1. Planteamiento del problema

Hoy en día dado el avance y desarrollo de la tecnología y la globalización de internet, se hace necesario innovar en dispositivos que hagan uso de la red y ayude a resolver de manera más rápida problemas de la vida cotidiana. Por ello a lo largo del tiempo se ha implementado estructuras IoT basadas en gestionar y controlar remotamente elementos tales como luces, temperatura, electrodomésticos,etc.

Estos sistemas constan de varias partes, tienen un mecanismo que ayuda con la recolección de datos, continuando con un servidor que permite almacenar estos y finalmente tendrá las herramientas necesarias para interactuar con la información previamente obtenida.

Mediante estos sistemas se busca poder tener cierta inteligencia para comunicar información sobre ellos mismos o acceder a información que ha sido generada por otros objetos. Proyectos enfocados y que cuenten con bases en el internet de las cosas, proporcionan una interacción entre los mundos real-físico y digital-virtual, a través del buen manejo de esta información previamente realizando las acciones necesarias que permiten respuestas en un tiempo corto con el objetivo de poder recopilar información, consecuentemente guardarla con la ayuda de herramientas cloud y posteriormente realizar una extracción que sea usada en pro de la aplicación.

2. Objetivos

a) **Objetivo general**

Desarrollar un sistema de monitoreo remoto de temperatura y humedad inalámbrico de adquisición de datos mediante un API REST y la implementación de tarjetas ESP8266.

b) **Objetivos específicos**

- Investigar y aprender sobre el funcionamiento de las tarjeta ESP8266 y sus ventajas dentro del desarrollo del IoT.
- Realizar un API que permita la intercomunicación cliente servidor tanto local como remotamente.
- Diseñar un cliente desktop y un cliente web para el monitoreo de datos de temperatura y humedad.

3. Estado del Arte

(Irigoyen Gallego, 2018) menciona que " Se puede desarrollar un sistema de control domótico, es decir, una estructura IoT basada en gestionar y controlar remotamente los

elementos de un hogar. Basándonos en tecnologías como NodeMCU gestionaremos sensores, actuadores y desarrollaremos una comunicación con una Raspberry Pi. Esta será la que ejerza la función de servidor a través del servidor web Apache y montará una base de datos MySQL.
Con estos elementos creamos una estructura autosuficiente que nos permitirá desde cualquier lugar ver el estado del hogar e interactuar con él en tiempo real gracias a las soluciones implementadas."

Como se ha mencionado el proyecto ya realizado tiene una gran similitud al propuesto, ya que con la ayuda de las tarjetas ESP8266 se gestiona la obtención de datos tales como temperatura y humedad, posteriormente mediante un micro servicio se almacena en una base de datos. Adicionalmente esta información se la puede visualizar desde una página HTML o desde una interfaz gráfica realizada en Java. De esta manera se está realizando la monitorización de datos con la ayuda de una estructura IoT.

Según (Bucurú, 2018) " Hoy en día dado el avance y desarrollo de la tecnología y la globalización de internet, se hace necesario innovar en dispositivos que haga uso de la red y ayude a resolver de manera más rápida problemas de la vida cotidiana.
Se desarrolló un sistema de gestión y monitoreo de ambulancias, mediante el cual, el paramédico a bordo de la ambulancia puede generar un reporte del paciente que transporta de emergencias, así como también el sistema ambulancia es capaz de leer el pulso cardiaco automáticamente y registrarlo en el sistema de gestión. El sistema a bordo de la ambulancia es capaz de registrar la posición actual y enviarla automáticamente al sistema de gestión, este a su vez posee conexión a internet.
El sistema de gestión está soportado en un servidor WEB ubicado físicamente en el hospital y es accesible desde cualquier parte de internet, en el cual, el encargado del sistema de gestión en el hospital, puede ver la ubicación actual de la ambulancia y tiene constante comunicación con el paramédico a bordo, además, es capaz de ver los cambios en el pulso cardiaco medido automáticamente por el sistema ambulancia y puede ver los reportes generados de diagnóstico del paramédico, para así dar las ordenes de recibida y traslado en el hospital."

En este documento el autor muestra el beneficio totalmente que tendrá mediante su proyecto, al constatarlo con el presente proyecto las posibilidades son similares, puestos que la recolección de datos se lo puede realizar desde cualquier posición que el usuario desee, y las consultas se lo hace de manera remota en otra posición en-

contrando una gran ventaja en esto, como la descrita en el campo de la medicina, el presente proyecto proyecto lo podemos relacionar para el campo de la agricultura, de la meteorología, entre otros.

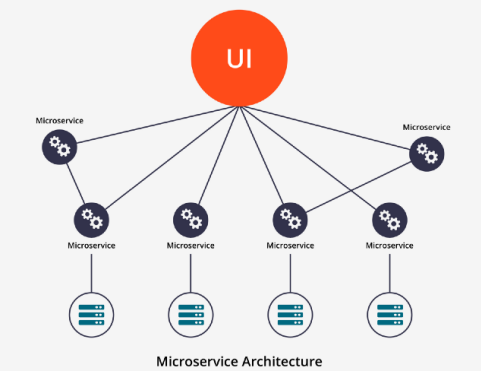
(Pardo-García, 2017) menciona en su publicación que " El desarrollo de un sistema para el monitoreo inalámbrico de variables climáticas, se realizó a partir de microcontroladores de Microchip, los cuales realizan la adquisición, almacenamiento y transmisión inalámbrica de las señales digitales. Igualmente, el microcontrolador emplea un reloj en tiempo real para saber la fecha y hora de adquisición de las muestras. El hardware también cuenta con cinco canales para la conexión de sensores y una memoria Micro SD para almacenamiento de información, junto con un módulo Wi-Fi para la supervisión inalámbrica de las variables. La información se sube a un servidor que aloja la página web, diseñada para visualizar los datos desde cualquier ordenador con conexión a internet. Adicionalmente, se desarrolló una aplicación Android que permite visualizar los datos desde dispositivos móviles con ese sistema operativo. El rendimiento del sistema fue satisfactorio, luego de comparar los datos adquiridos con los de una estación meteorológica comercial, que sirvió de patrón. Se concluye que los microcontroladores continúan siendo dispositivos adecuados para implementar sistemas de adquisición de datos, que al ser combinados con aplicativos desarrollados a la medida, brindan soluciones competitivas y a un costo razonable. "

Después de analizar el articulo del autor y comparar con el presente proyecto se puede ver como hoy en día es de vital importancia tener acceso a internet en cualquier lugar, a cualquier hora y sobre todo de cualquier objeto ya que debido a esto se puede automatizar procesos que parecen muy sencillos pero de gran importancia para una empresa o para el usuario.

4. Marco Teórico

Microservicios

Los microservicios son un “tipo de arquitectura de software” y no un API o tecnología, la cual podemos instalar y utilizar, sin embargo, sí que existen frameworks que implementan esta arquitectura para facilitarnos la vida, como es el caso de Spring Boot en Java o Express para NodeJS. La idea en la arquitectura de microservicios se centra en separar la funcionalidad en pequeñas aplicaciones independientes que puedan operar con completa autonomía, con la idea de que sean reutilizables.(Blancarte, 2015).



Jelastic PaaS

Es una de las soluciones más potentes e innovadoras del momento, un entorno Cloud PAAS (Plataform As A Service) que permite de forma rápida y sencilla la creación de entornos de trabajo que soportan diferentes lenguajes y tipos de base de datos. Una auténtica joya para desarrolladores. Cloud Jelactic Paas ofrece lo que todo desarrollador necesite para desplegar todo tipo de proyectos web. (InforTelecom, 2016).

Ventajas del Jelastic Cloud :

- **Soporta diferentes entornos y lenguajes de programación :** Con Jelastic Pass es posible desplegar proyectos webs basados en entornos como: Apache, GlassFish, IIS, Jbossas, Jetty, Nginx, Nodejs, Railo y Raptor final, además de los principales lenguajes de programación: NET, Java, Nodejs, PHP, Python, Ruby.
- **Despliegue rápido :** Tan simple como seleccionar el entorno a trabajar, los elementos necesarios y la configuración de su escalado y ya tendremos nuestro entorno cloud desplegado.
- **Escalado horizontal y vertical:** Una de las grandes ventajas de esta innovadora solución cloud es la capacidad de escalado dinámico; se utilizan solo los recursos necesarios en el momento oportuno.
- **Ahorro en costes:** ICon Jelastic Paas se paga únicamente por los recursos consumidos, siendo una opción eficaz y flexible.

MySQL

(Oracle, 2016) MySQL es la base de datos de código abierto más popular del mercado. Gracias a su rendimiento probado, a su fiabilidad y a su facilidad de uso, MySQL se ha convertido en la base de datos líder elegida para las aplicaciones basadas en web. Además, es una elección muy popular como base de datos integrada, distribuida por miles de ISV y OEM.

GearHost

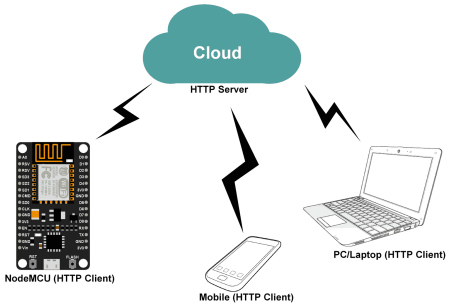
Cuenta con las siguientes características:

- **Aplicación Instantánea de Escalado:** Una interfaz que le permite a su CloudSite ser elástico a sus necesidades comerciales al instante.
- **Soporte de Base de Datos Fuerte:** GearHost es compatible con MSSQL 2016 y MySQL 5 en un entorno agrupado. Tanto las bases de datos gratuitas como las de pago están disponibles para su uso con la administración remota como SSMS o MySQL Toolbox.
- **Control de Panel Limpio y Fácil de Usar:** Una interfaz amigable para gestionar tus aplicaciones. Crear, implementar y escalar con un solo clic. Una experiencia diseñada a tu alrededor en lugar de tecnología.

Cliente Servidor con ESP8266

El cliente HTTP ayuda a enviar solicitudes HTTP y recibir respuestas HTTP desde el servidor HTTP. Se usa ampliamente en aplicaciones integradas basadas en IoT. NodeMCU es una plataforma de IoT de código abierto. Es un firmware que se ejecuta en el SoC

Wi-Fi ESP8266 de Espressif Systems. Tiene a bordo wi-fi disponible a través del cual las aplicaciones de IoT son fáciles de construir.



HTTP Client.png

Spring Boot

Spring Boot es una infraestructura ligera que elimina la mayor parte del trabajo de configurar las aplicaciones basadas en Spring.Spring Boot busca que el desarrollador solo se centre en el desarrollo de la solución, olvidándose por completo de la compleja configuración que actualmente tiene Spring Core para poder funcionar.

Características Spring Boot :

- Resolución de Dependencias
- Configuración
- Despliegue
- Métricas
- Extensible

Web HTML

HTML no es un lenguaje de programación, HTML es un lenguaje de marcado de hipertexto o “HyperText Markup Language” por el desarrollo de sus iniciales en inglés, básicamente este lenguaje se escribe en su totalidad con elementos, estos elementos están constituidos por etiquetas, contenido y atributos. HTML es un lenguaje que interpreta el navegador web para mostrar los sitios o aplicaciones web tal y como estamos acostumbrados.

Código fuente de la página web

```
<p>Esto es un párrafo de texto.</p>

<p>Esto es otro párrafo.</p>
```

Cómo se ve en el navegador

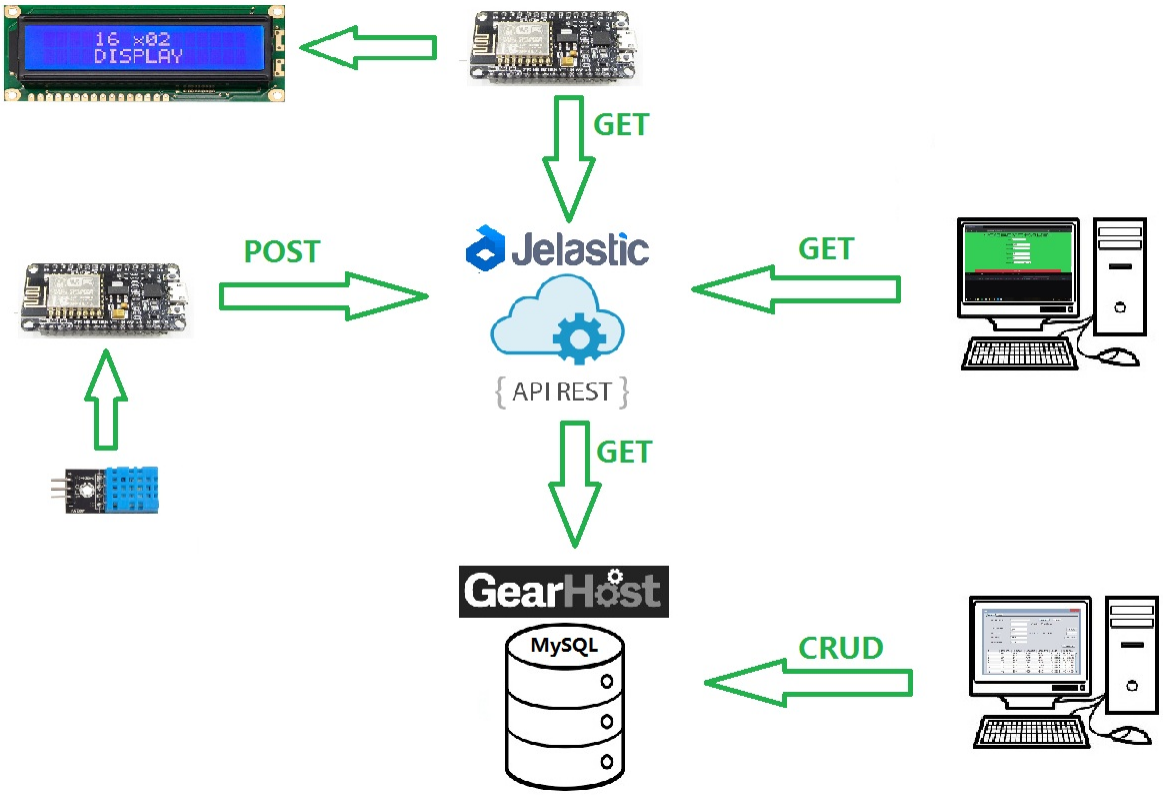
Esto es un párrafo de texto.

Esto es otro párrafo.

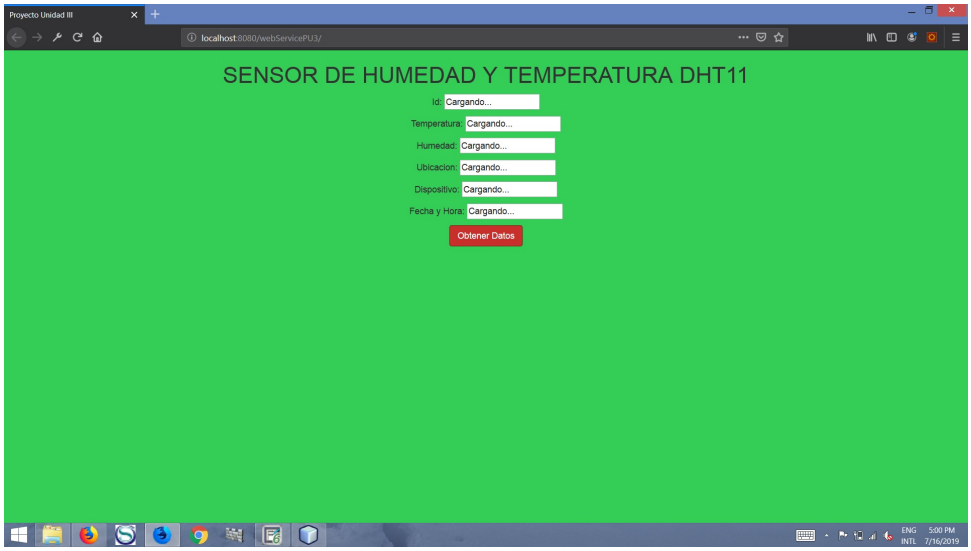
Las marcas se escriben entre desigualdades (<p>, <h1>, , etc.) y suelen ir por parejas, rodeando porciones de texto.

5. Diagrama

Diagrama Esquemático



6. Explicación del código fuente



Vista general de la página web desde la cual se puede visualizar la base de datos.

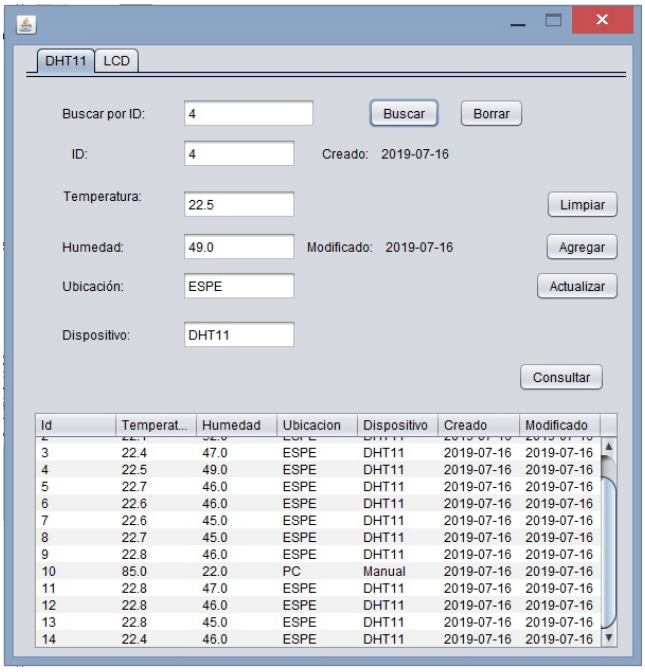
```
    var contenido = document.querySelector('#contenido')
    function traerDatos(){
        fetch('http://pu3-tec.jl.serv.net.mx/pu3/cweb')

                .then (res=> res.json())
                .then (data => {
                  console.log(data);
                  document.getElementById('ide').value = data.id;
                  document.getElementById('temperatura').value = data.temp;
                  document.getElementById('humedad').value = data.humed;
                  document.getElementById('ubicacion').value = data.ubica;
                  document.getElementById('dispositivo').value = data.dispositivo;
                  document.getElementById('fecha').value = data.creatAt;
                })
    }
</script>
```

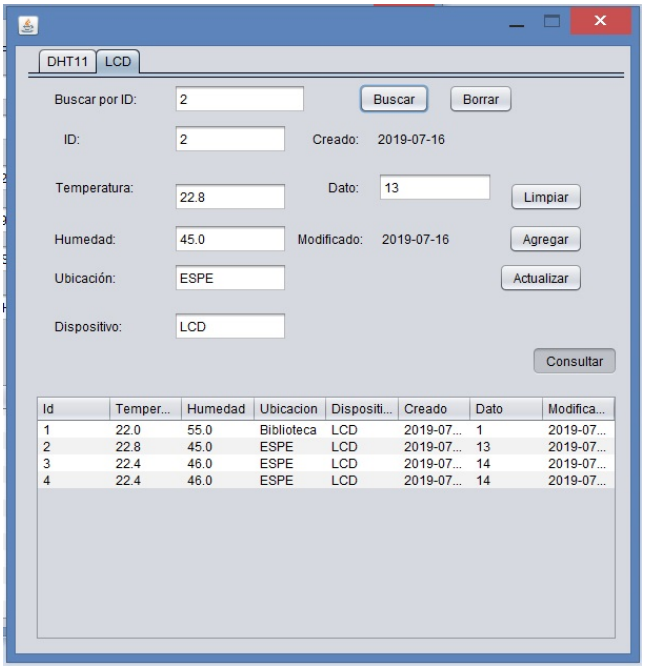
En esta parte del código se crear un script para acceder al API para poder ingresar a la base de datos y llamar al método get.

```
<h1 align="center">SENSOR DE HUMEDAD Y TEMPERATURA DHT11</h1>
<div id="informacion">
    <p align="center">Id: <input type="text" id="ide" value="Cargando..."/></p>
    <p align="center">Temperatura: <input type="text" id="temperatura" value="Cargando..."/></p>
    <p align="center">Humedad: <input type="text" id="humedad" value="Cargando..."/></p>
    <p align="center">Ubicacion: <input type="text" id="ubicacion" value="Cargando..."/></p>
    <p align="center">Dispositivo: <input type="text" id="dispositivo" value="Cargando..."/></p>
    <p align="center">Fecha y Hora: <input type="text" id="fecha" value="Cargando..."/></p>
</div>
<div class="container my-5 text-center">
    <button class="btn btn-danger w-100" onclick="traerDatos()">Obtener Datos</button>
    <div class="mt-5" id="contenido">
    </div>
</div>
```

En este segmento de código se comienza a crear la parte visual de la página web y se setea los valores que el script obtiene.

Controlador Base de Datos





Vistas generales de la Interfaz gráfica del Controlador de la Base de Datos.

Paquete Controlador

```
public PU3(String database, String usu, String pass) throws Exception {

    this.url ="jdbc:mysql://den1.mysql1.gear.host/"+database;;
    this.usu=usu;
    this.pass=pass;

    //Cargamos el driver
    DriverManager.registerDriver(new org.gjt.mm.mysql.Driver());
    System.out.println("Driver cargado...");

    //Establecemos la conexion
    conexion = DriverManager.getConnection(url, usu, pass);
    System.out.println("Conexion establecida".toUpperCase());

    //Creamos la instancia
    sentencias = conexion.createStatement();
    System.out.println("Instancia creada!\n");
}
```

Se inicia la Conexión con la base de datos,se le otorga el url de GearHost, el password y el usuario.

```
public ArrayList<Object> consultarTabla(String tabla) throws Exception{

    ArrayList<Object> aux= new ArrayList<Object>();

    resultado = sentencias.executeQuery("select * from "+tabla+ " ;");
    while(resultado.next()){
        if(tabla=="lcd"){
            Lcd aux1=new Lcd();
           aux1.setDato(resultado.getLong("dato"));
           aux1.setId(resultado.getLong("id"));
           aux1.setFecha(resultado.getDate("creat_At"));
           aux1.setDispositivo(resultado.getString("dispositivo"));
           aux1.setTemp(resultado.getDouble("temp"));
           aux1.setHumed(resultado.getLong("humed"));
           aux1.setUbica(resultado.getString("ubica"));
           aux1.setUpdate(resultado.getDate("update_At"));
           aux.add(aux1);
    }
        if(tabla=="dht11"){
            Sensor aux2=new Sensor();
           aux2.setId(resultado.getLong("id"));
           aux2.setFecha(resultado.getDate("creat_At"));
           aux2.setDispositivo(resultado.getString("dispositivo"));
           aux2.setTemp(resultado.getDouble("temp"));
           aux2.setHumed(resultado.getLong("humed"));
           aux2.setUbica(resultado.getString("ubica"));
           aux2.setUpdate(resultado.getDate("update_At"));
           aux.add(aux2);
```

Se crea un método Consultar Tabla donde se va a obtener cada atributo de la base de datos tanto como la tabla LCD y la tabla sensor.

```
//    Guarda la informacion
    public boolean guardarInformacion(Object objeto, String tabla) throws SQLException{

        if(tabla=="lcd"){
            Lcd aux1 = (Lcd)objeto;

            //insert into lcd values (105,CURRENT_TIMESTAMP,55,"Manual",22.6,85,"PC",CURRENT_TIMESTAMP );
            sentencias.executeUpdate("insert into "+tabla+" values("
                    +aux1.getId()+",CURRENT_TIMESTAMP,"
                    +aux1.getDato()+",\""+aux1.getDispositivo()+"\","+aux1.getHumed()+","
                    +aux1.getTemp()+",\""+aux1.getUbica()+"\",CURRENT_TIMESTAMP);");
            return true;

        }

        if(tabla=="dht11"){
            Sensor aux2 = (Sensor)objeto;
            //insert into Productos values('1','Patatas','200 Pta','kilo');
            sentencias.executeUpdate("insert into "+tabla+" values("
                    +aux2.getId()+",CURRENT_TIMESTAMP,\""+aux2.getDispositivo()+"\","+aux2.getHumed()+","
                    +aux2.getTemp()+",\""+aux2.getUbica()+"\",CURRENT_TIMESTAMP);");
            return true;
        }

         return false;
    }
//
```

Se crea un método para poder añadir un nuevo elemento a la base de datos, lo que se hace es obtener los parámetros escritos en cada campo de texto y se envía una sentencia Mysql para que se guarde directamente en la base de datos.

```
//Borrar informacion
public boolean borrarInformacion(Object objeto, String tabla) throws SQLException{

    if(tabla=="lcd"){
        Lcd aux1 = (Lcd)objeto;
        //delete from Clientes where codigo='3';
        sentencias.executeUpdate("delete from "+tabla+" where id='"+aux1.getId()+"';");
        return true;
    }
    if(tabla=="dht11"){
        Sensor aux2 = (Sensor)objeto;
        //delete from Clientes where codigo='3';
        sentencias.executeUpdate("delete from "+tabla+" where id='"+aux2.getId()+"';");
        return true;
    }
    return false;
}
//
```

Se crea el método Borrar este lo que hace es enviar el id del elemento que se desea borrar y la sentencia Mysql para que se borre directamente en la base de Datos.

```
//Busca informacion
public boolean buscarInformacion(Object objeto, String tabla) throws SQLException{

    if(tabla=="lcd"){
        Lcd aux1 = (Lcd)objeto;
        //select * from Clientes where codigo='3';
        resultado=sentencias.executeQuery("select * from lcd where id='"+aux1.getId()+"';");
        while(resultado.next()){
            aux1.setDato(resultado.getLong("dato"));
           aux1.setId(resultado.getLong("id"));
           aux1.setFecha(resultado.getDate("creat_At"));
           aux1.setDispositivo(resultado.getString("dispositivo"));
           aux1.setTemp(resultado.getDouble("temp"));
           aux1.setHumed(resultado.getLong("humed"));
           aux1.setUbica(resultado.getString("ubica"));
           aux1.setUpdate(resultado.getDate("update_At"));
        }
        return true;
    }
    if(tabla=="dht11"){
        Sensor aux2 = (Sensor)objeto;
        //select * from Productos where referencia='3';
        resultado=sentencias.executeQuery("select * from dht11 where id='"+aux2.getId()+"';");
        while(resultado.next()){

           aux2.setId(resultado.getLong("id"));
           aux2.setFecha(resultado.getDate("creat_At"));
           aux2.setDispositivo(resultado.getString("dispositivo"));
           aux2.setTemp(resultado.getDouble("temp"));
           aux2.setHumed(resultado.getLong("humed"));
           aux2.setUbica(resultado.getString("ubica"));
           aux2.setUpdate(resultado.getDate("update_At"));
        }
```

Para la búsqueda de información lo que se hace es enviar el id del elemento y la base de datos nos retorna todos los atributos de ese elemento y se procede a setear dichos atributos en los campos de texto correspondientes.

```
//Actualiza
public boolean actualizarInformacion(Object objeto, String tabla,Long id) throws SQLException{
    if(tabla=="lcd"){
        Lcd aux1 = (Lcd)objeto;
        //update Clientes set nombre='hola' , direccion='ecu' where codigo='5';
        //update lcd set  update_at=CURRENT_TIMESTAMP,id=1254, dato=666,temp=12.12,humed=5.5,dispositivo="Update",ubica="La colina" where id=10;

        sentencias.executeUpdate("update lcd set update_at=CURRENT_TIMESTAMP, id="
                +aux1.getId()+", dato="
                +aux1.getDato()+", temp="+aux1.getTemp()+",humed="+aux1.getHumed()+",dispositivo=\""
                +aux1.getDispositivo()+"\",ubica=\""+aux1.getUbica()+"\" where id='"
                +id+"';");
        return true;
    }
    if(tabla=="dht11"){
        Sensor aux3 = (Sensor)objeto;
        //update Productos set nombre='Sandias' , precio='750 Pta' , concepto='libra' where referencia='3';
        sentencias.executeUpdate("update dht11 set update_at=CURRENT_TIMESTAMP, id="
                +aux3.getId()+",temp="+aux3.getTemp()+",humed="+aux3.getHumed()+",dispositivo=\""
                +aux3.getDispositivo()+"\",ubica=\""+aux3.getUbica()+"\" where id='"
                +id+"';");
        return true;
    }
    return false;
}
```

En el método actualizar se manda una sentencia Mysql para que las tablas se actualicen cada cierto tiempo.

Clase LCD

```java
public class Lcd {
    private Long id;
    private Date fecha;
    private Long dato;
    private String dispositivo;
    private double temp;
    private double humed;
    private String ubica;
    private Date update;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
```

```java
public class Sensor {
    private Long id;
    private Date fecha;
    private String dispositivo;
    private double temp;
    private double humed;
    private String ubica;
    private Date update;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
```

En esta clase se setea los atributos del lcd con sus getters y setters respectivos, lo mismo se procede con la clase sensor.

Interfaz Gráfica

```
public class PU3 extends javax.swing.JFrame {
    public controlador.PU3 obj = new controlador.PU3("tecpu3","tecpu3","Gn46_uMTv_xJ");
        Lcd aux=new Lcd();
        Sensor aux1=new Sensor();
    /**
     * Creates new form PU3
```

Se crea un objeto de tipo controlador donde se debe enviar el usuario, el password y el nombre de la base de datos también secrea un objeto de tipo LCD y Sensor de cada clase respectivamente.

```
private void btnConsultarActionPerformed(java.awt.event.ActionEvent evt) {
    ArrayList<Object> aux = null;
    try {
        aux = obj.consultarTabla("lcd");
    } catch (Exception ex) {
        Logger.getLogger(PU3.class.getName()).log(Level.SEVERE, null, ex);
    }
    vaciarTabla("lcd");
    llenarTabla(aux,"lcd");

}
```

Cuando se presione el botón consultar se llama al método de consultar, se envía el nombre de la tabla de la cual se va a consultar y lo mismo se hace con cada botón y su respectivo método.

API

```
@RestController
@RequestMapping(path="/pu3")

public class SensorController {

    @Autowired //identidifca el nombre del repositorio
    SensorRepository sensorRepository;
    @Autowired
    LcdRepository lcdRepository;

    @PostMapping(path="/sensor")
    public Sensor createNote(@Valid @RequestBody Sensor sensor) {
        return sensorRepository.save(sensor);
    }

    @GetMapping(path="/cweb")
    public  Optional<Sensor> getAll(){
        return sensorRepository.findById(sensorRepository.count());
    }

    @GetMapping(path="/notes/last")
    public @ResponseBody String getForLcd(){
        Sensor aux=sensorRepository.getOne(sensorRepository.count());
        Lcd aux1= new Lcd();
        aux1.setTemp(aux.getTemp());
        aux1.setHumed(aux.getHumed());
        aux1.setUbica(aux.getUbica());
        aux1.setCreatAt(aux.getCreatAt());
        aux1.setUpdateAt(aux.getUpdateAt());
        aux1.setDispositivo("LCD");
        aux1.setDato(aux.getId());
        lcdRepository.save(aux1);
        return aux.getHora()+","+Double.toString(aux1.getTemp())+","+Double.toString(a
```

Aquí se crea un controlador por cada elemento este es el controlador de la tabla del sensor, se puede ver los métodos de encontrar por id, obtener el último id y guardar un nuevo elemento.

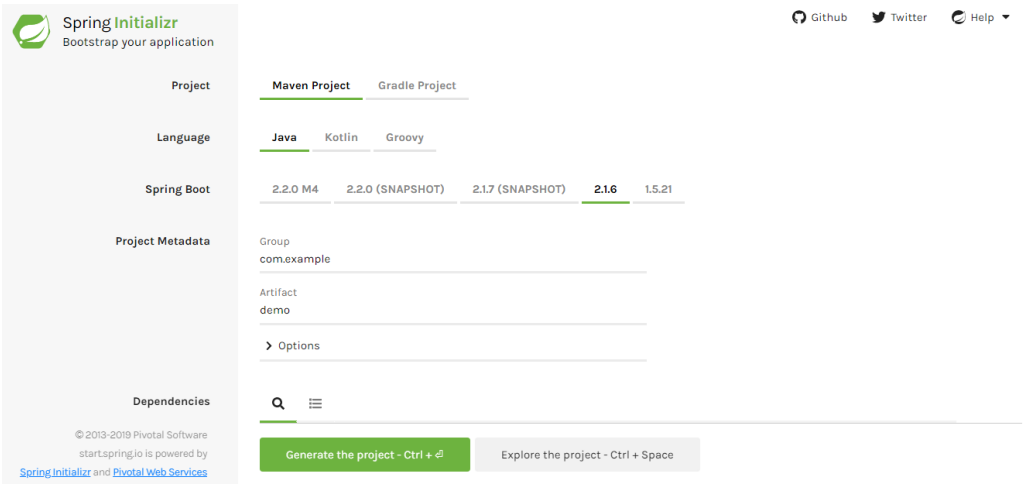
SensorController.java Lcd.java

```
51
52          //fecha en la cual se actualizo
53          @Column(nullable=false, updatable=false)
54          @Temporal(TemporalType.TIMESTAMP)
55          @CreatedDate
56          private Date creatAt;
57
58          //Fecha en la cual se creo el dato
59          @Column(nullable=false)
60          @Temporal(TemporalType.TIMESTAMP)
61          @LastModifiedDate
62          private Date updateAt;
63
64          public Long getId() {
65              return id;
66          }
67
68          public void setId(Long id) {
69              this.id = id;
70          }
71
72          public double getTemp() {
73              return temp;
74          }
75
76          public void setTemp(double temp) {
77              this.temp = temp;
78          }
79
80          public double getHumed() {
81              return humed;
82          }
83
84          public void setHumed(double humed) {
85              this.humed = humed;
86          }
```

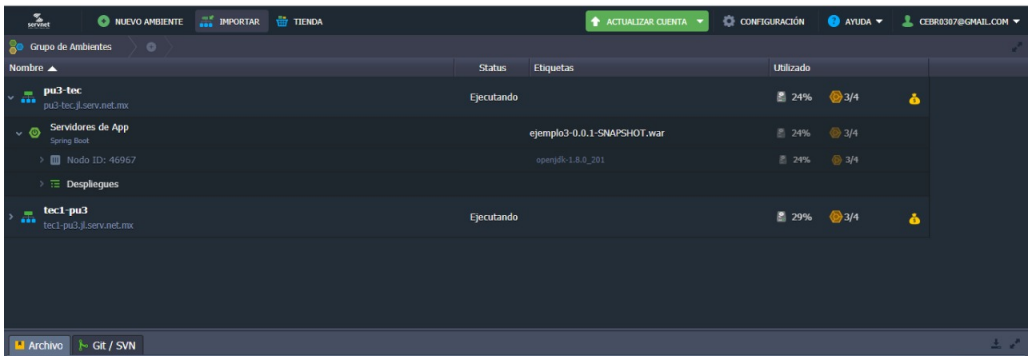
Se crea una clase Lcd con sus atributos y sus setters y getters, lo mismo se procede con la clase sensor.

7. Descripción de Prerrequisitos y Configuración

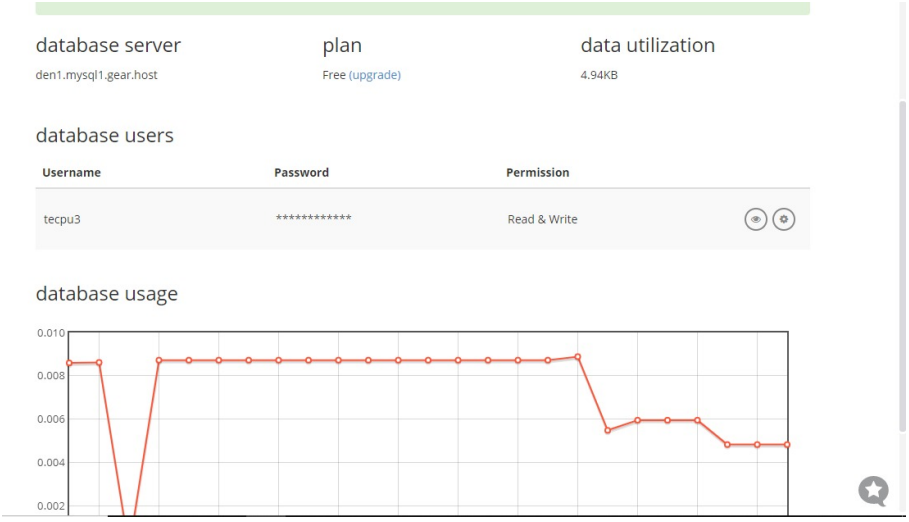
Primeramente para la creación del proyecto de microservicio con SpringBoot, se parte de una congifuración inicial que nos proporcionan los desarrolladores de la herramienta Spring Initializr.



Adicionalmente debemos crear una cuenta en Jelastic para subir nuestra API y que pueda ejecutarse en línea.

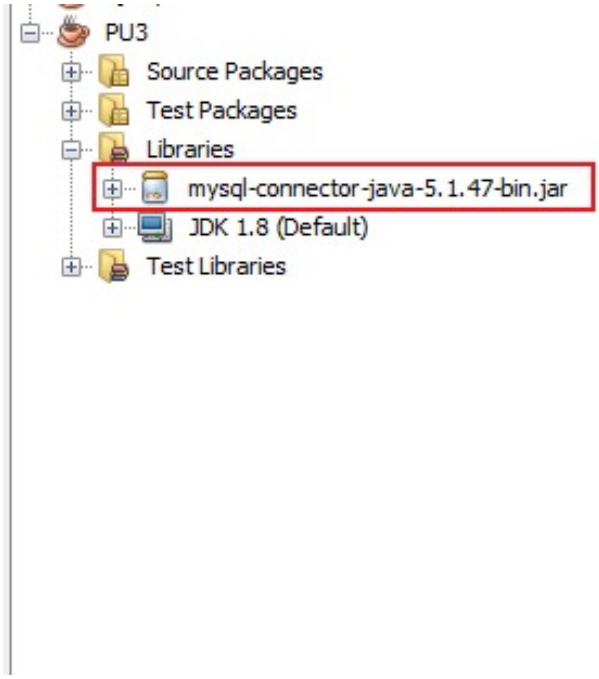


También creamos una cuenta en GearHost el cual permitirá el alojamiento en la nube de nuestra base de datos.



JAVA

Incluímos el MySQL CONECTOR que es el controlador oficial JDBC para MySQL en la carpeta LIBRARIES de nuestro proyecto ÇlienteDesktop"que nos proporcionará el medio para comunicarse con las bases de datos.



Arduino

Inlcluimos la libreria "Separador.h"la cual permite al momento de extraer los datos y posteriormente escribirlos en el LCD estos datos tengan una separación, genera una presentación más estética al proyecto.

PU3.ClientHTTP.GET §

```
#include <Separador.h>
#include <ESP8266WiFi.h> //modulo wifi
#include <ESP8266HTTPClient.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define WIFI_SSID "CESAR"    //"CALLE_STEP2"
#define WIFI_PASSWORD "cebr0307"  //"483572@A"
String cabecera="Hora: Temp: Hum:" ;
Separador s;


LiquidCrystal_I2C lcd(0x27,2,1,0,4,5,6,7,3, POSITIVE);
```

8. Conclusiones

- Se concluye que hoy en día es un gran beneficio tener conocimiento sobre el manejo de tarjetas ESP8266 ya que con ellas se puede crear servidores remotos los cuales ayuden a la automatización de procesos.
- Luego de realizar el presente proyecto se determinó que utilizar un API REST para realizar una comunicación de datos es de gran ayuda ya que al momento de hostear en la nube se tiene acceso a ella desde cualquier lugar con acceso a internet.
- Se puede concluir que el monitoreo y el control del sensor de temperatura puede tornarse un poco tedioso para lo cual se automatizo este trabajo diseñando un cliente desktop el cual puede controlar los datos en tiempo real desde la base de datos pero a su ves se realizo un cliente web el cual obtiene el ultimo valor adquirido y lo presenta mediante una pagina web a la cual se puede tener acceso remotamente desde cualquier lugar con acceso a internet, automatizando y agilizando así el proceso de control y monitoreo de datos.

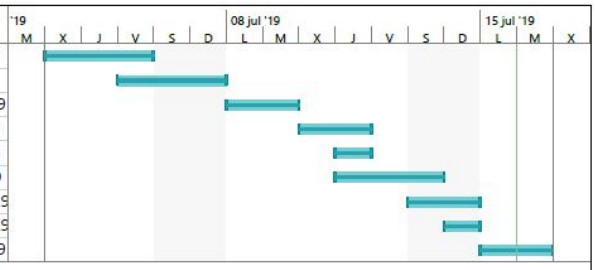
9. Recomendaciones

- Se concluye que hoy en día es un gran beneficio tener conocimiento sobre el manejo de tarjetas ESP8266 ya que con ellas se puede crear servidores remotos los cuales ayuden a la automatización de procesos.
- Luego de realizar el presente proyecto se determinó que utilizar un API REST para realizar una comunicación de datos es de gran ayuda ya que al momento de hostear en la nube se tiene acceso a ella desde cualquier lugar con acceso a internet.
- Se puede concluir que el monitoreo y el control del sensor de temperatura puede tornarse un poco tedioso para lo cual se automatizo este trabajo diseñando un cliente desktop el cual puede controlar los datos en tiempo real desde la base de datos pero a su ves se realizo un cliente web el cual obtiene el ultimo valor adquirido y lo presenta mediante una pagina web a la cual se puede tener acceso remotamente desde cualquier lugar con acceso a internet, automatizando y agilizando así el proceso de control y monitoreo de datos.

10. **Cronograma**

Diagarama Gantt

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		Busqueda de Información	3 días	mié 03/07/19	vie 05/07/19
2		Segmentar Información	2 días	vie 05/07/19	dom
3		Planteamiento de	2 días	lun 08/07/19	mar 09/07/19
4		Elaboración de Diagramas	2 días	mié 10/07/19	jue 11/07/19
5		Analizis de Diagramas	1 día	jue 11/07/19	jue 11/07/19
6		Programación del Código	3 días	jue 11/07/19	sáb 13/07/19
7		Informe	2 días	sáb 13/07/19	dom 14/07/19
8		Paper	1 día	dom 14/07/19	dom 14/07/19
9		Revisión completa	2 días	lun 15/07/19	mar 16/07/19



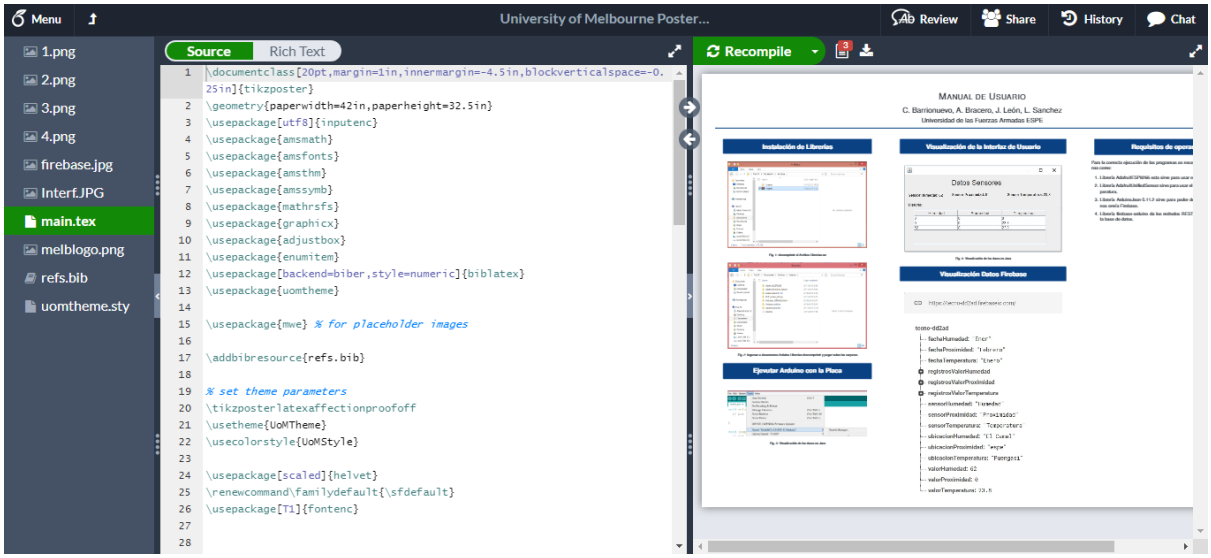
Proyecto: CronoPU3
Fecha: mar 16/07/19

Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Progreso
Progreso manual

11. Herramientas Cloud

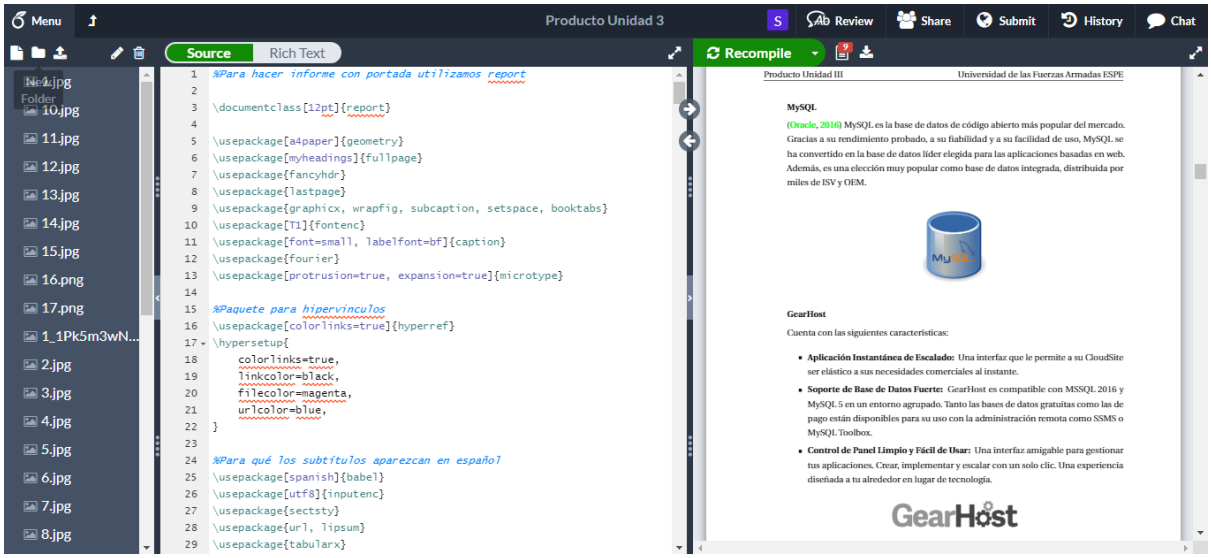
Plotter

https://www.overleaf.com/read/nkxttmndtkqr



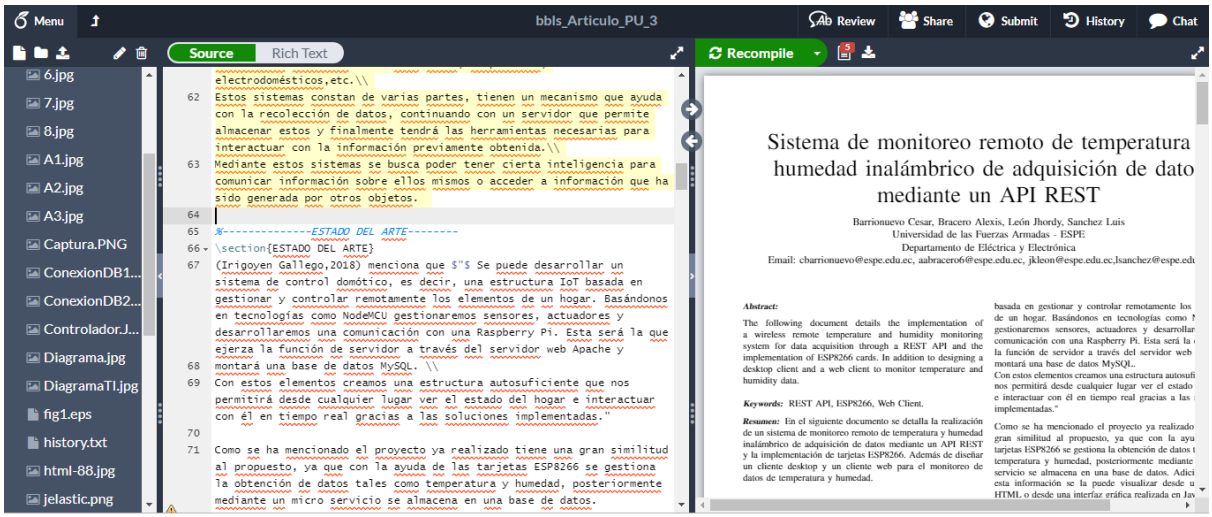
Informe

https://es.overleaf.com/read/kxdvnhrppqzb



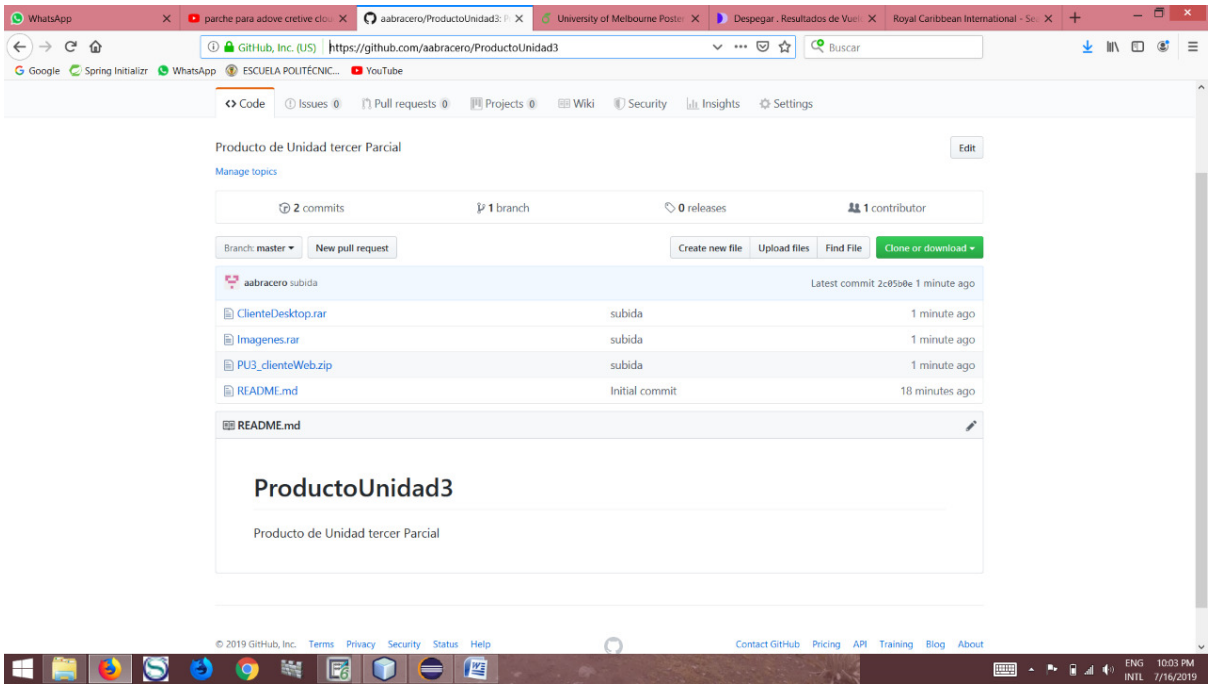
Paper

https://www.overleaf.com/read/tcjcrxjnxwnp



GitHub

https://github.com/aabracero/ProductoUnidad3



Referencias

Blancarte, O. (2015). Software architect. *IEEE*.

Bucurú, B. A. B. (2018). DiseÑo y desarrollo de sistema de gestiÓn ambulancia-hospital para enviar datos personales, de ubicaciÓn y pulso cardiaco del paciente. *IEEE*, *3*(7).

InforTelecom. (2016). ¿qué es jelastic paas? *IEEE*.

Irigoyen Gallego, R. (2018). Internet de las cosas. sistema electrónico de control basado en arduino. *Universitat Oberta de Catalunya*.

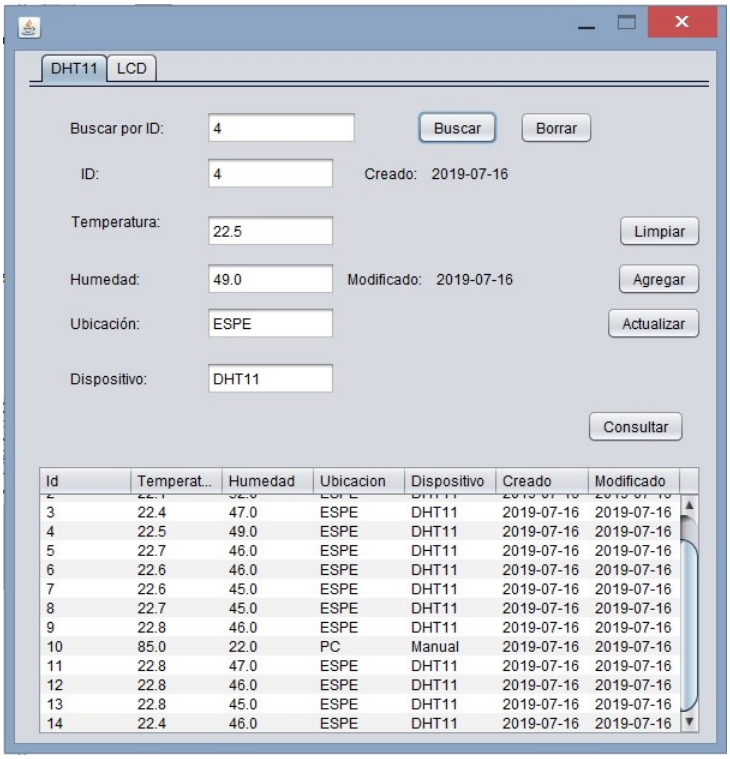
Oracle. (2016). La base de datos de código abierto más popular del mercado. *IEEE*.

Pardo-García, D. C. R.-A. C. A. V.-H. A. (2017). Monitoreo de variables meteorológicas a través de un sistema inalámbrico de adquisición de datos. *IEEE*.

12. Anexos

Manual de Usuario

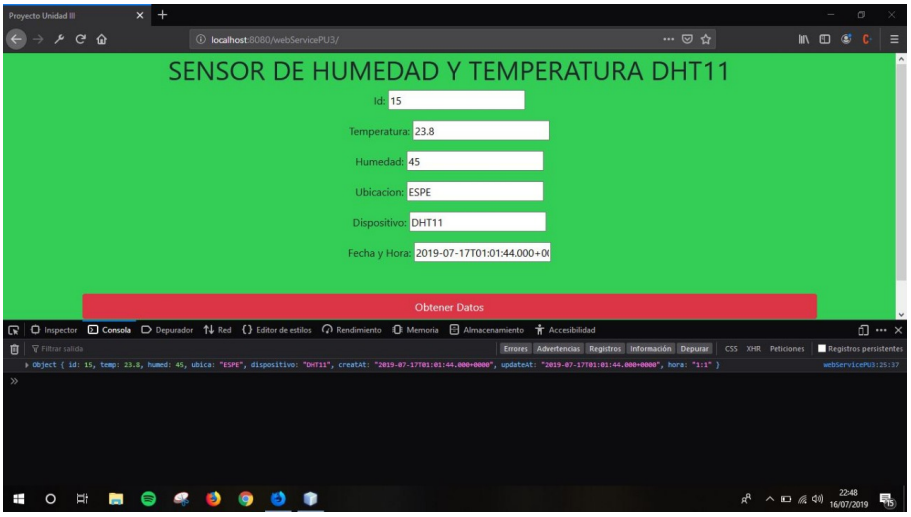
1.- Abrir el Cliente Desktop y ejecutarlo.



Esta es la pantalla del controlador de base de datos

- En esta pantalla tenemos el controlador para la tabla del sensor y el controlador para la tabla del lcd.
- Tenemos los botones Buscar, Borrar, Limpiar, Agregar, Actualizar y Consultar.
- En el botón Buscar lo que hacemos es escribir en el campo de texto Buscar por ID ponemos el ID del que queremos modificar eliminar o buscar.
- Los demás botones cumplen la función descrita por su propio nombre.

2.- Abrir el cliente Web ya sea ejecutando el código o mediante el path `http://localhost:8080/webServicePU3`



- Al momento de realizar la consulta, podremos visualizar en cada casilla la información correspondiente al último dato recolectado por el sensor.

Nota: Se debe tener en cuenta que el servidor del JPS debe estar corriendo localmente en el dispositivo desde el cual se quiere acceder a la página web.