Imagen que contiene exterior, cielo, nieve, agua

Descripción generada con confianza muy alta

**ESCUELA SUPERIOR POLITÉNICA DEL LITORAL**

**PROYECTO DE APLICACIONES MÓVILES Y SERVICIOS TELEMÁTICOS**

**MANUAL TÉCNICO**

EXPEDIA TISCH

RESTAURANTE INTELIGENTE

**MG. ADRIANA COLLAGUAZO**

**GRUPO # 2**

Christin Ochoa Hoover

Danilo Torres Intriago

Javier Benavides Vilema

José Espinales Chiriguayo

**PARALELO 1**

2T – 2019

**Manual de Implementación Técnica del Proyecto**

El siguiente manual técnico expone los detalles presentes en el proyecto realizado. Este proyecto tiene como fin comunicar la disponibilidad de las mesas de un restaurante mediante el uso de un sensor realizado utilizando velostat y tela conductiva, detalles que serán expuestos a lo largo de este manual.

**Introducción del sistema implementado**

El sistema mostrado a continuación describe la funcionalidad general de la solución. La implementación parte con la base de que, al usuario encargado de las mesas de un restaurante, se le debe informar de la disponibilidad de estas mediante una aplicación en tiempo real. Para ello fue necesario el desarrollo de un sensor de presión que se ubicará debajo de una mesa, para lo cual fue necesario el establecimiento de umbrales de presión mismos que conforman parte de la estructura lógica de decisión sobre un cambio de disponibilidad en el restaurante. Este sensor se conecta a una placa de desarrollo Arduino y este a su vez a un módulo Thinxtra. Este último modulo nos permite realizar envió de notificaciones mediante la red celular hacia la plataforma de Sigfox, misma que mediante métodos preconfigurados denominados Callbacks realizan el envío de información hacia una base de datos levantada en la nube a través del Herokuapp. El usuario final, es decir el encargado de las mesas, podrá administrar en tiempo real las mesas del restaurante gracias a la información en tiempo real que puede obtener de la base de datos que está en constante cambio debido a la acción del código implementado en Arduino para generar cambios siempre y cuando exista alguna variación importante de disponibilidad.

1. **Diagrama del sistema**

A close up of a map

Description generated with high confidence

**2.** **Recursos**

**Para realizar el proyecto se necesitó recursos como hardware y software, mostrados en la siguiente tabla:**

|  |  |
| --- | --- |
| Hardware | Software |
| Ibutton ($120) | Android Studio |
| 15 jumpers ($1) | IDE Arduino |
| Arduino UNO ($20) | Herramienta colaborativa Asana |
| Velostat ($5) | Plaforma SigFox |
| Relé de Arduino ($4) | Base de datos en Firebase |
| Pila de 9 V ($5) |  |
| Dos tiras de tela conductiva ($10 lamina A4) |  |
| Esponja ($0.80) |  |
| Cable UTP Cat-7 ($1) |  |
| Cinta aislante ($0.50) |  |
| Cartón prensado 10g ($1.50) |  |
| $168.00 |  |

**3.** **Historias de Usuario**

**Historia de usuario 1:** Como encargado de piso requiero poder consultar la cantidad y ubicación de mesas disponibles para poder dirigir a los clientes que esperan ser atendidos a una mesa.

**Criterio de aceptación 1:** Dado que le sistema IoT se encuentra censando el suelo donde se encuentran las sillas (sensor de presión) cuando el usuario de clic en “Ver mesas" entonces aparecerán las ubicaciones de las mesas libres (con verde) y mesas ocupadas (con rojo).

**Criterio de aceptación 2:** Dado que le sistema IoT se encuentra censando el suelo (sensor de presión) y no se encuentras mesas disponibles cuando el usuario de clic en “Ver mesas" entonces aparecerán las ubicaciones de las mesas y el mensaje “No se encuentra mesas disponibles”.

**Criterio de aceptación 3:** Dado que le sistema IoT se encuentra censando el suelo (sensor de presión) y no se dispone de conexión cuando el usuario de clic en “Ver mesas" entonces aparecerá el siguiente mensaje “No se dispone de conexión de internet” y se reinicia la comunicación Sigfox.

**Historia de usuario 2:** Como encargado de piso requiero indicar que estoy esperando una mesa (cualquiera) y recibir una notificación cuando una mesa sea liberada para para ubicar a un nuevo cliente.

**Criterio de aceptación 1:** Dado que todas las mesas se encuentran ocupadas cuando seleccione la opción “Notificar cuando una mesa esté disponible” y una mesa se libera entonces llegara una notificación indicando "Una mesa se encuentra disponible”.

**Criterio de aceptación 2:** Dado que existen mesas disponibles cuando seleccione la opción “Notificar cuando una mesa esté disponible” entonces llegara una notificación indicando "Actualmente existen mesas disponibles”.

**Historia de usuario 3:** Como usuario requiero encender mi dispositivo IoT e iniciar comunicaciones para comenzar a utilizar el sistema.

**Criterio de aceptación 1:** Dado que mi dispositivo IoT tiene una fuente de poder estable cuando se presione el botón de encendido en el dispositivo entonces se establecerá la comunicación Sigfox enviando un mensaje indicando el estado del dispositivo y aparecerá en la aplicación móvil “Sistema listo para usar”.

**Criterio de aceptación 2:** Dado que mi dispositivo IoT tiene una fuente de poder estable cuando se presione el botón de encendido en el dispositivo, pero no puede establecer comunicación Sigfox entonces no se enviará el mensaje inicial indicando el estado del dispositivo y se presentará en la aplicación móvil “No se puede establecer comunicación con la nube”.

**Historia de usuario 4:** Como usuario requiero apagar mi dispositivo IoT cuando termine de usarlo para poder ahorrar energía de la batería.

**Criterio de aceptación 1:** Dado que mi dispositivo IoT tiene una fuente de poder estable y no necesito que siga encendido cuando se presione el botón de apagado en el dispositivo entonces deberá terminar la conexión Sigfox, interrumpir la conexión con la batería y mostrar en la aplicación móvil “Estado del dispositivo - Apagado”.

**4.** **Diagrama de Despliegue**

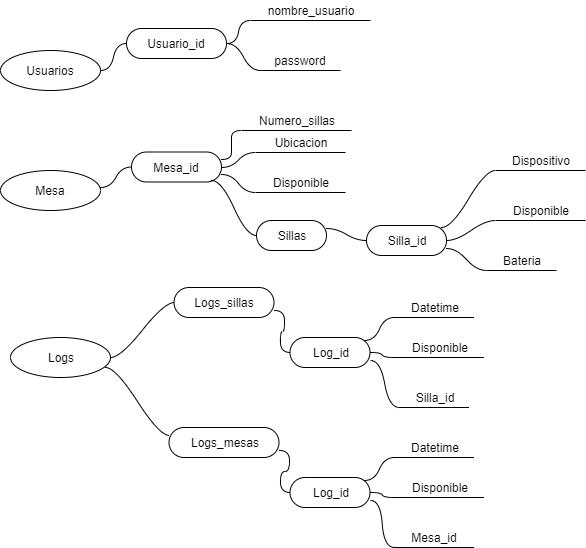
El diagrama de despliegue mostrado a continuación explica acerca de la interacción entre los diferentes módulos que forman parte del proyecto. El primer módulo que se observa es el dispositivo Android, el cual contendrá la aplicación y se podrán observar los datos almacenados en la base de datos. Segundo se encuentra la base de datos la cual almacenara los datos enviados por la plataforma de Sigfox. Tercero, está la plataforma de Sigfox la cual enviara los datos capturados por el dispositivo IoT mediante los callbacks a la base de datos. Por último, se encuentra el dispositivo IoT que permite obtener los valores de los sensores y los enviara como mensajes a la plataforma de Sigfox.

A close up of a sign

Description automatically generated

**5.**    **Modelo de la Base de Datos**

Al ser una base de datos no relacional describimos la estructura de árbol con el siguiente diagrama. Donde tanto Mesa como Sillas pueden tener varios Mesa\_id y Silla\_id respectivamente.

****

**6.** **Diagrama de Actividades**

Describe la interacción del usuario mediante el aplicativo móvil con el dispositivo IoT, pudiendo identificar las acciones que mediante la aplicación se espera el usuario pueda realizar. Citando una de ellas, cada usuario debe poder ingresar en la aplicación, y una vez validado su ingreso podrá hacer solicitudes indirectas a la base de datos sobre la información en tiempo real del estado de las mesas en el restaurante, pero también podrá revisar la batería del sistema que permite dicha notificación de información remota.

A close up of text on a white background

Description generated with high confidence

**7.** **Diagrama de circuito**

Para el circuito se utilizo el dispositivo IoT (ibutton) acoplado al Relé de Arduino, el cual mediante una señal de entrada determinaba el envió de mensajes del IoT. La señal de entrada era establecida por el Arduino UNO, quien leía en el puerto Ao la variación de voltaje que generaba el dispositivo creado.

Si el dispositivo no era presionado enviaba 5 voltios al puerto Ao por lo que no pasaba nada, pero al ser presionado me variaba el voltaje por la salida digital A7 me enviaba una señal para que el relé se cierre.

Para que el Arduino ejecute dichas funciones fue necesario establecer un código previamente, donde se determino que al existir la variación de voltaje menor o mayor a cuatro se enviaba la señal para cerrar el Relé. Adicionalmente se implementó un led indicador de voltaje, el cual se encenderá únicamente si el voltaje es menor a 5, el cual es requerido para que el circuito funcione correctamente y el envío de mensajes de Ibutton sea de una manera eficiente.

A close up of a device

Description automatically generated

**8.** **Bases de Datos**

La base de datos está alojada en firebase a traves del enlace <https://console.firebase.google.com/u/0/project/restaurante-inteligente-2/database/restaurante-inteligente-2/data>. Esta almacena los datos de disponibilidad de las mesas, sillas y recibe los registros de sigfox.

**9.** **Explicación del código fuente Arduino desarrollado con los comentarios correspondientes**

Para la implementación correcta del circuito, el Arduino Uno era el controlador principal del proyecto por lo que se generó un código a partir de la historia de usuarios requeridas, permitiéndome obtener el estado actual de la silla y controlar el voltaje que le es suministrado al Arduino.

1. **Explicación del código**

**Archivo:** Restaurante\_Inteligente.ino

Declaramos las constantes con los pines donde conectamos los LEDs y el pin analógico.

#define LEDROJO 4

#define ANALOGPILA 0

Lo siguiente es declarar las variables y los umbrales. Estos últimos dependerán del tipo de pila. En este caso se incorporó una pila 9V.

int analogValor = 0;

float voltaje = 0;

int ledDelay = 800;

float minimo = 5;

**Función setup**

En la función *setup* inicializamos una sola vez en el programa, definiendo los pines de salida a utilizar en el Arduino.

void setup() {

pinMode(7,OUTPUT);

Serial.begin(9600);

digitalWrite(7,HIGH);

pinMode(LEDROJO, OUTPUT);

}

**Función loop**

Comenzamos la función loop que se repetirá continuamente, validamos las variables a utilizar.

void loop() {

int sensorValue = analogRead(A0);

float voltage = sensorValue \* (5.0 / 1023.0);

analogValor = analogRead(ANALOGPILA);

voltaje = 0.0048 \* analogValor;

Serial.print("Voltaje: ");

Serial.println(voltaje);

En la siguiente parte vamos a enviar una señal para cerra el relé. Esto lo hacemos a través de las estructuras condicionales *if*.

if (voltage<4){

digitalWrite(7,LOW);

delay (3000);

digitalWrite(7,HIGH);}

Serial.println(voltage);

while (voltage<4){

sensorValue = analogRead(A0);

voltage = sensorValue \* (5.0 / 1023.0);

Serial.println(voltage);

}

if (4<voltage){

digitalWrite(7,LOW);

delay (3000);

digitalWrite(7,HIGH);}

Serial.println(voltage);

while (4<voltage){

sensorValue = analogRead(A0);

voltage = sensorValue \* (5.0 / 1023.0);

Serial.println(voltage);

}

En la siguiente parte vamosa encender el led Rojo. Esto lo hacemos a través de las estructuras condicionales *if*.

if (voltaje < minimo){

digitalWrite(LEDROJO, HIGH);

delay(ledDelay);

digitalWrite(LEDROJO, LOW);

}

**10.** **Código en Java**

**Métodos**

**IniciarSesion:** Utiliza un intent de google para proceder a realizar el inicio de sesión. Si el inicio es correcto se inicia el intent del menú.

**firebaseAuthWithGoogle:** Permite el uso de la base de datos firebase a través de la cuenta de google.

**iniciarBaseDeDatos:**  Inicializa la referencia a la base de datos firebase en la variable DatabaseReference db\_reference.

**consultarSilla:** Recibe dos Strings indicando “Mesa\_id” y “Silla\_id”, indica el valor del campo “Disponible” en la silla indicada del nodo “Grupo” y lo guarda en la variable boolean valor.

**escucharEButton:** Verifica que cuando se agrega un campo en el nodo “Registros” en la base de datos, se lea a que mesa y silla pertenece ese dispositivo para cambiar el estado de dicha mesa y silla en el nodo “Grupo”. Se obtiene el estado a través del método “consultarSilla”. Solo toma en cuenta cada 2 registros ya que el eButton envía dos datos cuando se lo presiona una vez.

**verMesas:** Inicia el intent de EstadoMesa.

**verBateria:** Inicia el intent de EstadoBateria.

**estadoRestaurante:** Inicia el intent de EstadoRestaurante.

**consultarBaterias:** Recorre la base de datos y crea por cada campo “Bateria” un TextView en el cual se indica la mesa, dispositivo y bateria.

**consultarMesas:** Recorre la base de datos y crea por cada campo “Mesa\_id” un TextView en el cual se indica la mesa y el estado, además si este es disponible es de color verde y sino de color rojo.

**consultar:** Recorre el nodo “Registros” y por cada dos registros que sean de la misma silla crea un HashMap donde guarda las veces que se ocupó esa silla. Luego recorre el HashMap para crear un gráfico de líneas con esa información.