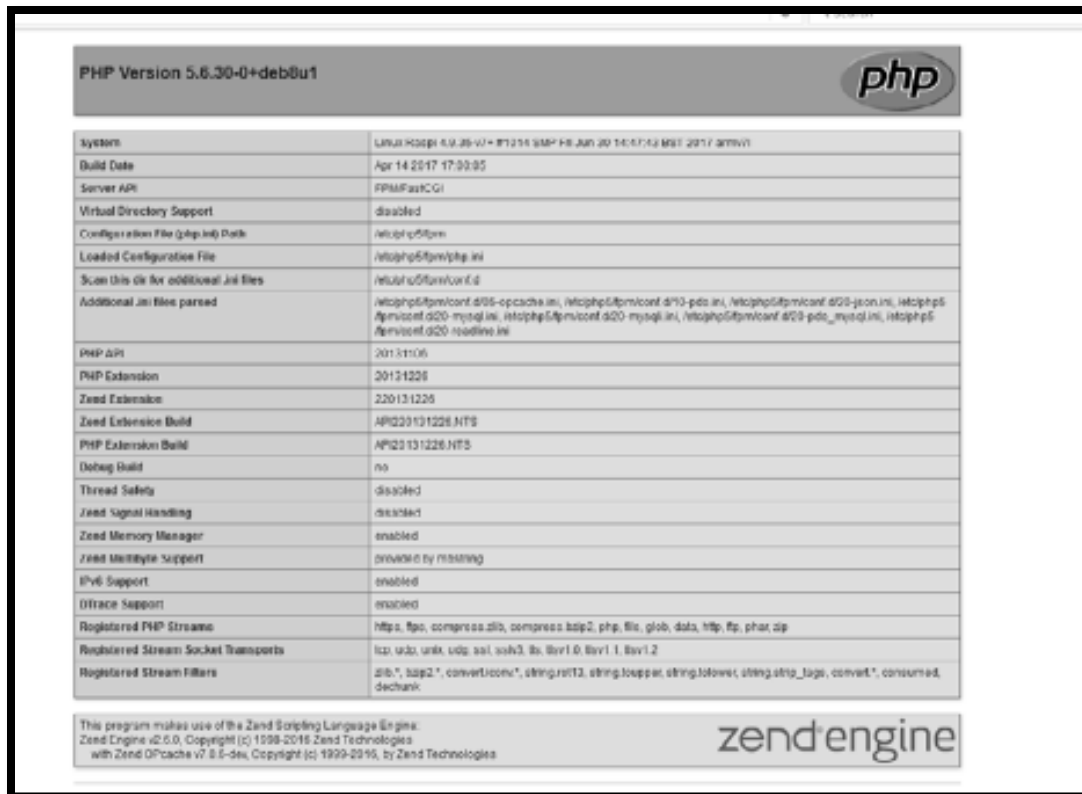
```
?>
```

Se guarda (CTRL + O, Intro, CTRL + X) y se reinicia nginx y PHP:

```
sudo service nginx restart
```

```
sudo service php7.3-fpm restart
```

Abrimos en el navegador del PC, la IP local de la FPGA, seguida del nombre del fichero PHP que hemos creado antes <http://localhost/prueba.php>, nos aparecerá:



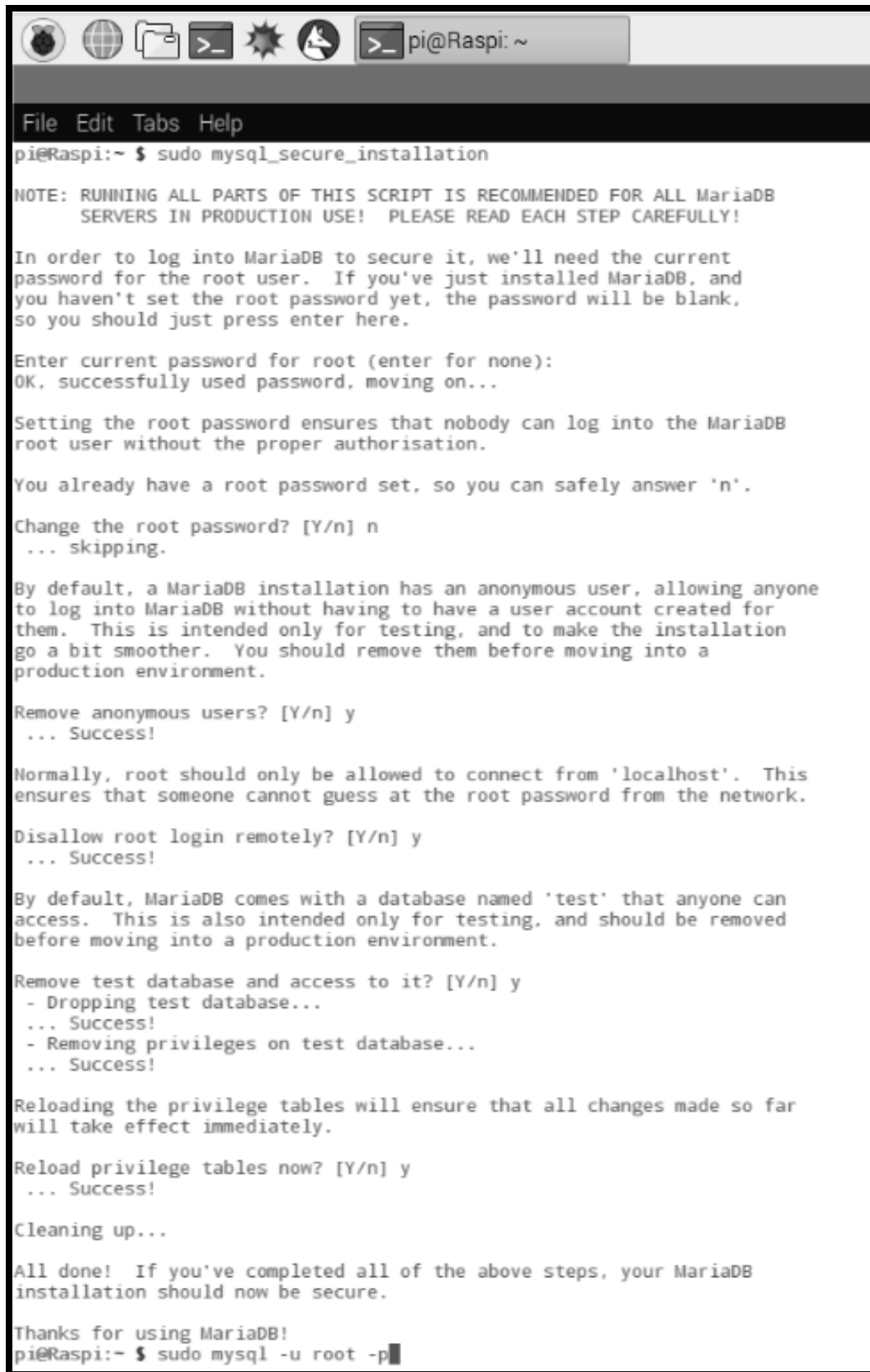
## 4. Instalación de MariaDB

Con las últimas versiones de Raspbian, MariaDB está presente en los repositorios oficiales y para instalarlo tan solo ejecutamos el siguiente comando:

```
sudo apt-get install mariadb-server
```

Para ejecutar este script pondremos en la línea de comandos:

```
sudo mysql_secure_installation
```



```
pi@Raspi:~ $ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

You already have a root password set, so you can safely answer 'n'.

Change the root password? [Y/n] n
... skipping.

By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
pi@Raspi:~ $ sudo mysql -u root -p
```

## 5. Configuración de NGINX y uWSGI para scripts CGI

Añadir una ubicación en el archivo de configuración `/etc/nginx/sites-available/default`. Esto debe colocarse **dentro del bloque "servidor" de la configuración**, después de una de las secciones existentes de "location".

```
server {  
location ~ \.py$ {  
    include uwsgi_params;  
    uwsgi_modifier1 9;  
    uwsgi_pass 127.0.0.1:9000;  
}  
}
```

```
server {  
    location ~ \.py$ {  
        include uwsgi_params;  
        uwsgi_modifier1 9;  
        uwsgi_pass 127.0.0.1:9000;  
    }  
}
```

Guardar los cambios realizados (CTRL + O seguido Intro) y salir del editor nano (CTRL + X), volviendo a cargar el archivo de configuración en la línea de comandos:

```
sudo /etc/init.d/nginx reload
```

### 2. Instalar uWSGI con el plugin CGI

Iremos a la carpeta Desktop:

```
cd /home/pi/Desktop
```

Ejecutamos los siguientes comandos:

```
sudo tar zxvf uwsgi_latest_from_installer.tar.gz
```

```
cd uwsgi-2.0.18
```

```
python uwsgiconfig.py --plugin plugins/cgi
```



```
sudo make PROFILE=cgi
```

```

[thread 3][arm-linux-gnueabihf-gcc -pthread] plugins/transformation_chunked/chunked.o
[thread 1][arm-linux-gnueabihf-gcc -pthread] plugins/transformation_offload/offload.o
[thread 2][arm-linux-gnueabihf-gcc -pthread] plugins/router_memcached/router_memcached.o
[thread 0][arm-linux-gnueabihf-gcc -pthread] plugins/router_redis/router_redis.o
[thread 3][arm-linux-gnueabihf-gcc -pthread] plugins/router_hash/router_hash.o
[thread 1][arm-linux-gnueabihf-gcc -pthread] plugins/router_expires/expires.o
[thread 2][arm-linux-gnueabihf-gcc -pthread] plugins/router_metrics/plugin.o
[thread 0][arm-linux-gnueabihf-gcc -pthread] plugins/transformation_template/tt.o
[thread 3][arm-linux-gnueabihf-gcc -pthread] plugins/stats_pusher_socket/plugin.o
** object linking **
arm-linux-gnueabihf-gcc -pthread -o uwsgi core/utils.o core/protocol.o core/socket.o core/logging.o core/master.o core/emperor.o core/notify.o core/mule.o core/subscription.o core
re/stats.o core/sendfile.o core/async.o core/master_checks.o core/fifo.o core/offload.o core/io.o core/static.o core/websockets.o core/spooler.o core/smp.o core/exceptions.o core/config.o core/setu
r_utils.o core/clock.o core/ini.o core/buffer.o core/reader.o core/writer.o core/alarm.o core/cron.o core/hooks.o core/plugins.o core/lock.o core/cache.o core/daemons.o core/errors.o core/hash.o co
re/master_events.o core/chunked.o core/queue.o core/event.o core/signal.o core/strings.o core/progress.o core/timebmb.o core/ini.o core/fsmon.o core/mount.o core/metrics.o core/builder.o co
re/loop.o core/rpc.o core/getway.o core/loop.o core/cookie.o core/querystring.o core/rb_timers.o core/transformations.o core/uwsgi.o proto/base.o proto/uwsgi.o proto/http.o proto/fastcgi.o p
roto/scgi.o proto/pwsgi.o lib/linux.ns.o core/zlib.o core/yaml.o core/core.o core/legion.o core/xmllconf.o core/dot.h.o core/config.py.o plugins/cgi/cgi_plugin.o plugins/ping/ping_plugin.o plugins/cac
he/cache.o plugins/nagios/nagios.o plugins/rdrtool.o plugins/carbon/carbon.o plugins/rpc/rpc_plugin.o plugins/corerouter/cr_common.o plugins/corerouter/cr_map.o plugins/corerouter/corerouter.o
plugins/fastrouter/fastrouter.o plugins/http/http.o plugins/http/keepalive.o plugins/http/https.o plugins/http/spdy3.o plugins/signal/signal_plugin.o plugins/syslog/syslog_plugin.o plugins/syso
log/syslog_plugin.o plugins/legosocket/legosocket_plugin.o plugins/router_uwsgi/router_uwsgi.o plugins/router_redirect/router_redirect.o plugins/router_basicauth/router_basicauth.o plugins/zergpool/z
ergpool.o plugins/redislog/redislog_plugin.o plugins/mongodblog/mongodblog_plugin.o plugins/router_rewrite/router_rewrite.o plugins/router_http/router_http.o plugins/logfile/logfile.o plugins/router_c
ache/router_cache.o plugins/rawrouter/rawrouter.o plugins/router_static/router_static.o plugins/sslrouter/sslrouter.o plugins/spooler/spooler_plugin.o plugins/cheaper_business/cheaper_business.o p
lugins/symcall/symcall_plugin.o plugins/transformation_tofile/tofile.o plugins/transformation_gzip/gzip.o plugins/transformation_chunked/chunked.o plugins/transformation_offload/offload.o plugins/rout
er_memcached/router_memcached.o plugins/router_redis/router_redis.o plugins/router_hash/router_hash.o plugins/router_expires/expires.o plugins/router_metrics/plugin.o plugins/transformation_template
/tt.o plugins/stats_pusher_socket/plugin.o -lpthread -lm -rdynamic -ldl -lz -luuid -lssl -lcrypto -lexpat -lcrypt
##### uwsgi configuration #####
pcrc = False
kernel = Linux
malloc = libc
execinfo = False
ifaddr = True
ssl = True
zlib = True
locking = pthread_mutex
plugin_dir =
timer = timerfd
yaml = embedded
json = False
http_notify =
routing = False
debug = False
ucontext = False
capabilities = False
ini = expat
event = epoll

##### end of uwsgi configuration #####
total build time: 59 seconds
** uwsgi is ready, launch it with ./uwsgi ***
pi@raspberrypi:~/Desktop/uwsgi-2.0.18 $

```

3. Crear el archivo “**uwsgi.ini**” de configuración en el escritorio, archivo que es necesario para decirle a uWSGI como manejar las solicitudes.

```
[uwsgi]
plugins = cgi
socket = 127.0.0.1:9000
module = pyindex
cgi = /usr/share/nginx/www
cgi-allowed-ext = .py
cgi-helper = .py=python
```

```
[uwsgi]

plugins = cgi

socket = 127.0.0.1:9000

module = pyindex

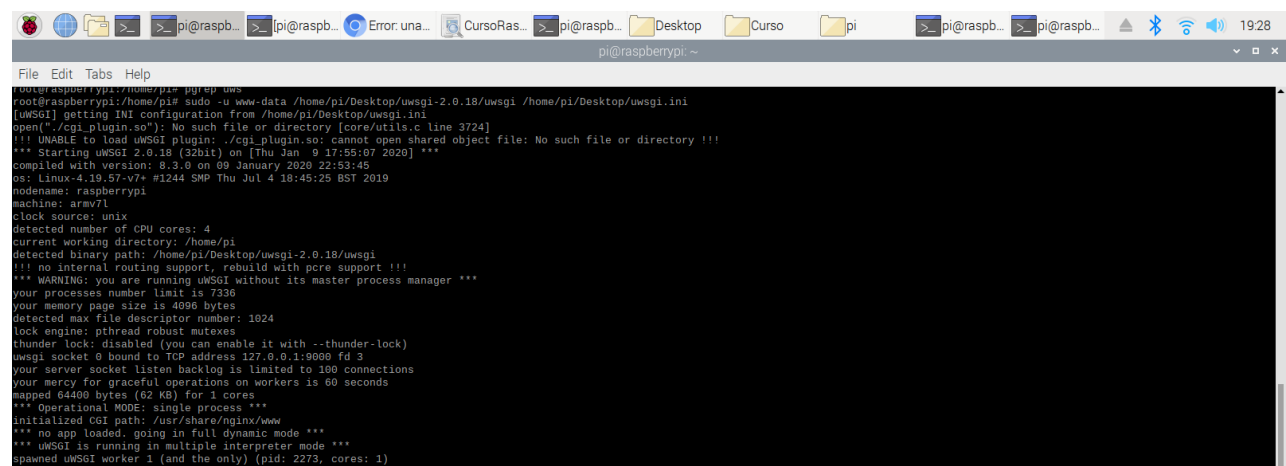
cgi = /usr/share/nginx/www

cgi-allowed-ext = .py

cgi-helper = .py=python
```

Iniciar el servicio uWSGI

```
sudo -u www-data /home/pi/Desktop/uwsgi-2.0.18/uwsgi /home/pi/Desktop/uwsgi.ini
```



```
File Edit Tabs Help
root@raspberrypi:~/Desktop# sudo -u www-data /home/pi/Desktop/uwsgi-2.0.18/uwsgi /home/pi/Desktop/uwsgi.ini
[uWSGI] getting INI configuration from /home/pi/Desktop/uwsgi.ini
open("./cgi_plugin.so"): No such file or directory [core/utls.c line 3724]
!!! UNABLE to load uWSGI plugin: ./cgi_plugin.so: cannot open shared object file: No such file or directory !!!
*** Starting uWSGI 2.0.18 (32bit) on [Thu Jan  9 17:55:07 2020] ***
compiled with version: 8.3.0 on 09 January 2020 22:53:45
os: Linux-4.19.57-v7+ #1244 SMP Thu Jul 4 18:45:25 BST 2019
nodename: raspberrypi
machine: armv7l
clock source: unix
detected number of CPU cores: 4
current working directory: /home/pi
detected binary path: /home/pi/Desktop/uwsgi-2.0.18/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 7336
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uwsgi socket 0 bound to TCP address 127.0.0.1:9000 fd 3
your server socket listen backlog is limited to 100 connections
your mercy for graceful operations on workers is 60 seconds
mapped 61400 bytes (62 KB) for 1 cores
*** Operational MODE: single process ***
initialized CGI path: /usr/share/nginx/www
*** no app loaded, going in full dynamic mode ***
*** uWSGI is running in multiple interpreter mode ***
spawned uWSGI worker 1 (and the only) (pid: 2273, cores: 1)
```

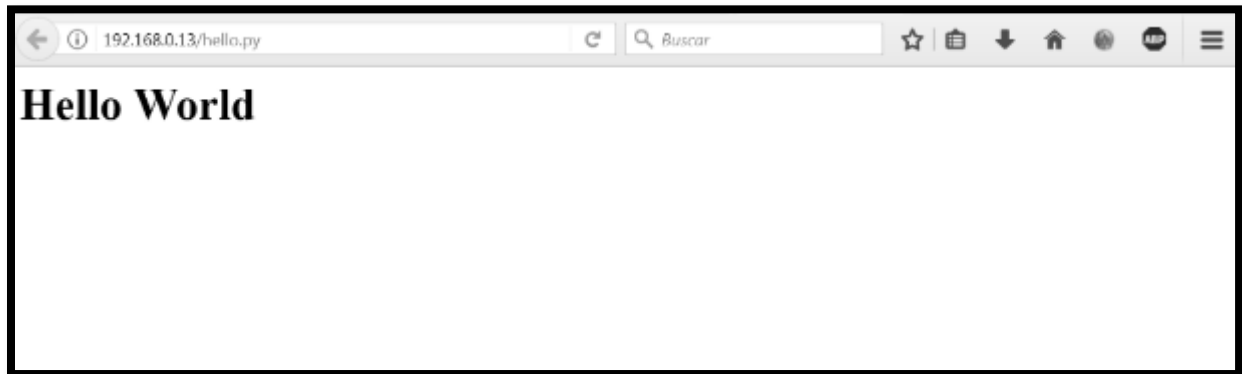
5. El último paso es configurar un script de prueba para comprobar que está funcionando correctamente. Se almacena el archivo en la ubicación **/usr/share/nginx/www/hello.py**

```
#!/usr/bin/env python
```

```
print "Content-type: text/html\n\n"
```

```
print "<h1>Hello World</h1>"
```

Ahora en un navegador ponemos **http://direcciónip/hello.py** y si todo está correcto, tendremos que ver el mensaje **“Hello World”**



## Red neural usada en el CP1 de la FPGA:

The screenshot shows the Eclipse IDE with the Nios II - NIOS1/Neuronal.c project. The main editor displays the following C code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*===== niNN vector de entrada =====*/
float niNN[]={42, 27.1, 119.76, 0, 434}; // 'Humedad Aire', 'Temperatura', 'Humedad Suelo', 'Lluvia',
/*===== funcion mapminmax =====*/
float *mapminmax(float input[])
{
    int i;
    float ymax=1,ymin=-1;
    float xmin[]={42,27.1,434,119.76,0};
    float xmax[]={69,32.9,1001,9581.99,2};
    static float output[5];
    for(i=0;i<5;i++)
    {
        output[i]=((ymax-ymin)/(xmax[i]-xmin[i]))*(input[i]-xmin[i])+ymin;
    }
    return output;
}

/*===== producto punto =====*/
float DotProduct(float input1[], float input2[])
{
    int i=0;
    float acumulador=0;
    for (i=0;i<5;i++)
    {
        //-----segun el numero de entradas
        //-----segun el numero de entradas
    }
}
```

The right sidebar shows the Outline view with the following structure:

- stdio.h
- stdlib.h
- math.h
- niNN: float[]
- mapminmax(float[]): float\*
- DotProduct(float[], float[]): float
- tanisig(float): float\*
- layer1(float): float\*
- DotProduct(float[], float[]): float
- mapminreverse(float): float
- layer2(float): float
- main(): int

The bottom console shows the following output:

```
Downloading 04020000 ( 0%)
Downloading 04027848 (97%)
Downloaded 31KB in 0.0s

Verifying 04020000 ( 0%)
Verifying 04027848 (97%)
Verified OK
Starting processor at address 0x04020230
```

The screenshot shows the Eclipse IDE with the Nios II - Nios1/Neuronal.c project. The main editor displays the following C code:

```
/*===== main =====*/
int main()
{
    int n;
    float *GPP1=mapminmax(niNN);
    float input1[6];
    float input2[10];
    for(n=0;n<5;n++)
    {
        input1[n]=GPP1[n];
    }
    float *GPP2=layer1(input1);
    for(n=0;n<10;n++)
    {
        input2[n]=GPP2[n];
    }
    float output=layer2(input2);
    printf("%f\n",output);
    return 0;
}
```

The right sidebar shows the Outline view with the following structure:

- stdio.h
- stdlib.h
- math.h
- niNN: float[]
- mapminmax(float[]): float\*
- DotProduct(float[], float[]): float
- tanisig(float): float\*
- layer1(float): float\*
- DotProduct(float[], float[]): float
- mapminreverse(float): float
- layer2(float): float
- main(): int

The bottom console shows the following output:

```
Downloading 04020000 ( 0%)
Downloading 04027848 (97%)
Downloaded 31KB in 0.0s

Verifying 04020000 ( 0%)
Verifying 04027848 (97%)
Verified OK
Starting processor at address 0x04020230
```

passing argument 1 of 'printf' makes pointer from integer without a cast [-Wint-conversion]

## Montaje de Gráficas en el servidor

Dirigirse a la ruta `/usr/share/nginx/www` y ahí debe pegar todos los archivos que se encuentran en: <https://github.com/istevensp/dsd20192TSSJP.git>

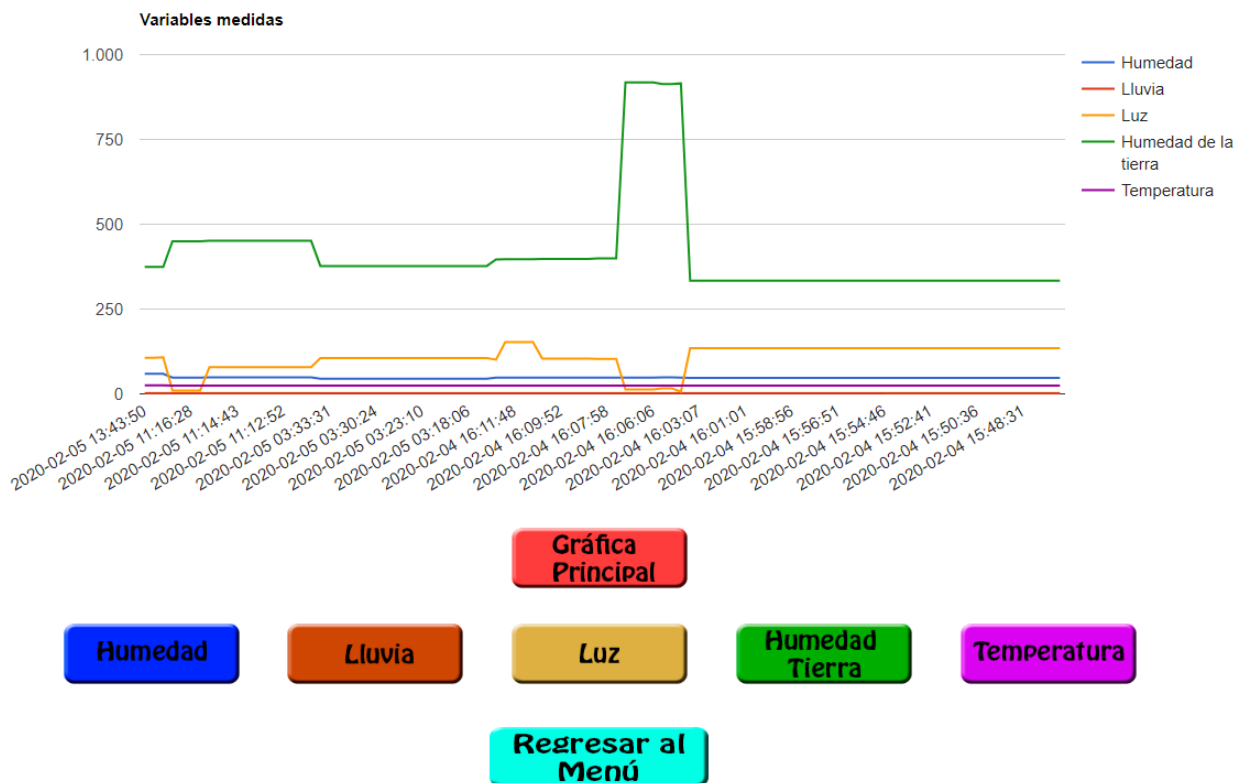
Para clonar dicho repositorio usar el comando:

Git clone <https://github.com/istevensp/dsd20192TSSJP.git>

Luego de eso diríjase a la ruta `http://localhost`.

Finalmente podremos visualizar nuestro servidor:

### Grafica en tiempo real



Ahora para poder recuperar la información desde nuestros sensores, debemos cargar el código en cada dispositivo en el Arduino cargaremos el código del archivo: [dsd.ino](#) que se encuentra en el repositorio indicado anteriormente, además también en el microcontrolador ESP8266 debemos también cargarle el código del archivo [dsdESP8266.ino](#) que también se encuentra en el repositorio.

Para comenzar la recolección de datos desde la base de datos debemos ejecutar el siguiente comando:

**Sudo Python /usr/share/nginx/www/Read.py &**

Luego de esto automáticamente nuestro servidor recibirá la información que se encuentra contenida en FIREBASE y llenará la base de datos de la cual obtendremos los valores para graficar.