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:

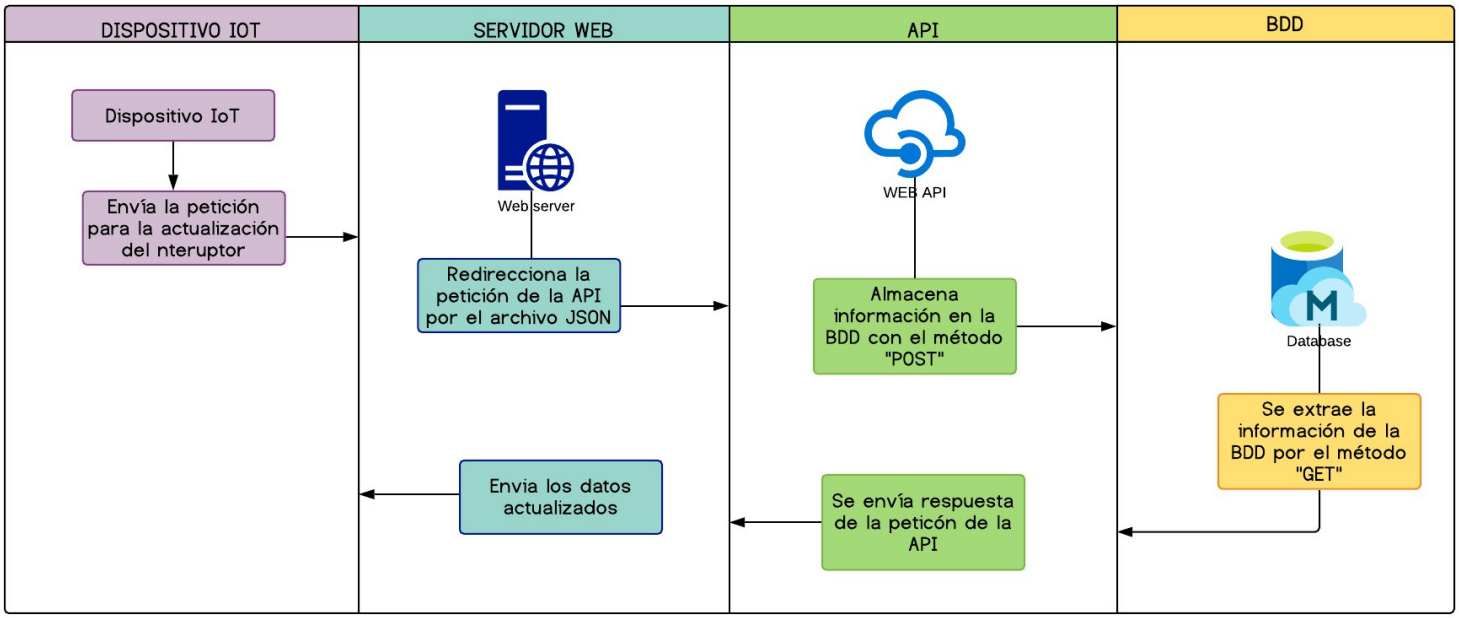
4. ArangoDB

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.

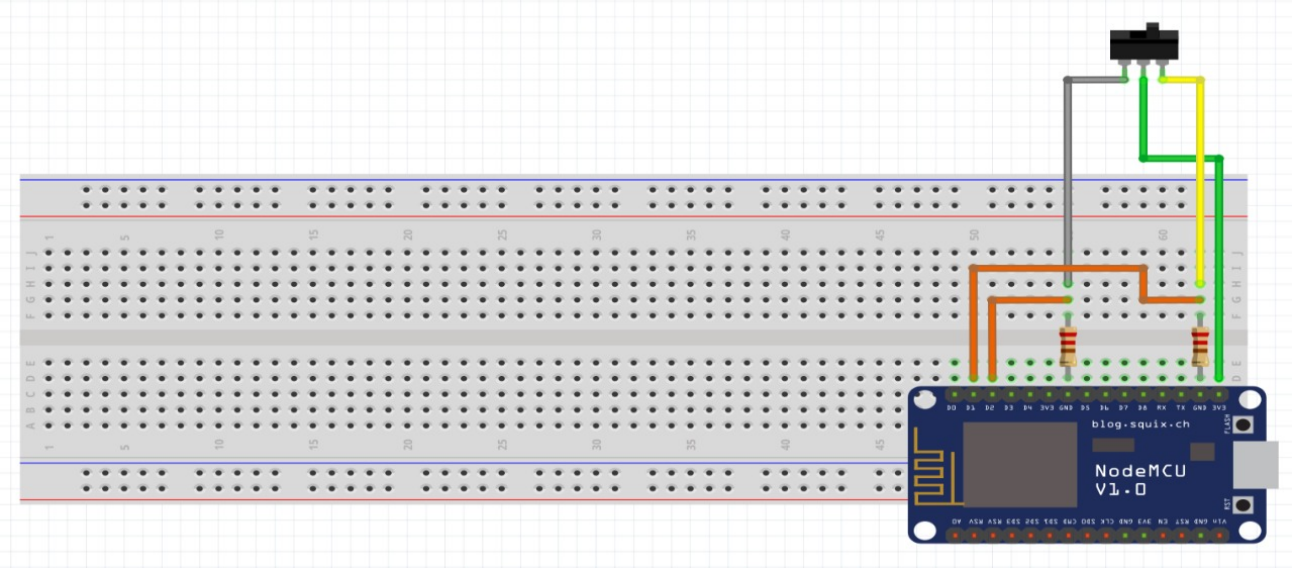
5. Arduino IDE

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.

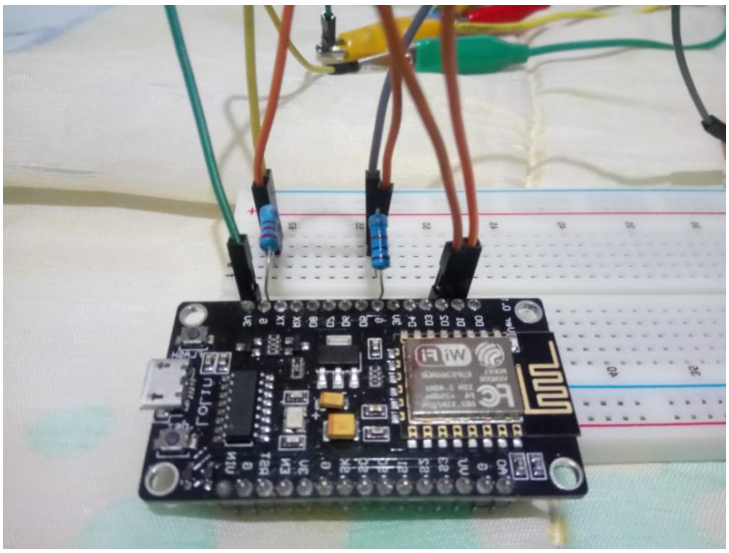
6. Diagrama UML



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.



8. Códigos

API

```
'use strict';
const _ = require('lodash');
const joi = require('joi');

module.exports = {
  schema: {
    // Describe the attributes with joi here
    _key: joi.string(),
    estado:joi.number()
  },
  forClient(obj) {
    // Implement outgoing transformations here
    obj = _.omit(obj, ['_id', '_rev', '_oldRev']);
    return obj;
  },
  fromClient(obj) {
    // Implement incoming transformations here
    return obj;
  }
};

```

```python
from sanic import Sanic
from sanic import response
from sanic.response import json
from pyArango.connection import *
import json
import asyncio
import requests
import aiohttp
app = Sanic("Hola Mundo!")

# URL LOCAL : http://localhost:8529/_db/_system/Estado/estado
# URL SERVIDOR : https://arango.conoce360.tech

# VARIABLES DE CONEXIÓN CON ARANGODB!
con = Connection(arangoURL="https://arango.conoce360.tech",
username="root",password="conoce360")
bd = con["_system"]


# METODO GET
@app.route('/obtenerData', methods=['GET'])
async def estado(request):
    aql = 'For g1 in Estado_Estado return g1'
    queryResult = bd.AQLQuery(aql, rawResults = True, batchSize = 100)
    result = [r1 for r1 in queryResult]
    print("ECHO!")
    return response.json(result)



# METODO POST
@app.route('/Estado', methods=['POST'])
async def echo(request):
    if request.json:
        resp=(request.json)
        resp2 = {"estado":resp}
        aql = "INSERT {@resp2} INTO Estado_Estado"
        queryResult = bd.AQLQuery(aql, bindVars=resp2)
        return json(request.json)
    return json({"RESP":"ERROR"})

if __name__ == '__main__':
    app.run()
```

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

```
#include <ArduinoJson.h>

#include <ESP8266HTTPClient.h>
#include <ESP8266WiFi.h>

#include <WiFiClient.h>
//
**********************************CONEXION*****************************
**
const char* ssid = "INFINITUM3F9C_2.4";
const char* password = "Mariana016";
HTTPClient http;
//*************************************VARIABLES DE CONEXION AL
SERVIDOR*************************************
String server="http://arango.conoce360.tech/_db/_system/datos/datos";


const int pulsadorPin = 4;
const int pulsadorPin2 = 5;
const int ledPin = 2;
String estado = "";
int val=0;
int val2=0;
void setup() {
  // Activamos los pines de entrada y salida
    pinMode(pulsadorPin, INPUT);
    pinMode(ledPin, OUTPUT);
    Serial.begin(9600);
    Serial.print("Conectando con ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println(WiFi.localIP());
}

void loop(){
  val = digitalRead(pulsadorPin);
  val2 = digitalRead(pulsadorPin2);

  if(val == HIGH){
    digitalWrite(ledPin, HIGH);
    estado = "Derecha";
    estado2 = 1;
    Serial.println(estado);
    envioDatos();
    delay(2500);
    }
  if(val2 == HIGH){
    digitalWrite(ledPin, HIGH);
    estado = "Izquierda";
    estado2 = 2;
    Serial.println(estado);
    envioDatos();
    delay(2500);
    }
```

localhost:4649/?mode=clike 1/2

Ilustración 3 Conexión a la BD

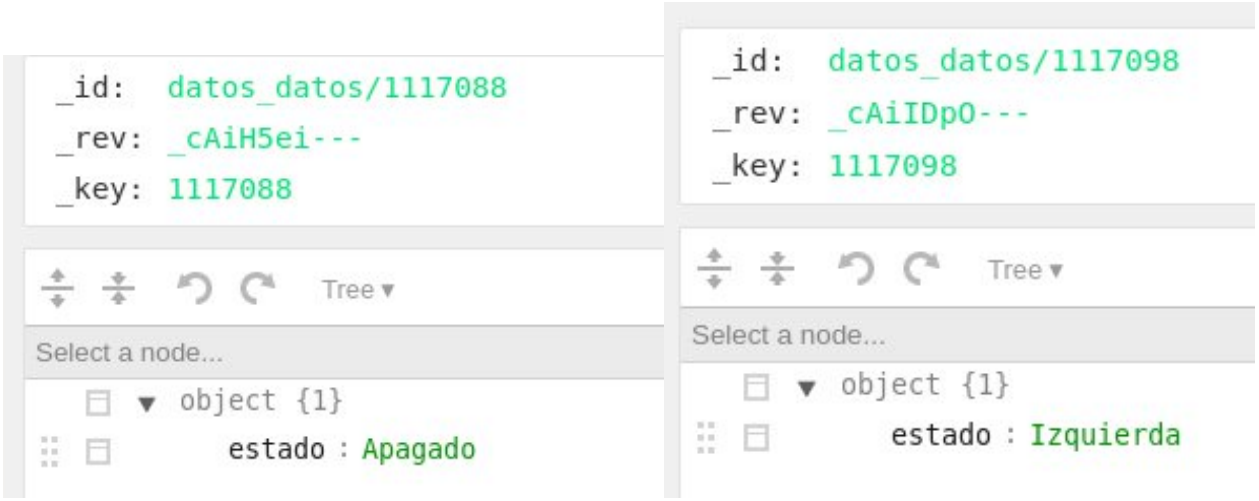
12/3/2021 Prueba.ino

```
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:*

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":"_cAiIDpO---",…	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: _cAiIDpO---
_key: 1117098
Tree
Select a node...
object {1}
estado : Izquierda