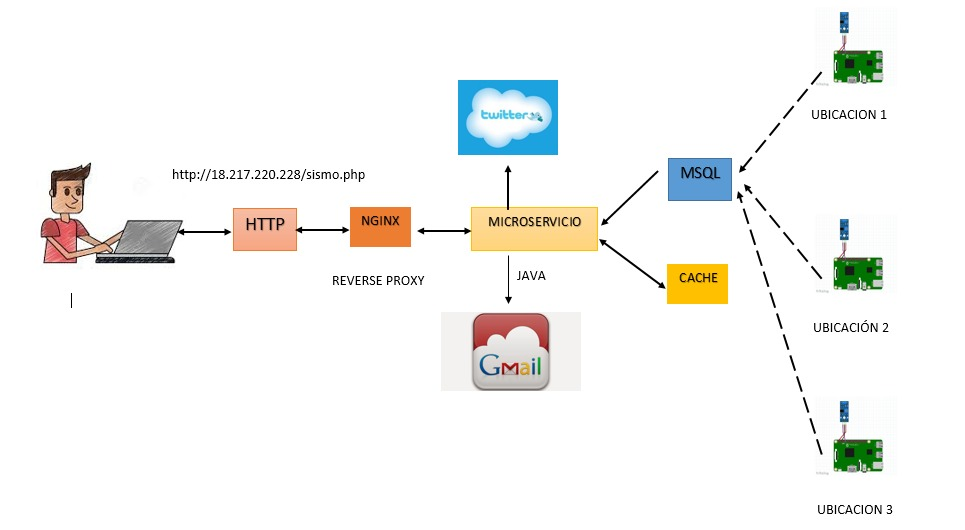
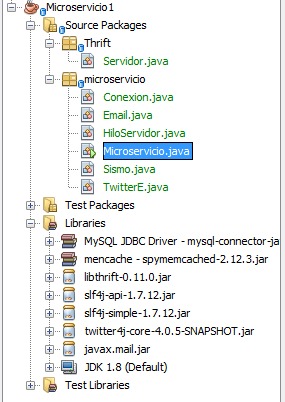
# Manual técnico

Nuestra página web está alojada en un servidor de Amazon (AWS) con dirección pública 18.217.220.228 para realizar el Frontend se utilizó php, y para la parte interna se realizó mediante programación en lenguaje Java, la página web es la que interactúa el usuario para observar los sismos que se han generado en el año, esta página web consta de un reverse proxy para garantizar la seguridad de la página, todo interacción que tenga el usuario con la página web se comunica con el microservicio mediante comunicación binaria, utilizando un GRPS(Apache Thrift), el microservicio se encarga de enviar mensajes de correo, generar un Tweet y actualizar la página web, este microservicio se conecta a la base de datos para observar la información que emiten los 3 sensores y dar alertas de sismos si los 3 sensores han sido activados, la parte de cache la utilizamos para optimizar el tiempo, entre mas rápido sea este proceso, mayor será la eficiencia del servicio

****

Ahora explicaremos la parte del Microservicio



Para poder ejecutar este micro servició se necesitan las siguientes librerías, las cuales las podemos encontrar en nuestro repositorio de github

Ahora vamos explicar en detalle cada objeto y clase creada en java

## Códigos en Java para microservicios

* *Clase “Sismo”, aquí tenemos los atributos que necesitaremos para que el servidor interprete las señales, en este caso usaremos un ID, el nombre del sensor, la fecha y la hora.*

public class Sismo implements Serializable {

private int id;

private String sensor;

private Date fecha;

private Time hora;

public Sismo() {

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getSensor() {

return sensor;

}

public void setSensor(String sensor) {

this.sensor = sensor;

}

public Date getFecha() {

return fecha;

}

public void setFecha(Date fecha) {

this.fecha = fecha;

}

public Time getHora() {

return hora;

}

public void setHora(Time hora) {

this.hora = hora;

}

@Override

public int hashCode() {

int hash = 7;

return hash;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

final Sismo other = (Sismo) obj;

if (this.id != other.id) {

return false;

}

if (!Objects.equals(this.sensor, other.sensor)) {

return false;

}

if (!Objects.equals(this.fecha, other.fecha)) {

return false;

}

if (!Objects.equals(this.hora, other.hora)) {

return false;

}

return true;

}

@Override

public String toString() {

return "," + sensor + "," + fecha + "," + hora ;

}

* *Clase “Email”****,*** *en esta clase enlazamos las alertas a nuestro correo.*

public class Email {

public void enviarConGMail() {

**// Esto es lo que va delante de @gmail.com en tu cuenta de correo. Es el remitente también.**

String remitente = "rwivanco95@gmail.com"; //Para la dirección nomcuenta@gmail.com

Properties props = System.getProperties();

props.put("mail.smtp.host", "smtp.gmail.com"); **//El servidor SMTP de Google**

props.put("mail.smtp.user", remitente);

props.put("mail.smtp.clave", "clave"); **//La clave de la cuenta**

props.put("mail.smtp.auth", "true"); **//Usar autenticación mediante usuario y clave**

props.put("mail.smtp.starttls.enable", "true**"); //Para conectar de manera segura al servidor SMTP**

props.put("mail.smtp.port", "587"); **//El puerto SMTP seguro de Google**

Session session = Session.getDefaultInstance(props);

MimeMessage message = new MimeMessage(session);

try {

message.setFrom(new InternetAddress(remitente));

message.addRecipients(Message.RecipientType.TO, "rwivanco95@gmail.com"); **//Se podrían añadir varios de la misma manera**

message.setSubject("URGENTE!!!"); **//Asunto del correo**

message.setText("Se está sucitando un terremoto en estos instantes, tomar las respectivas precauciones."); **//Mensaje del correo**

Transport transport = session.getTransport("smtp"); **//Se setea el protocolo con el que se abrira sesion**

transport.connect("smtp.gmail.com", remitente, "clave");

transport.sendMessage(message, message.getAllRecipients());

transport.close();

}

catch (MessagingException me) {

me.printStackTrace(); //Si se produce un error

}

}

}

* *Clase “Conexión” aquí accederemos a la base de datos y haremos las comparativas para los mensajes de alerta, si 3 coinciden con la hora y fecha se mandará la respectiva alerta.*

public class Conexion {

private Connection cnx = null;

InetSocketAddress[] servers = new InetSocketAddress[]{ new InetSocketAddress("127.0.0.1", 11211)};

MemcachedClient mc;

public Conexion() {

try {

this.mc = new MemcachedClient(servers);

} catch (IOException ex) {

System.out.println("");;

}

}

public LinkedList<Sismo> obtener() {

try {

Class.forName("com.mysql.jdbc.Driver");

**//Conexion con la base de datos, usuario y clave**

cnx = DriverManager.getConnection("jdbc:mysql://13.58.55.118/prueba", "root", "centos");

**//se establece la conexion**

Statement st= cnx.createStatement();

System.out.println("Se establecio con exito la conexion");

**//Parametros para consulta de la base de datos**

PreparedStatement consulta = cnx.prepareStatement("SELECT id,sensor,fecha,hora FROM sismo" );

**//contiene todo el resultado de la consulta**

ResultSet resultado = consulta.executeQuery();

LinkedList<Sismo> lista =new LinkedList<Sismo>();

Sismo not;

**//recorremos el resultado de la consulta de la base de datos**

while(resultado.next()){

not=new Sismo();

**//guardamos en una Noticia cada parametro**

not.setId(resultado.getInt("id"));

not.setSensor(resultado.getString("sensor"));

not.setFecha(resultado.getDate("fecha"));

not.setHora(resultado.getTime("hora"));

lista.add(not);

}

LinkedList<Sismo> lista1 =new LinkedList<Sismo>();

LinkedList<Sismo> lista2 =new LinkedList<Sismo>();

Map<String,LinkedList<Sismo>> mapa =new HashMap();

if (mc.get("lista")==(null)){

mc.set("lista",36999, lista);

for(Sismo sismo: lista){

if (mapa.isEmpty()){

lista1.add(sismo);

mapa.put(sismo.getHora()+"", lista1);

}else{

if(mapa.containsKey(sismo.getHora()+"")){

lista1=mapa.get(sismo.getHora()+"");

lista1.add(sismo);

mapa.remove(sismo.getHora()+"");

mapa.put(sismo.getHora()+"", lista1);

} else{

lista1 =new LinkedList<Sismo>();

lista1.add(sismo);

mapa.put(sismo.getHora()+"", lista1);

}

}

}

mc.set("mapa", 36999, mapa);

Iterator it = mapa.keySet().iterator();

while(it.hasNext()){

String key = (String) it.next();

if(mapa.get(key).size()==3){

lista2.add(mapa.get(key).get(0));

}

}

mc.shutdown();

return lista2;

}else{

lista1 = (LinkedList<Sismo>)mc.get("lista");

System.out.println(lista.size());

System.out.println(lista1.size());

if(lista.size()==(lista1.size())){

mapa=(Map<String, LinkedList<Sismo>>) mc.get("mapa");

System.out.println(mapa);

Iterator it = mapa.keySet().iterator();

while(it.hasNext()){

String key = (String) it.next();

if(mapa.get(key).size()==3){

lista2.add(mapa.get(key).get(0));

}

}

mc.shutdown();

return lista2;

}else if(lista.size()<(lista1.size())){

mc.set("lista", 36999, lista);

lista1 =new LinkedList<Sismo>();

mapa =new HashMap();

for(Sismo sismo: lista){

if (mapa.isEmpty()){

lista1.add(sismo);

mapa.put(sismo.getHora()+"", lista1);

}else{

if(mapa.containsKey(sismo.getHora()+"")){

lista1=mapa.get(sismo.getHora()+"");

lista1.add(sismo);

mapa.remove(sismo.getHora()+"");

mapa.put(sismo.getHora()+"", lista1);

} else{

lista1 =new LinkedList<Sismo>();

lista1.add(sismo);

mapa.put(sismo.getHora()+"", lista1);

}

}

}

mc.set("mapa", 36999, mapa);

Iterator it = mapa.keySet().iterator();

while(it.hasNext()){

String key = (String) it.next();

if(mapa.get(key).size()==3){

lista2.add(mapa.get(key).get(0));

}

}

mc.shutdown();

return lista2;

}else {

lista1 = (LinkedList<Sismo>)mc.get("lista");

mapa=(Map<String, LinkedList<Sismo>>) mc.get("mapa");

for(Sismo sismo: lista){

if(lista1.contains(sismo)){

System.out.println("ssadas");

}else{

if (mapa.isEmpty()){

lista1 =new LinkedList<Sismo>();

lista1.add(sismo);

mapa.put(sismo.getHora()+"", lista1);

}else{

if(mapa.containsKey(sismo.getHora()+"")){

lista1=mapa.get(sismo.getHora()+"");

lista1.add(sismo);

mapa.remove(sismo.getHora()+"");

mapa.put(sismo.getHora()+"", lista1);

} else{

lista1 =new LinkedList<Sismo>();

lista1.add(sismo);

mapa.put(sismo.getHora()+"", lista1);

}

}

}

}

mc.set("lista", 36999, lista);

mc.set("mapa", 36999, mapa);

Iterator it = mapa.keySet().iterator();

while(it.hasNext()){

String key = (String) it.next();

if(mapa.get(key).size()==3){

lista2.add(mapa.get(key).get(0));

}

}

mc.shutdown();

return lista2;

}

}

} catch (Exception ex) {

System.out.println("Error");

}

return null;

}

}

* *Clase HiloServidor, aquí nos aseguraremos de que corra en tiempo real el proceso con el servidor, conectándonos al mismo*

public class HiloServidor implements Runnable{

@Override

public void run() {

while (true) {

try {

TServerSocket serverTransport = new TServerSocket(7911); **//Conexion al puerto 7911 del servidor**

Servidor.Processor processor = new Servidor.Processor (new Microservicio()); **//Procesos levantados para generar registros usando hilos**

TServer server = new TThreadPoolServer(new TThreadPoolServer.Args(serverTransport).processor(processor));

System.err.println("Servidor en escucha puerto 7915...");

server.serve();

} catch (Exception e) {

}

}

}

}

* *Clase “Twitter”, este Código nos permitirá generar los tweets de alerta.*

public void Tweet(String hora) throws TwitterException{

Twitter twitter;

ConfigurationBuilder cb = new ConfigurationBuilder();

cb.setDebugEnabled(true)

.setOAuthConsumerKey("5i8LHKhjOxCrNdtr7Dbw96JCj")

.setOAuthConsumerSecret("6a6XjenWRtj8BXvklFJIRh1Z")

.setOAuthAccessToken("234858708-qddkuAluOKaAS7pTx8mDBoKpWi6")

.setOAuthAccessTokenSecret("Hh7wltP4VgfkJbCubTuFby7EbHsTyI");

TwitterFactory tf = new TwitterFactory(cb.build());

twitter=(Twitter) tf.getInstance();

Paging pagina = new Paging();

Status tweetEscrito = twitter.updateStatus("ALERTA SISMO!!! MEJORAMIENTO ES TODO\nPERO AHORA SALVA TU VIDA\n"+hora);

}

}

* *Clase principal “Microservicio”, código principal donde ejecutamos nuestras clases creadas que volverán operativo nuestros microservicios, aquí nos enlazamos a la base de datos y enviamos las distintas alertas*

public class Microservicio implements Servidor.Iface {

private static LinkedList<Sismo> noticias;

public static void main(String[] args) {

HiloServidor hs = new HiloServidor();

Thread t = new Thread(hs);

t.start();

}

@Override

public String top10(String ho) {

Date date = new Date(); **//Creación de objeto date de tipo Date**

DateFormat hourFormat = new SimpleDateFormat("HH:mm:ss"); **//Con este objeto se setea el formato de la hora deseado**

Conexion con =new Conexion();

noticias=con.obtener();

String top="";

int acum=0;

for(Sismo noticia: noticias){

if(noticia.getHora().getMinutes()==date.getMinutes()){

try { **//Se aplica un try catch para validar credenciales**

TwitterE twitter =new TwitterE() {}; **//Crea un objeto tipo Twitter**

twitter.Tweet(date+""); //Llama a la funcion Tweet y pasa el parámetro date

Email email =new Email(); **//Crea un objeto tipo Email**

email.enviarConGMail(); **//Llama al metodo envirConGmail**

} catch (TwitterException ex) {

System.out.println("No se puede hacer el tweet");

}

}

acum=acum+1;

top=top+acum+noticia+";"; **//Ingresa los registros de la base de datos**

}

return top; **// Retorno de la lista enlazada**

}

@Override

public String optenerLista() throws TException {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

}

## Programación de Raspberry

* Programa desarrollado en Pythom
* Este código permite a las RaspBerry conectarse a la base de datos almacenada en AWS y enviar a la base cuando detecte movimiento que contiene la fecha, hora y servidor destino

DB\_HOST = '13.58.55.118'

DB\_USER = 'root'

DB\_PASS = 'centos'

DB\_NAME = 'prueba'

datos = [DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME]

conn = MySQLdb.connect(\*datos) # Conectar a la base de datos

cursor = conn.cursor() # Crear un cursor

#cursor.execute(query) # Ejecutar una consulta

channel =17

GPIO.setmode (GPIO.BCM)

GPIO.setup(channel, GPIO.IN)

username = "root"

password = "centos"

def callback(channel):

if GPIO.input(channel):

cursor.execute('INSERT INTO registro1(Fecha,Hora,Servidor) VALUES ("sadsa","asdasd","sad")')

conn.commit()

else:

print "nononon"

GPIO.add\_event\_detect(channel, GPIO.BOTH, bouncetime=1500)

GPIO.add\_event\_callback(channel, callback)

## Configuración del servidor

* A continuación los pasos para la configuración del servidor
* Para poder ingresar a los recursos de nuestra máquina virtual en la nube requerimos de dos herramientas.
* Putty
* PuttyGen

PuttyGen: Encargada de convertir llaves de tipo .pem a .ppk

Putty: herramienta la cual nos permite ingresar remotamente a nuestra máquina virtual, requerimos de la llave que nos genera PuttyGen y de la ip publica de la maquina creada en la nube.

Para la implementación de nuestra red de microservicio es necesario instalar los siguientes paquetes en los servidores a usar.

**Instalacion de servidor web (Nginx)**

#yum install nginx

**Instalacion de apache**

#yum install httpd

**Instalacion de mysql**

1. Descargue y agregue el repositorio, luego actualice

#wget <http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm>

#sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm

#yum update

1. Instalar mysql

#sudo yum install mysql-server

**Instalacion de phpmyadmin**

#yum -y install phpmyadmin

**Instalacion de Php5-Fpm**

#yum install php5-fpm

**Instalación de Memcached**

#yum –y install memcached

**Instalación de Apache Thrift**

1. Descargamos el archivo de la página oficial

#wget [http://thrift.apache.org/download/[thrift-0.11.0.tar.gz](http://www.apache.org/dyn/closer.cgi?path=/thrift/0.11.0/thrift-0.11.0.tar.gz)](http://thrift.apache.org/download/nombredelarchivo)

1. Descomprimimos el archive

tar -xvf thrift-0.11.0.tar.gz

1. Nos movemos dentro del documento descomprimido y ejecutamos

cd thrift-0.11.0

./bootstrap.sh

./configure

**Habilitar los Servicios**

[root@localhost lmoreno]# systemctl start nginx

[root@localhost lmoreno]# systemctl start httpd

[root@localhost lmoreno]# systemctl start php-fpm.service

[root@localhost lmoreno]# systemctl start mysqld

[root@localhost lmoreno]#systemctl start memcached

**CONFIGURACION DE NGINX (ReverseProxy)**

Nos dirigimos al directorio en donde se encuentra la configuración de nuestro servidor

# cd /etc/nginx/conf.d

Editando: (#gedit default.conf)

server {

listen 80;

root /var/www/html/;

index index.php index.html index.htm;

server\_name \_ www.distribuidos\_noticias.com;

location / {

try\_files $uri $uri/ /index.php;

}

location ~ \.php$ {

fastcgi\_index index.php;

fastcgi\_pass 127.0.0.1:9000;

include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

fastcgi\_param SCRIPT\_NAME $fastcgi\_script\_name;

}

}

Procedemos a guardar la nueva configuración.

**Ahora:**

Regresamos un directorio atrás (# cd /etc/nginx) y modificaremos el archivo nginx.conf(#gedit nginx.conf)

user apache;

**Observacion:** Modificar el puerto de php a puerto 8080, debido a que nginx trabaja en puerto 80.

**HABILITAMOS EL SERVICIO**

Cuando ingresamos vía putty nos dirigimos al directorio subir donde están alojados mis servicios

#cd subir

Ahora:

Ejecutamos los microservicios en segundo plano

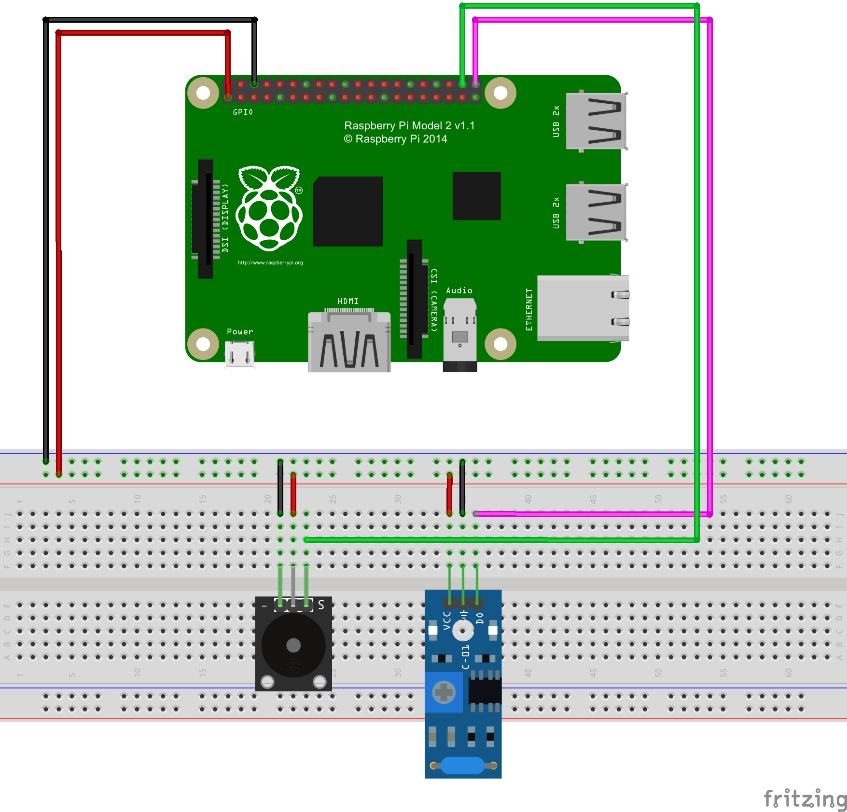
#Sudo java -jar Microservicio1.jar &

#Sudo java -jar Microservicio.jar &

Finalmente probamos la conexión en la página web.



# Esquemático sensor



PIN 2 Raspberry - Alimentación del sensor 5V – Cable Rojo – PIN 1 Sensor de Movimiento

PIN 6 Raspberry – GND del Sensor – Cable Negro – PIN 2 Sensor de Movimiento

PIN 40 Raspberry – Receptor de datos que envía el sensor – Cable Fucsia – PIN 3 Sensor de Movimiento

PIN 2

PIN 1

PIN 3

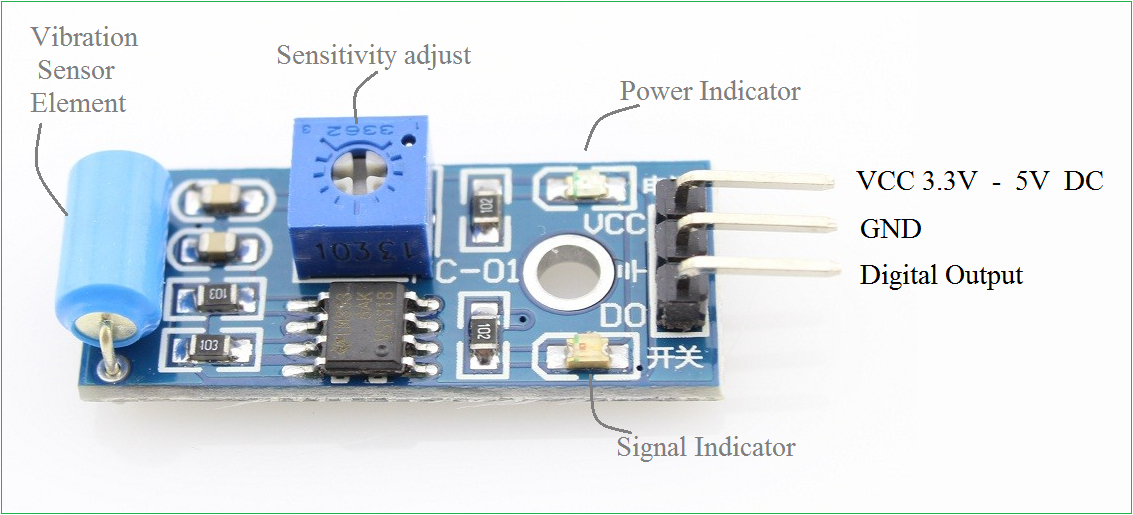
PIN 2

PIN 6

PIN 38

PIN 40

# PINES SENSOR



**ANEXO**

