**Control de calidad en una camaronera**

Programación de Sistemas Telemáticos

Winter Emanuel Álava Intriago; Cesar Antonio Coloma Guzmán

Dario Javier Erreyes Ordoñez; Jancarlo André Valencia Noboa

Facultad de Ingeniería en Electricidad y Computación

Escuela Superior Politécnica del Litoral

**Guía**

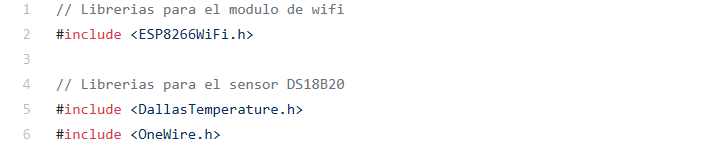
**Código de Arduino**

**Lectura y subida de datos**

Para la lectura y subida de datos de los sensores a la base de datos al servidor remoto se utilizó un módulo Wifi de Arduino programable (ESP8266).

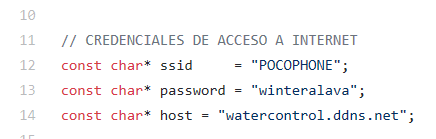
**Librerías utilizadas:**

Para la lectura de datos de temperatura se utilizó una librería de Adafruit para el sensor DS18B20 además de la librería OneWire, para el módulo de Wifi se utilizó la librería ESP8266Wifi. Para la lectura de datos de pH se calibró dicho sensor para poder determinar analógicamente qué medida de pH corresponde a la salida de voltaje del mismo.

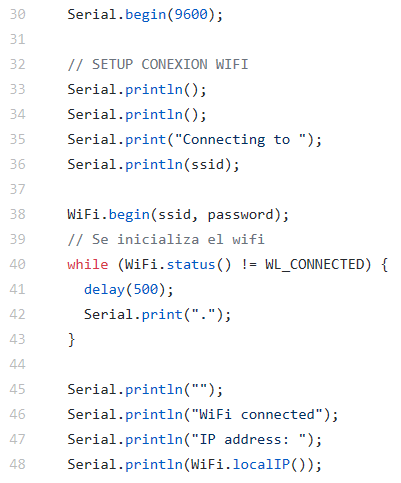


**Conexión a Internet:**

Para establecer una conexión a internet se debía definir las credenciales que daría el acceso a internet. En la presentación de nuestro proyecto se decidió utilizar los datos móviles de un celular. Además, se intenta acceder a nuestro servidor web cuyo dominio es el declarado en la variable host.



En el setup se abre el monitor serial para poder imprimir en pantalla los pasos de conexión a internet del módulo wifi. Para hacerlo se instancia el objeto Wifi y se espera su conexión a internet. Se espera que se conecte y al hacerlo se imprimirá un mensaje de conexión exitosa.



Posteriormente, en el loop se envían los datos al servidor web por medio de un objeto WifiClient en donde se establece una conexión a un archivo php del servidor el cual utilizando métodos GET se logran ejecutar queries que insertan datos a la base de datos requerida. Se verifica que la conexión no haya tardado demasiado, considerando 5 segundos como tiempo suficiente para determinar que la conexión no fue exitosa.

**Creación del Servidor Web**

El primer paso en la configuración de la RPI3 como un servidor web es de montar la imagen, es decir, el sistema operativo, una vez realizado este paso, es aconsejable asignar una contraseña al usuario root:

|  |  |
| --- | --- |
| 1 | sudo su |
| 2 | passwd |
| ingrese contraseña: \*\*\*\* | |
| repita contraseña:\*\*\*\* | |
| 3 | exit |

Posteriormente se requiere designar una ip estática en la red a la que esta accediendo a internet la RPI3:

|  |  |
| --- | --- |
| 1 | ifconfig |
| obtenga su ip y mascara de subred | |
| ip: 192.168.100.182  mask: 255.255.255.0 | |
| 4 | route -ne |
| obtenga el puerta de enlace de la red | |
| gateway: 192.168.100.1 | |
| 5 | sudo nano /etc/dhcpcd.conf |
| edite el archivo de la siguiente manera | |
| interface wlan0  static ip\_address=192.168.100.182  static routers=192.168.100.1  static domain\_name\_server=8.8.8.8 8.8.4.4 | |
| 6 | sudo reboot |

A continuacions e van a instalar los paquetes que porveen la creación y la administración del servidor LAMP, es decir: APACHE, PHP, PHPMYADMIN MYSQL, y sus respectivas configuraciones:

|  |  |
| --- | --- |
| 1 | sudo reboot |
| 2 | sudo apt-get update |
| 3 | sudo apt-get upgrade |
| 4 | sudo apt-get install apache2 |
| 5 | sudo apt-get install php libapache2-mod-php |
| 6 | sudo apt-get install php-mcrypt php-mysql |
| 7 | sudo nano /etc/apache2/mods-enabled/dir.conf |
| edite el archivo de la siguiente forma | |
| <ifmodule mod\_dir.c>  directoryindex index.php index.html index.cgi index.pl index xhtml  </ifmodule>  # vim: syntax=apache2 ts=4 sw=4 sts=4 sr noet | |
| 8 | sudo apt-get install mariadb-server-10.0  -seleccionar apache2  -seleccionar si  -designar contrasena al admin de mysql |
| 9 | sudo apt-get install phpmyadmin php-mbstring php-gettext |
| 10 | sudo apt install php7.1-mcrypt |
| 11 | sudo ln -s /etc/php/7.1/mods-available/mcrypt.ini /etc/php/7.3/mods-available/ |
| 12 | sudo phpenmod mcrypt |
| 13 | sudo apache2ctl configtest |
| 14 | sudo systemctl restart apache2 |
| 15 | reboot |

A continuación, se instalan los paquetes que permiten la apertura del servidor local hacia las demás redes de internet, siguiendo los siguientes pasos:

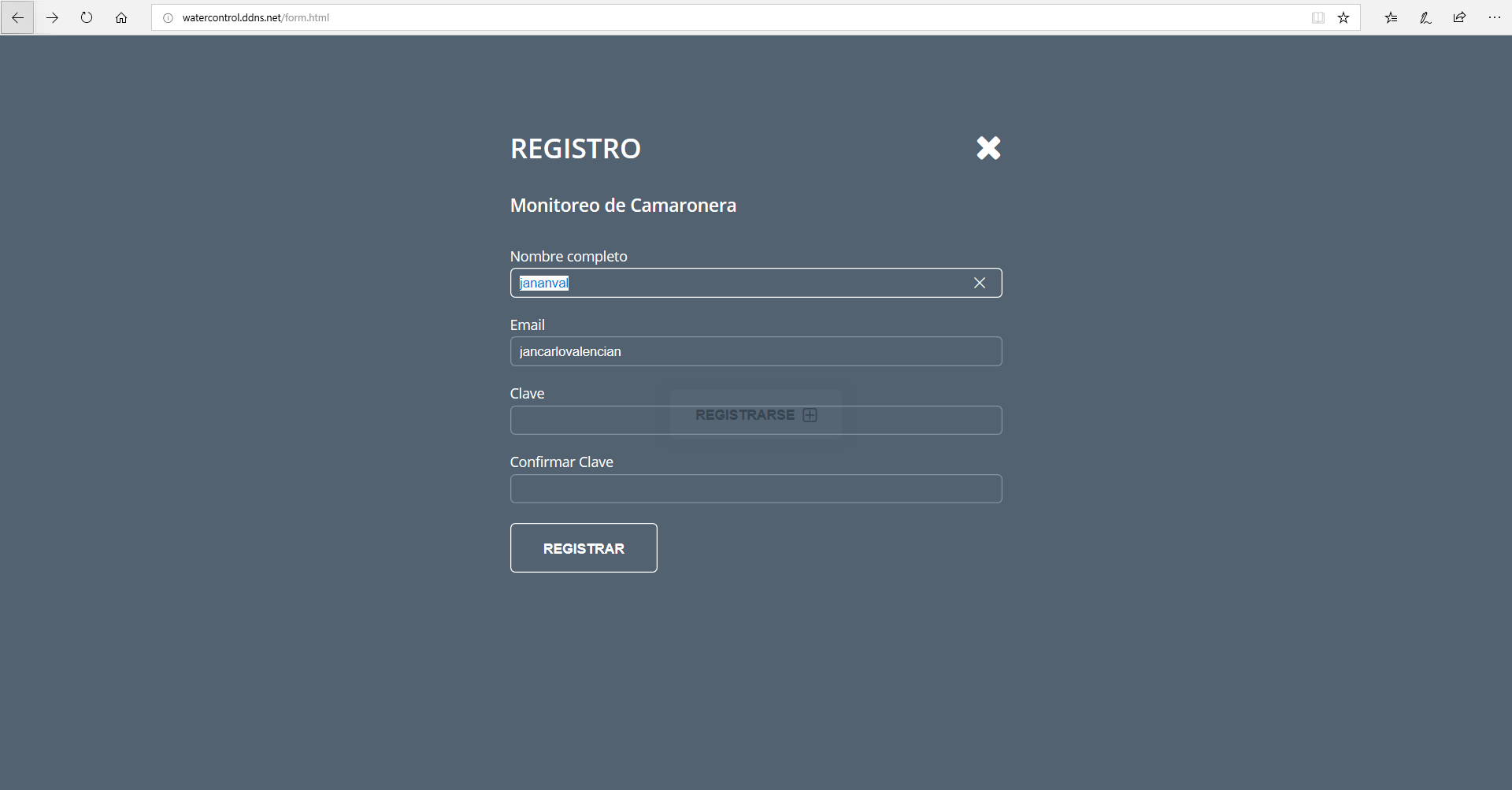
(Nota: es necesario abrir el puerto 80 en el router de su red para que el servidor pueda recibir y enviar peticiones, si no puede manipular el router por su cuenta, es necesario que llame a su proveedor de internet y pedirle explícitamente que abra dicho puerto para la ip de la RPI3)

|  |  |
| --- | --- |
| 1 | sudo apt-get install ddclient libjson-any-perl  -seleccionar [www.dyndns.com](http://www.dyndns.com) |
| 2 | sudo wget <https://files.pimylifeup.com/portforwarding/ddclient-3.9.0.tar.gz> |
| 3 | sudo tar -zxvf ddclient-3.9.0.tar.gz |
| 4 | sudo cp -f ddclient-3.9.0/ddclient /usr/sbin/ddclient |
| 5 | sudo mkdir /etc/ddclient |
| 6 | sudo mv /etc/ddclient.conf /etc/ddclient |
| crearse una cuenta en noip.com  iniciar sesión  cree un hostname (dominio web) | |
| 7 | sudo nano /etc/ddclient/ddclient.conf |
| use=web, web=checkip.dyndns.com/, web-skip= ‘ip address’  ssl=yes  protocol=dyndns2  server-dynupdate.no-ip.com  login=usuariodesucuenta  password=contrasenadesucuenta  dominioescogido.ddns.net | |
| 8 | sudo /etc/init.d/ddclient restart |
| 9 | sudo nano /etc/default/ddclient |
| editar el archivo de tal forma que los parámetros queden de la siguiente forma | |
| run\_dhclient=”false”  run\_ipup=”false”  run\_daemon=”true”  run\_interval=”300” | |
| 10 | sudo nano /etc/cron.weekly/ddclient |
| edite el archivo de la siguiente forma: | |
| # /bin/sh  /usr/sbin/ddclient -force | |
| 11 | sudo nano chmod +x /etc/cron.weekly/ddclient |
| 12 | sudo reboot |

Una vez concluidos estos pasos, usted posee un servidor web, que puede ser accedido desde cualquier otra red con acceso a internet, bajo un dominio web asignado por noip.com, bastará con encender la RPI3 para que el servidor se ponga en marcha

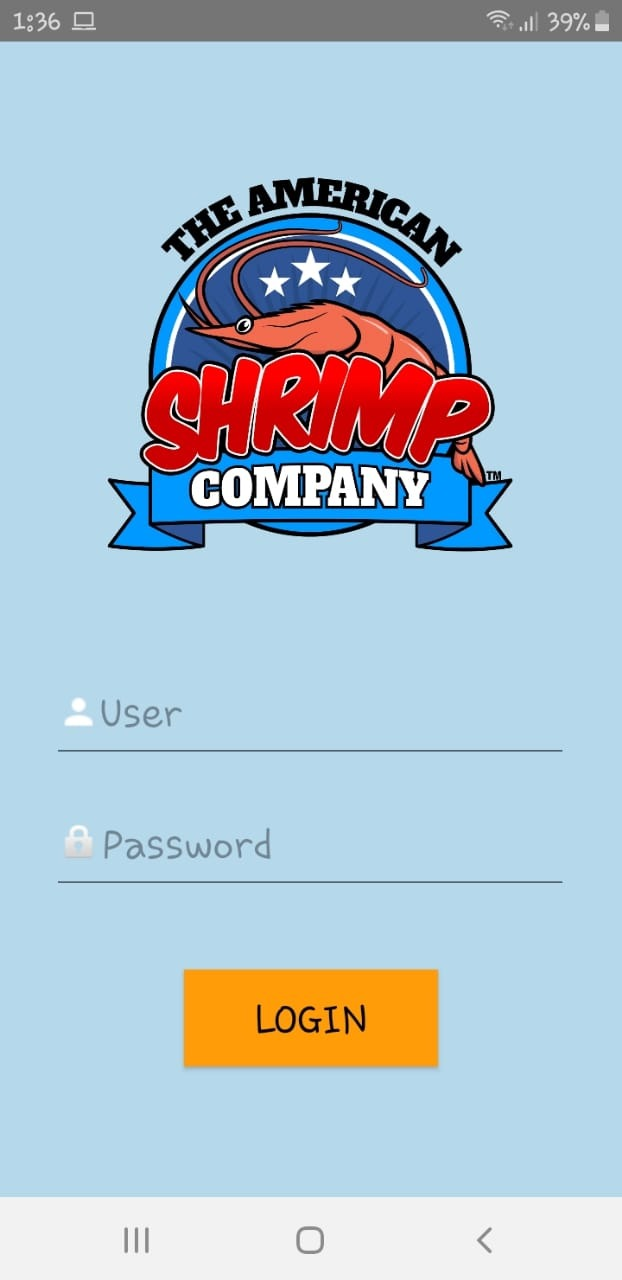
**Página Web**

Se realiza una pagina web con un formulario para un ingreso de usuario, se puede descargar y modificar un tipo de formulario de internet. Además, se agrega el código php para ingresar un usuario en la base de datos de phpmyadmin.



**Aplicación**

En el MainActivity.xml se crea el siguiente diseño para inicio de sesión:



Luego en su clase de java se coloca el código para que se realice el inicio de sesión, manteniendo su sesión abierta hasta que el usuario la cierre.

**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **if** (ObtenercontraseñaGuardada().equals(**c**) && ObtenerUsuarioGuardado().equals(**u**)){  
 Intent i = **new** Intent(**this**, InicioActivity.**class** );  
 startActivity(i);  
 Notificaciones.*guardarEstado*(MainActivity.**this**,**true**);  
 startService(**new** Intent (**this**,Notificaciones.**class**));  
 finish();  
 }  
  
 **user**= (EditText) findViewById(R.id.***user***);  
 **password**= (EditText) findViewById(R.id.***password***);  
}  
  
*/\*\*  
 \* Este metodo valida el usuario y contraseña ingresado, de ser correctos los dirige a la siguiente ventana, caso contrario indica que los datos ingresados son incorrectos.  
 \*/***public void** Ingresar (View v) {  
 String usuario=**user**.getText().toString().trim();  
 String contrasena=**password**.getText().toString().trim();  
  
 **if**(usuario.equals(**u**) && contrasena.equals(**c**)){  
 Intent i = **new** Intent(**this**, InicioActivity.**class** );  
 startActivity(i);  
 guardarInicio();  
 Notificaciones.*guardarEstado*(MainActivity.**this**,**true**);  
 startService(**new** Intent (**this**,Notificaciones.**class**));  
 finish();  
 }  
 **else** {  
 Toast.*makeText*(**this**, **"El usuario o contraseña ingresados son incorrectos!"**,Toast.***LENGTH\_SHORT***).show();  
 **user**.setText(**""**);  
 **password**.setText(**""**);  
 }  
  
}  
  
*/\*\*  
 \* Este metodo permite guardar la informacion del usuario con el fin de que este no tenga q iniciar sesion si llegara a cerrar la app  
 \*/***public void** guardarInicio(){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 preferences.edit().putString(***USUARIO***,**user**.getText().toString().trim()).apply();  
 preferences.edit().putString(***CONTRASEÑA***,**password**.getText().toString().trim()).apply();  
}  
  
*/\*\*  
 \* Este metodo elimina los datos del usuario cuando este cierra sesion en el aplicativo.  
  
 \*/***public static void** CerrarSesion(Context c,String users, String passwords){  
 SharedPreferences preferences=c.getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 preferences.edit().putString(***USUARIO***,users).apply();  
 preferences.edit().putString(***CONTRASEÑA***,passwords).apply();  
}  
  
*/\*\*  
 \* Obtiene al nombre del usurio que inicia sesion desde la memoria del aplicativo  
 \*/***public** String ObtenerUsuarioGuardado(){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 **return** preferences.getString(***USUARIO***,**""**);  
}  
  
*/\*\*  
 \* Retorna la contraseña del usurio que inicia sesion guardada en la memoria del aplicativo  
 \*/***public** String ObtenercontraseñaGuardada(){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 **return** preferences.getString(***CONTRASEÑA***,**""**);  
}

Luego se crea una nueva Activity con el nombre de InicioActivity, y en su archivo xml se diseña el siguiente menú:



Agregamos los métodos para el método on click que tendrá cada botón del menú.

**public void** Monitorear (View v){  
 Intent i = **new** Intent(**this**, MonitoreoActivity.**class** );  
 startActivity(i);  
}  
  
*/\*\*  
 \* metodo onClick del boton agregar donde nos lleva a la vantana agregar piscina  
 \*/***public void** Agregar(View v) {  
 Intent i = **new** Intent(**this**, AgregarActivity.**class** );  
 startActivity(i);  
}  
  
*/\*\*  
 \* metodo onClick del boton salir que cierra sesion del usuario llevandonos a la ventana de ingreso  
 \*/***public void** Salir(View v){  
 MainActivity.*CerrarSesion*(InicioActivity.**this**, **""**,**""**);  
 Intent i = **new** Intent(**this**, MainActivity.**class** );  
 stopService(**new** Intent (**this**,Notificaciones.**class**));  
 startActivity(i);  
 finish();  
}

Luego creamos una nueva activity llamada MoritoreoActivity, en donde agregaremos los botones referentes a cada piscina de manera dinámica.

@Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_monitoreo***);  
 **contenedor**= (LinearLayout) findViewById(R.id.***contenedor***);  
 ArrayList<Boton> botones= **new** ArrayList<Boton>();  
 **for** (**int** i=1; i<=(cantPiscinas()); i++){  
 botones.add(**new** Boton(**"Piscina "**+String.*valueOf*(i),i));  
 }  
 **for** (**final** Boton bt:botones){  
 Button btn = **new** Button(**this**);  
 btn.setText(bt.**texto**);  
 btn.setId(bt.**ID**);  
 btn.setTextSize(18);  
 btn.setBackgroundColor(Color.*parseColor*(**"#FFE794"**));  
 btn.setTextColor(Color.***BLACK***);  
 **contenedor**.addView(btn);  
 **contenedor**.addView(**new** TextView(**this**));  
 btn.setOnClickListener(**new** View.OnClickListener() {  
 **public void** onClick(View v) {  
 Intent i = **new** Intent(MonitoreoActivity.**this**,EstadoActualActivity.**class** );  
 i.putExtra(**"ID"**,String.*valueOf*(bt.**ID**));  
 startActivity(i);  
 }  
 });  
 }  
 }  
  
  
 */\*\*  
 \* Retorna la cantidad de piscinas ingresadas en la base de datos tomando en cuenta al usuario  
 \*/* **public int** cantPiscinas(){  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"SELECT \* FROM Piscina;"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 Toast.*makeText*(MonitoreoActivity.**this**,**"Conexión Establecida"**, Toast.***LENGTH\_LONG***).show();  
  
 String resultadoConsulta = resultadoSQL[0];  
 String numFilas = resultadoSQL[1];  
 String numColumnas = resultadoSQL[2];  
 **return** Integer.*parseInt*(numFilas);  
 }**catch**(Exception ex)  
 {  
 Toast.*makeText*(**this**, **"Error al obtener resultados de la consulta Transact-SQL: "** + ex.getMessage(), Toast.***LENGTH\_LONG***).show();  
 **return** 0;  
 }  
 }  
  
 **class** Boton{  
 String **texto**;  
 **int ID**;  
  
 **public** Boton(String texto, **int** ID) {  
 **this**.**texto** = texto;  
 **this**.**ID** = ID;  
 }  
 }  
}

Luego creamos las siguientes Activities con los codigos que se muestran a continuación:

EstadoActualActivity.java

**public class** EstadoActualActivity **extends** AppCompatActivity {  
 **private** String **piscina**,**ID**;  
 **private** TextView **name**;  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
 **private** TextView **consulta**;  
  
 */\*\*  
 \* En este método sobreescrito se inicializan las variables a usar y que se encuentran definidas en el archivo XML  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_estado\_actual***);  
 Bundle bundle = getIntent().getExtras();  
 **ID**= bundle.getString(**"ID"**);  
 **piscina**=**"Piscina #"** + **ID**;  
 **name**=(TextView) findViewById(R.id.***title***);  
 **consulta**=(TextView) findViewById(R.id.***historial***);  
 **name**.setText(**piscina**);  
 MostrarDatos();  
 }  
  
 */\*\*  
 \* Este genera un cuadro de dialogo de seleccion que permite que el usuario elija la opcion de historial que desea ver  
 \*/* **public void** Historial(View v){  
 **final** String[] items = {**"General"**, **"Alertas"**};  
 AlertDialog.Builder builder =**new** AlertDialog.Builder(**this**);  
 builder.setTitle(**"HISTORIAL"**).setItems(items, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** item) {  
 **if** (items[item].equals(**"General"**)){  
 Intent i = **new** Intent(EstadoActualActivity.**this**, HistorialActivity.**class** );  
 i.putExtra(**"IDpiscina"**,String.*valueOf*(**ID**));  
 startActivity(i);  
 }  
 **else**{  
 Intent i = **new** Intent(EstadoActualActivity.**this**, AlertasActivity.**class** );  
 i.putExtra(**"IDpiscina"**,String.*valueOf*(**ID**));  
 startActivity(i);  
 }  
 }  
 });  
  
 builder.create().show();  
 }  
  
 */\*\*  
 \* Este metodo muestra los datos de la piscina que desea monitorear  
 \*/* **public void** MostrarDatos(){  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"SELECT \* FROM Piscina WHERE ID="** +**ID**+**";"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 Toast.*makeText*(EstadoActualActivity.**this**,**"Conexión Establecida"**, Toast.***LENGTH\_LONG***).show();  
  
 String[] resultadoConsulta = resultadoSQL[0].split(**","**);  
 **consulta**.setText(**"Datos:\n"**+**"Largo= "**+resultadoConsulta[1] + **"\nAncho= "** +resultadoConsulta[2] + **"\nProfundidad= "** +  
 resultadoConsulta[3] + **"\n\nDatos de Control:\npH maximo= "** +resultadoConsulta[4] + **"\npH minimo= "** +resultadoConsulta[5]  
 + **"\nTemperatura Maxima= "** +resultadoConsulta[6] + **" ºC\nTemperatura minima= "** +resultadoConsulta[7]+**" ºC"**);  
 }**catch**(Exception ex)  
 {  
 Toast.*makeText*(**this**, **"Error al obtener resultados de la consulta Transact-SQL: "** + ex.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
  
 }  
}

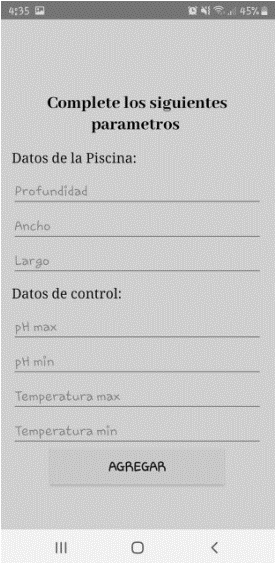
HistorialActivity.java

**public class** HistorialActivity **extends** AppCompatActivity {  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
 **private** String **ID**;  
 **private** TableLayout **table**;  
  
 */\*\*  
 \* En este método sobreescrito se inicializan las variables a usar y que se encuentran definidas en el archivo XML  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_historial***);  
 Bundle bundle = getIntent().getExtras();  
 **ID**= bundle.getString(**"IDpiscina"**);  
 **table** = (TableLayout) findViewById(R.id.***tablelayout***);  
 mostrarResultados();  
  
 }  
  
 */\*\*  
 \* Este metodo permite mostrar los datos de las lecturas tomadas por los sensores ubicadas en la base de datos  
 \*/* **public void** mostrarResultados()  
 {  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"SELECT \* FROM Lectura WHERE idPiscina="**+**ID**+**";"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 Toast.*makeText*(HistorialActivity.**this**,**"Conexión Establecida"**, Toast.***LENGTH\_LONG***).show();  
 String[] cabecera={**"\n ,Fecha y Hora,Valor de pH,Temperatura, "**,**"1"**,**"5"**};  
 Tablas(cabecera);  
 Tablas(resultadoSQL);  
 *// String resultadoConsulta = resultadoSQL[0];  
 //filasconsulta.setText("ID, pH, Temperatura, #Piscina\n"+resultadoConsulta );* }**catch**(Exception ex)  
 {  
 Toast.*makeText*(**this**, **"Error al obtener resultados de la consulta Transact-SQL: "** + ex.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
 }  
  
 */\*\*  
 \* Este metodo nos permite colocar los datos obtenidos de la base de datos y colocarlos dentro de un Table Layout, con el fin de mejorar la presentacion de dicha infromación  
 \*/* **public void** Tablas(String[] resultado) {  
  
 String resultadoConsulta = resultado[0];  
 **int** NUM\_ROW = Integer.*parseInt*(resultado[1]);  
 **int** NUM\_COLS = Integer.*parseInt*(resultado[2]);  
 String[] filas = resultadoConsulta.split(**"\n"**);  
 **for** (**int** i = 1; i < (NUM\_ROW+1); i++) {  
 TableRow tableRow = **new** TableRow(HistorialActivity.**this**);  
 **table**.addView(tableRow);  
 tableRow.setBackgroundColor(Color.***GRAY***);  
 String[] columnas = filas[i].split(**","**);  
  
 **for** (**int** j = 1; j < (NUM\_COLS-1); j++) {  
 TextView valor = **new** TextView(getApplicationContext());  
 valor.setText(columnas[j] + **" "**);  
 valor.setTextColor(Color.***BLACK***);  
 tableRow.addView(valor);  
 }  
 }  
 }  
}

AlertasActivity.java

**public class** AlertasActivity **extends** AppCompatActivity {  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
 **private** String **ID**;  
 **private** TableLayout **table**;  
  
 */\*\*  
 \* En este método sobreescrito se inicializan las variables a usar y que se encuentran definidas en el archivo XML, o gruadados en extras  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_alertas***);  
 Bundle bundle = getIntent().getExtras();  
 **ID**= bundle.getString(**"IDpiscina"**);  
 **table** = (TableLayout) findViewById(R.id.***tablealertas***);  
 mostrarResultados();  
 }  
  
 */\*\*  
 \* Este metodo permite mostrar los resultados de las alertas registradas en la bases de datos.  
 \*/* **public void** mostrarResultados()  
 {  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"SELECT \* FROM Alerta WHERE idpiscina="**+**ID**+**";"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 Toast.*makeText*(AlertasActivity.**this**,**"Conexión Establecida"**, Toast.***LENGTH\_LONG***).show();  
 String[] cabecera={**"\n ,Fecha y Hora,Descripcion, "**,**"1"**,**"4"**};  
 Tablas(cabecera);  
 Tablas(resultadoSQL);  
 }**catch**(Exception ex)  
 {  
 Toast.*makeText*(**this**, **"Error al obtener resultados de la consulta Transact-SQL: "** + ex.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
 }  
  
 */\*\*  
 \* Este metodo nos permite colocar los datos obtenidos de la base de datos y colocarlos dentro de un Table Layout, con el fin de mejorar la presentacion de dicha infromación  
 \** ***@param resultado*** *\*/* **public void** Tablas(String[] resultado) {  
  
 String resultadoConsulta = resultado[0];  
 **int** NUM\_ROW = Integer.*parseInt*(resultado[1]);  
 **int** NUM\_COLS = Integer.*parseInt*(resultado[2]);  
 String[] filas = resultadoConsulta.split(**"\n"**);  
 **for** (**int** i = 1; i < (NUM\_ROW+1); i++) {  
 TableRow tableRow = **new** TableRow(**this**);  
 **table**.addView(tableRow);  
 tableRow.setBackgroundColor(Color.***LTGRAY***);  
 String[] columnas = filas[i].split(**","**);  
  
 **for** (**int** j = 1; j < (NUM\_COLS-1); j++) {  
 TextView valor = **new** TextView(getApplicationContext());  
 valor.setText(columnas[j] + **" "**);  
 valor.setTextColor(Color.***BLACK***);  
 tableRow.addView(valor);  
 }  
 }  
 }  
}

Para agregar una pisicina se crea una nueva activity con el nombre de AgregarActivity y se realiza el siguiente diseño en su archivo xml:



Y En su archivo de Java se copia el siguiente código:

AgregarActivity.java

**public class** AgregarActivity **extends** AppCompatActivity {  
 **private** EditText **alto**, **ancho**, **largo**, **pHmax**, **pHmin**, **TempMax**, **TempMin**;  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
  
 */\*\*  
 \* En este método sobreescrito se inicializan las variables a usar y que se encuentran definidas en el archivo XML  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_agregar***);  
 **alto**=(EditText) findViewById(R.id.***alto***);  
 **ancho**=(EditText) findViewById(R.id.***ancho***);  
 **largo**=(EditText) findViewById(R.id.***largo***);  
 **pHmax**=(EditText) findViewById(R.id.***phmax***);  
 **pHmin**=(EditText) findViewById(R.id.***phmin***);  
 **TempMax**=(EditText) findViewById(R.id.***tempmax***);  
 **TempMin**=(EditText) findViewById(R.id.***tempmin***);  
 }  
  
 */\*\*  
 \* Este metodo permite agregar en la base de datos una nueva piscina de la camaronera, enviando los datos ingresados en la activity  
 \* a la base de datos del proyecto, validando todos los datos ingresados.  
 \*  
 \*/* **public void** agregar(View v){  
 **if** (validarDatos()){  
 AlertDialog.Builder builder =  
 **new** AlertDialog.Builder(**this**);  
  
 builder.setMessage(**"¿Desea Registrar la piscina?"**).setTitle(**"Mensaje de confirmación"**)  
 .setPositiveButton(**"YES"**, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** id) {  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"INSERT INTO Piscina (ID, largo, ancho, profundidad, phMax, phmin, tmax, tmin, idusuario) VALUES (NULL,"**+  
 **largo**.getText().toString()+**","**+**ancho**.getText().toString() +**","**+**alto**.getText().toString()+**","**+**pHmax**.getText().toString()  
 +**","**+**pHmin**.getText().toString()+**","**+**TempMax**.getText().toString()+**","**+**TempMin**.getText().toString()+**", 1);"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 Toast.*makeText*(AgregarActivity.**this**, **"Se ha registrado la piscina con exito!"**,Toast.***LENGTH\_SHORT***).show();  
 **alto**.setText(**""**); **ancho**.setText(**""**); **largo**.setText(**""**); **pHmax**.setText(**""**); **pHmin**.setText(**""**); **TempMax**.setText(**""**); **TempMin**.setText(**""**);  
  
 }**catch**(Exception ex)  
 {  
 Toast.*makeText*(AgregarActivity.**this**, **"Error al agregar la piscina."**,Toast.***LENGTH\_SHORT***).show();  
 }  
  
 }  
 }).setNegativeButton(**"NO"**,**new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** id) {  
 dialog.cancel();  
 }  
 });  
 builder.create().show();  
 }  
  
 **else**{  
 AlertDialog.Builder builder =  
 **new** AlertDialog.Builder(**this**);  
  
 builder.setMessage(**"La información ingresada no es valida."**)  
 .setTitle(**"Error"**)  
 .setPositiveButton(**"OK"**, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** id) {  
 dialog.cancel();  
 }  
 });  
 builder.create().show();  
 }  
 }  
  
 */\*\*  
 \* Esta funcion se encarga de validar que todos los datos sean de tipo float, ademas que el valor maximo de algun parametro sea siempre mayor que el minimo,  
 \* usando la funcion de try--catch donde si logra transformarlo a float todos los valores inresados, entonces retorna true, y de caso contrario retorna false.  
 \*  
 \** ***@return*** *\*/* **public** Boolean validarDatos(){  
 **try**{  
 Float.*parseFloat*(**alto**.getText().toString().trim());  
 Float.*parseFloat*(**ancho**.getText().toString().trim());  
 Float.*parseFloat*(**largo**.getText().toString().trim());  
 Float phM=Float.*parseFloat*(**pHmax**.getText().toString().trim());  
 Float phm=Float.*parseFloat*(**pHmin**.getText().toString().trim());  
 Float TM=Float.*parseFloat*(**TempMax**.getText().toString().trim());  
 Float Tm=Float.*parseFloat*(**TempMin**.getText().toString().trim());  
 **if** ((phm<phM)&&(phm>0)&&(phM<14)&&(TM>Tm)){  
 **return true**;  
 }  
 **return false**;  
 }**catch**(Exception e){  
 **return false**;  
 }  
 }  
}

Luego creamos una clase Java con el nombre de AsyncQuery donde se agrega el código para la conexión con la base de datos.

AsyncQuery.java

**public class** AsyncQuery **extends** AsyncTask<String[],Void,String[]> {  
  
 **private** Connection **conexionMySQL**;  
 **private** Statement **st** = **null**;  
 **private** ResultSet **rs** = **null**;  
  
 */\*\*  
 \* Este metodo nos permite ingresar a la base de datos obtener y enviar datos hacia ella.  
  
 \*/* **protected** String[] doInBackground(String[]... datos) {  
 String sql = datos[0][5];  
 String resultadoSQL = **""**;  
 String[] totalResultadoSQL = **null**;  
 **int** numColumnas = 0;  
 **int** numFilas = 0;  
 String SERVIDOR = datos[0][0];  
 String PUERTO = datos[0][1];  
 String BD = datos[0][2];  
 String USUARIO = datos[0][3];  
 String PASSWORD = datos[0][4];  
  
 **try**{  
 **conexionMySQL** = DriverManager.*getConnection*(**"jdbc:mysql://"** + SERVIDOR + **":"** + PUERTO + **"/"** + BD,  
 USUARIO,PASSWORD);  
  
 **st** = **conexionMySQL**.createStatement();  
 **if**(sql.contains(**"INSERT"**)){  
 Log.*d*(**"Query: "**,sql);  
 **st**.executeUpdate(sql);  
 }  
 **else** {  
 **rs** = **st**.executeQuery(sql);  
 **rs**.last();  
 numFilas = **rs**.getRow();  
 **if**(numFilas == 0)  
 {  
 resultadoSQL = **"No se ha producido ningún resultado. Revise la consulta realizada.\n"**;  
 }**else** {  
 **for**(**int** i=1;i<=numColumnas;i++){  
 **if**(i>1) resultadoSQL += **","**;  
 resultadoSQL += **rs**.getMetaData().getColumnName(i);  
 }  
 resultadoSQL += **"\n"**;  
 **rs**.beforeFirst();  
 **while** (**rs**.next())  
 {  
 numColumnas = **rs**.getMetaData().getColumnCount();  
 **for**(**int** i=1;i<=numColumnas;i++){  
 **if**(i>1) resultadoSQL += **","**;  
 resultadoSQL += **rs**.getString(i);  
 }  
 resultadoSQL += **"\n"**;  
 }  
 }  
 totalResultadoSQL = **new** String[]{ resultadoSQL,String.*valueOf*(numFilas),String.*valueOf*(numColumnas)};  
 }  
  
 }**catch**(SQLException ex)  
 {  
 Log.*d*(**"Error de conexion"**, ex.getMessage());  
 }  
 **finally** {  
 **try** {  
 **if**(**rs** != **null**)  
 {  
 **rs**.close();  
 }  
 **st**.close();  
 **conexionMySQL**.close();  
 } **catch** (SQLException e)  
 {  
 e.printStackTrace();  
 }  
 }  
 **return** totalResultadoSQL;  
 }  
  
}

Luego creamos una clase Java con el nombre de Notificaciones donde se crea al servidor para que envie una notificación cuando una alerta se haya generado.

Notificaciones.java

**public class** Notificaciones **extends** Service {  
 **private final static** String ***CHANNEL\_ID***=**"NOTIFICACION"**;  
 **private final static int *NOTIFICACION\_ID***=1;  
 **private static final** String ***STRING\_PREFERENCES***=**"alertas"**;  
 **private static final** String ***num***=**"alerta.cantidad"**;  
  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
  
 @Override  
 **public void** onCreate(){  
  
 }  
  
 */\*\*  
 \*Este metodo sobreescrito permite iniciar un servicio en segundo plano con el fin de retornar una notificacion si una alerta se produce en las piscinas  
 \*/* @Override  
 **public int** onStartCommand(Intent intent, **int** flag, **int** idProcess){  
 **if** (ObtenerEstado()) {  
 mostrarResultados();  
 startService(**new** Intent(**new** Intent(**this**, Notificaciones.**class**)));  
 }  
 **return *START\_STICKY***;  
  
 }  
  
 */\*\*  
 \* Este metodo permite detener el servicio iniciado, deteniendose en el momento que el usuario cierra sesion en el aplicativo.  
 \*/* @Override  
 **public void** onDestroy(){  
 SharedPreferences preferences= (SharedPreferences) getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 preferences.edit().putBoolean(**"servicio"**,**false**).apply();  
 }  
  
 @Override  
 **public** IBinder onBind(Intent intent) {  
 **return null**;  
 }  
  
 */\*\*  
 \* Este metodo permite generar una notificacion del aplicativo  
 \*/* **public void** generarNotificaicon(){  
 **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***O***){  
 CharSequence name =**"Alerta"**;  
 NotificationChannel notificacionChanel= **new** NotificationChannel(***CHANNEL\_ID***,name, NotificationManager.***IMPORTANCE\_DEFAULT***);  
 NotificationManager notification =(NotificationManager) getSystemService(***NOTIFICATION\_SERVICE***);  
 notification.createNotificationChannel(notificacionChanel);  
 }  
  
 NotificationCompat.Builder builder=**new** NotificationCompat.Builder(getApplicationContext(),***CHANNEL\_ID***);  
 builder.setSmallIcon(R.drawable.***alert***);  
 builder.setContentTitle(**"Alerta"**);  
 builder.setContentText(**"Ha ocurrido un cambio de parametros en el agua."**);  
 builder.setColor(Color.***BLUE***);  
 builder.setPriority(NotificationCompat.***PRIORITY\_DEFAULT***);  
 builder.setLights(Color.***MAGENTA***,1000,1000);  
 builder.setVibrate(**new long**[]{1000,1000,1000,1000});  
 builder.setDefaults(Notification.***DEFAULT\_SOUND***);  
 NotificationManagerCompat not=NotificationManagerCompat.*from*(getApplicationContext());  
 not.notify(***NOTIFICACION\_ID***,builder.build());  
 }  
  
 */\*\*  
 \* Este metodo permite determinar si ha sido agragada un nuevo dato en la tabla de alertas de la base de datos, generando una notificacion sobre esto.  
 \*/* **public void** mostrarResultados()  
 {  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"SELECT \* FROM Alerta;"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 String filas=resultadoSQL[1];  
 **int** filasanteriores=Integer.*parseInt*(ObtenerAlertas());  
 **if**(Integer.*parseInt*(filas)>filasanteriores){  
 guardarAlerta(filas);  
 generarNotificaicon();  
 }  
  
 }**catch**(Exception ex) { }  
 }  
  
 */\*\*  
 \* Este metodo permite guardar la ultima cantidad de datos guardados en la tabla de alertas  
 \** ***@param s*** *\*/* **public void** guardarAlerta(String s){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 preferences.edit().putString(***num***,s).apply();  
 }  
  
 */\*\*  
 \*Permite guardar el estado de inicio de sesion, siendo true si ha iniciado algun usuario o false si no lo ha hecho o ha cerrado sesion  
 \*/* **public static void** guardarEstado(Context c, **boolean** s){  
 SharedPreferences preferences=c.getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 preferences.edit().putBoolean(**"servicio"**,s).apply();  
 }  
  
 */\*\*  
 \* Permtie obtener el dato de la cantidad de los ultimos datos guardados en la tabla de alertas  
 \** ***@return*** *\*/* **public** String ObtenerAlertas(){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 **return** preferences.getString(***num***,**"0"**);  
 }  
  
 */\*\*  
 \* Permite Obtener los datos del estado de la app  
 \** ***@return*** *\*/* **public boolean** ObtenerEstado(){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 **return** preferences.getBoolean(**"servicio"**,**false**);  
 }  
  
}