



Instituto Tecnológico de San Juan del Río



INGENIERÍA EN SISTEMAS COMPUTACIONALES

**Almacenamiento en paralelo para IoT
Web Api concurrente con dispositivo de
IoT.**

P R E S E N T A N:

CRUZ MENDOZA LIZETH ABIGAIL 17590089

PALOMA VICTORIANO ARAEL 16590505

PÉREZ UGALDE JOSÉ IVÁN 16590110

ING. LEONARDO VALDÉS ARTEAGA
PERIODO
ENERO - JUNIO
2021





Web API Concurrente con Dispositivo IoT

1. Objetivo.

Desarrollar una prueba de concepto de un servicio web que recibe información en paralelo desde diferentes dispositivos IoT.

2. Materiales.

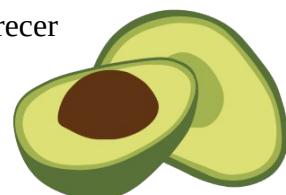
- ❑ Nodemcu
- ❑ Interruptores ON/OFF
- ❑ Resistencias
- ❑ Laptop
- ❑ Cables para protoboard

3. Software.

Para realizar el mini proyecto se utiliza los siguiente:

4. ArangoDB

Es una base de datos multi-modelo combinando los modelos: Clave/Valor, Documentos y Grafos en un solo núcleo desarrollado en C++ y haciendo uso de un sólo lenguaje de consultas, AQL (ArangoDB Query Language). Almacena los datos en documentos tipo JSON, para ofrecer escalabilidad horizontal, propone un esquema CP (teorema de CAP) Master/Master en el que no hay puntos únicos de fallo. Esto significa que, como casi todas las tecnologías NoSQL, ante un problema de red, elige la consistencia de la información frente a la disponibilidad y que su arquitectura garantiza que ante un eventual fallo, no se colapse el sistema completo.



ArangoDB

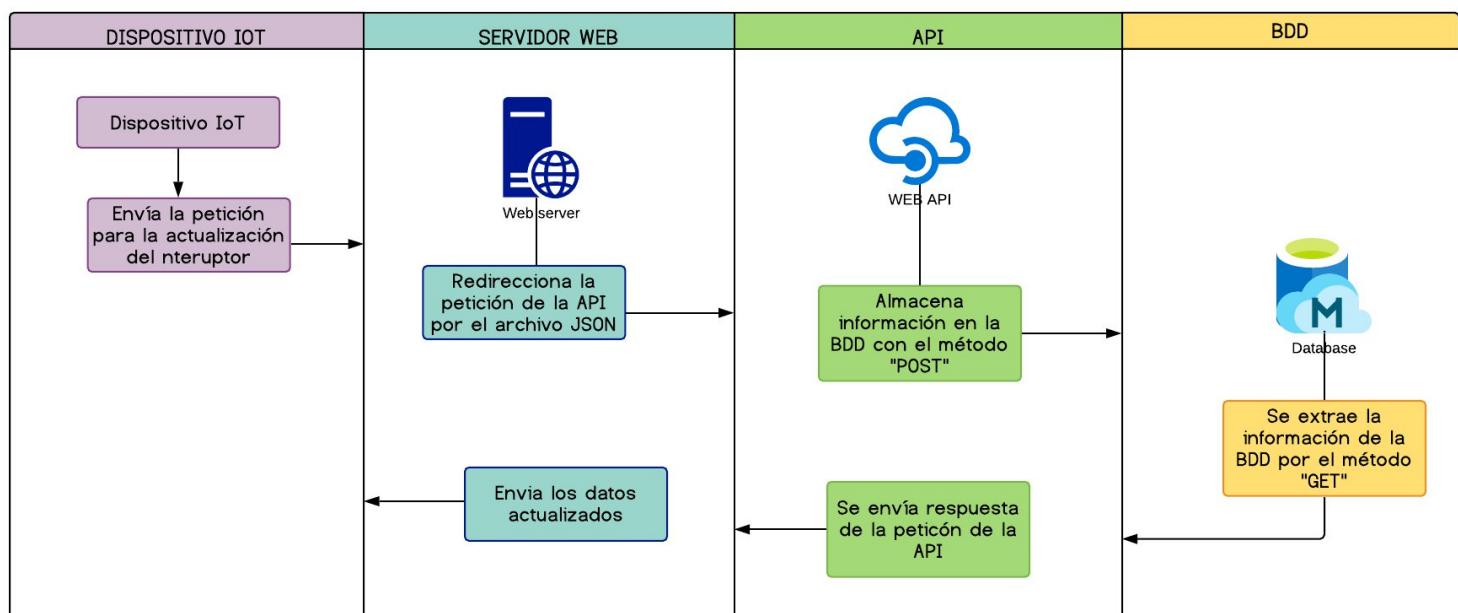




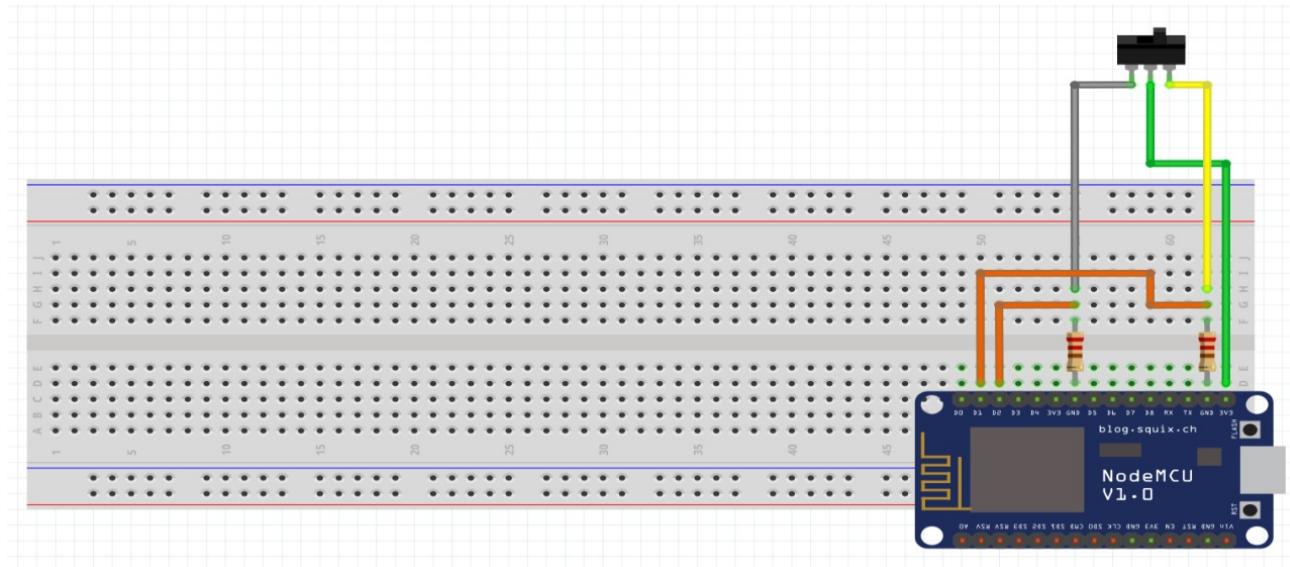
5. Arduino IDE

El software Arduino (IDE) de código abierto facilita la escritura de código y su carga en la placa, NodeMCU. Funciona en cualquier sistema operativo Windows, Mac Os y Linux. El entorno está escrito en Java y basado en Processing y otro software de código abierto. Además que la NodeMCU es una plataforma IoT de código abierto.

6. Diagrama UML



7. Circuito

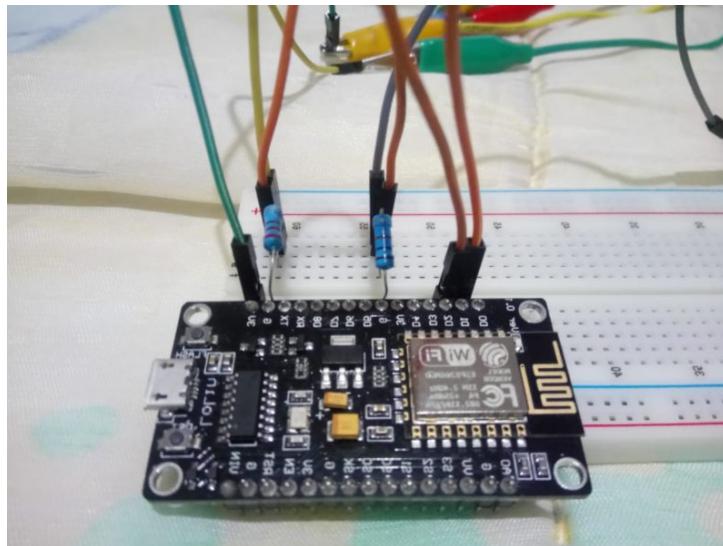


A continuación, se muestra el circuito construido en la protoboard conectado a la Nodemcu para realizar la prueba sobre su ejecución y funcionamiento, para el armado del circuito se utilizó un interruptor de 3 entradas (voltaje, tierra y señal), por la entrada del voltaje se envía el estado del interruptor.





Instituto Tecnológico de San Juan del Río





API

```
1 'use strict';
2 const _ = require('lodash');
3 const joi = require('joi');
4
5 module.exports = {
6   schema: {
7     // Describe the attributes with joi here
8     _key: joi.string(),
9     estado:joi.number()
10 },
11 forClient(obj) {
12   // Implement outgoing transformations here
13   obj = _.omit(obj, ['_id', '_rev', '_oldRev']);
14   return obj;
15 },
16 fromClient(obj) {
17   // Implement incoming transformations here
18   return obj;
19 }
20 };
21 }
```





```
12/3/2021                               datos.js
1 'use strict';
2 const dd = require('dedent');
3 const joi = require('joi');
4 const httpError = require('http-errors');
5 const status = require('statuses');
6 const errors = require('@arangodb').errors;
7 const createRouter = require('@arangodb/foxx/router');
8 const Dato = require('../models/dato');
9
10 const datos = module.context.collection('datos');
11 const keySchema = joi.string().required()
12 .description('The key of the dato');
13
14 const ARANGO_NOT_FOUND = errors.ERROR_ARANGO_DOCUMENT_NOT_FOUND.code;
15 const ARANGO_DUPLICATE = errors.ERROR_ARANGO_UNIQUE_CONSTRAINT_VIOLATED.code;
16 const ARANGO_CONFLICT = errors.ERROR_ARANGO_CONFLICT.code;
17 const HTTP_NOT_FOUND = status('not found');
18 const HTTP_CONFLICT = status('conflict');
19
20 const router = createRouter();
21 module.exports = router;
22
23
24 router.tag('dato');
25
26
27 router.get(function (req, res) {
28   res.send(datos.all());
29 }, 'list')
30 .response([Dato], 'A list of datos.')
31 .summary('List all datos')
32 .description(dd)
33   Retrieves a list of all datos.
34 );
35
36
37 router.post(function (req, res) {
38   const dato = req.body;
39   let meta;
40   try {
41     meta = datos.save(dato);
42   } catch (e) {
43     if (e.isArangoError && e.errorNum === ARANGO_DUPLICATE) {
44       throw httpError(HTTP_CONFLICT, e.message);
45     }
46     throw e;
47   }
48   Object.assign(dato, meta);
49   res.status(201);
50   res.set('location', req.makeAbsolute(
51     req.reverse('detail', {key: dato._key})
52   ));
53   res.send(dato);
54 }, 'create')
55 .body(Dato, 'The dato to create.')
56 .response(201, Dato, 'The created dato.')
57 .error(HTTP_CONFLICT, 'The dato already exists.')
58 .summary('Create a new dato')
59 .description(dd)
60   Creates a new dato from the request body and
```

localhost:4649/?mode=javascript

1/2

Ilustración 1 Creación del post





```
12/3/2021                               datos.js
61   returns the saved document.
62 `);
63
64
65 router.get('/:key', function (req, res) {
66   const key = req.pathParams.key;
67   let dato
68   try {
69     dato = datos.document(key);
70   } catch (e) {
71     if (e.isArangoError && e.errorNum === ARANGO_NOT_FOUND) {
72       throw httpError(HTTP_NOT_FOUND, e.message);
73     }
74     throw e;
75   }
76   res.send(dato);
77 }, 'detail')
78 .pathParam('key', keySchema)
79 .response(Dato, 'The dato.')
80 .summary('Fetch a dato')
81 .description(dd`
82   Retrieves a dato by its key.
83 `);
84
```

Ilustración 2 Creación del get





Arduino

Se agregaron varias librerías para poder conectar la placa mediante la red wifi y para poder conectarse al servidor mediante cliente.

```
12/3/2021           Prueba.ino
1 #include <ArduinoJson.h>
2
3 #include <ESP8266HTTPClient.h>
4 #include <ESP8266WiFi.h>
5
6 #include <WiFiClient.h>
7 //
*****CONEXION*****
8 const char* ssid = "INFINITUM3F9C_2.4";
9 const char* password = "Mariana016";
10 HTTPClient http;
11 //*****VARIABLES DE CONEXION AL
12 String server="http://arango.conoce360.tech/_db/_system/datos/datos";
13
14
15 const int pulsadorPin = 4;
16 const int pulsadorPin2 = 5;
17 const int ledPin = 2;
18 String estado = "";
19 int val=0;
20 int val2=0;
21 void setup() {
22     // Activamos los pines de entrada y salida
23     pinMode(pulsadorPin, INPUT);
24     pinMode(ledPin, OUTPUT);
25     Serial.begin(9600);
26     Serial.print("Conectando con ");
27     Serial.println(ssid);
28     WiFi.begin(ssid, password);
29     while (WiFi.status() != WL_CONNECTED) {
30         delay(500);
31         Serial.print(".");
32     }
33     Serial.println("");
34     Serial.println("WiFi conectado");
35     Serial.println(WiFi.localIP());
36 }
37
38 void loop(){
39     val = digitalRead(pulsadorPin);
40     val2 = digitalRead(pulsadorPin2);
41
42     if(val == HIGH){
43         digitalWrite(ledPin, HIGH);
44         estado = "Derecha";
45         estado2 = 1;
46         Serial.println(estado);
47         envioDatos();
48         delay(2500);
49     }
50     if(val2 == HIGH){
51         digitalWrite(ledPin, HIGH);
52         estado = "Izquierda";
53         estado2 = 2;
54         Serial.println(estado);
55         envioDatos();
56         delay(2500);
57     }
}
localhost:4649/?mode=clike
```

1/2

Ilustración 3 Conexión a la BD





12/3/2021

Prueba.ino

```
58 if(val == LOW && val2 == LOW){  
59     digitalWrite(ledPin, LOW);  
60     estado = "Apagado";  
61     Serial.println(estado);  
62     envioDatos();  
63     delay(2500);  
64 }  
65 }  
66  
67  
68 void envioDatos(){  
69     char json[256];  
70     StaticJsonBuffer<500> jsonBuffer;  
71     JsonObject& root = jsonBuffer.createObject();  
72     root["estado"] = estado;  
73     root.printTo(json, sizeof(json));  
74     http.addHeader("Content-Type", "application/json");  
75     Serial.println("\n");  
76     Serial.println(json);  
77     http.begin(server);  
78     http.POST(json);  
79     http.writeToStream(&Serial);  
80 }  
81
```

9. Pruebas:

- *Pruebas de forma local:*



Instituto Tecnológico de San Juan del Río

Actividades ▾ Brave Web Browser ▾

/dev/ttyUSB0 12 de mar

Autoscroll Mostrar marca temporal Nueva linea 9600 baudio Limpiar salida

```

M0P1 conectado
192.168.1.244
Apagado

{"estado": "Apagado"}
{"estado": "Apagado", "_key": "1117088"}Apagado

{"estado": "Apagado"}
Izquierda

{"estado": "Izquierda"}
{"estado": "Izquierda", "_key": "1117098"}Izquierda

{"estado": "Izquierda"}
Derecha

{"estado": "Derecha"}
{"estado": "Derecha", "_key": "1117108"}Derecha

{"estado": "Derecha"} 
```

Content	_key	+	-
{"_id": "datos_datos/1117088", "_key": "1117088", "_rev": "_cAiH5ei---", ...}	1117088	-	-
{"_id": "datos_datos/1117090", "_key": "1117090", "_rev": "_cAiIBiY---", ...}	1117090	-	-
{"_id": "datos_datos/1117098", "_key": "1117098", "_rev": "_cAiIDp0---", ...}	1117098	-	-
{"_id": "datos_datos/1117100", "_key": "1117100", "_rev": "_cAiILVC---", ...}	1117100	-	-
{"_id": "datos_datos/1117108", "_key": "1117108", "_rev": "_cAiIOC2---", ...}	1117108	-	-
{"_id": "datos_datos/1117110", "_key": "1117110", "_rev": "_cAiIVum---", ...}	1117110	-	-
{"_id": "datos_datos/1117118", "_key": "1117118", "_rev": "_cAiIYMK---", ...}	1117118	-	-





```
_id: datos_datos/1117088
_rev: _cAiH5ei...
_key: 1117088
```

Tree ▾

Select a node...

- object {1} ☰
 - estado : Apagado

```
_id: datos_datos/1117098
_rev: _cAiIDp0...
_key: 1117098
```

Tree ▾

Select a node...

- object {1} ☰
 - estado : Izquierda

