Proyecto Final

Redes de Computadoras III

Profr. Francisco Guillermo Gutierrez Nájera

Alumno:

Benjamín Dorantes García

Contenido

Introducción

Según datos del INEGI, actualmente viven en el estado de Aguascalientes 1,312,544 habitantes, de los cuales, 472,516 diariamente se desplazan en camión urbano, haciéndolo el principal medio de transporte público en Aguascalientes. [1]

"El problema principal de los usuarios del sistema de transporte colectivo en la ciudad de Aguascalientes: desconocen el tiempo que tardan en pasar las unidades de camiones urbanos a las estaciones lo cual, los orilla a invertir gran parte del tiempo total de traslado únicamente en esperar a que pase el camión que necesitan". [2]

La Asociación de Transportistas Urbanos y Suburbanos de Aguascalientes es la empresa concesionaria encargada de regular el transporte público en la capital del estado, cuenta con aproximadamente 700 camiones urbanos en operación, de los cuales 256 están fuera de norma. [3]

"A partir de este problema, se consideró útil … realizar el seguimiento de los trayectos de los camiones que actualmente están dentro de la norma sin necesidad de invertir mayor cantidad de recursos económicos en personal de monitoreo presencial y así mismo tener un mayor control de las unidades que están operando para garantizar la seguridad de usuarios y operadores". [2]

La elaboración de este proyecto se tomará en cuenta como parte de las pruebas de concepto a realizarse para sustentar y justificar la realización de mi Trabajo Terminal para obtener el grado de Ingeniero En Sistemas Computacionales en la Escuela Superior de Cómputo del Instituto Politécnico Nacional.

El sistema propuesto en el Trabajo Terminal tendrá dos objetivos: por una parte, que los usuarios conozcan los tiempos aproximados de llegada a la estación de los camiones de las distintas rutas, mediante módulos de información instalados en dichas estaciones y mediante una aplicación móvil en donde las personas podrán obtener una mayor cantidad de datos relevantes para su viaje como: el mapa de todas las rutas, el tiempo estimado de viaje, el tiempo de llegada de otras unidades, entre otros datos a definir durante el desarrollo del Trabajo Terminal.

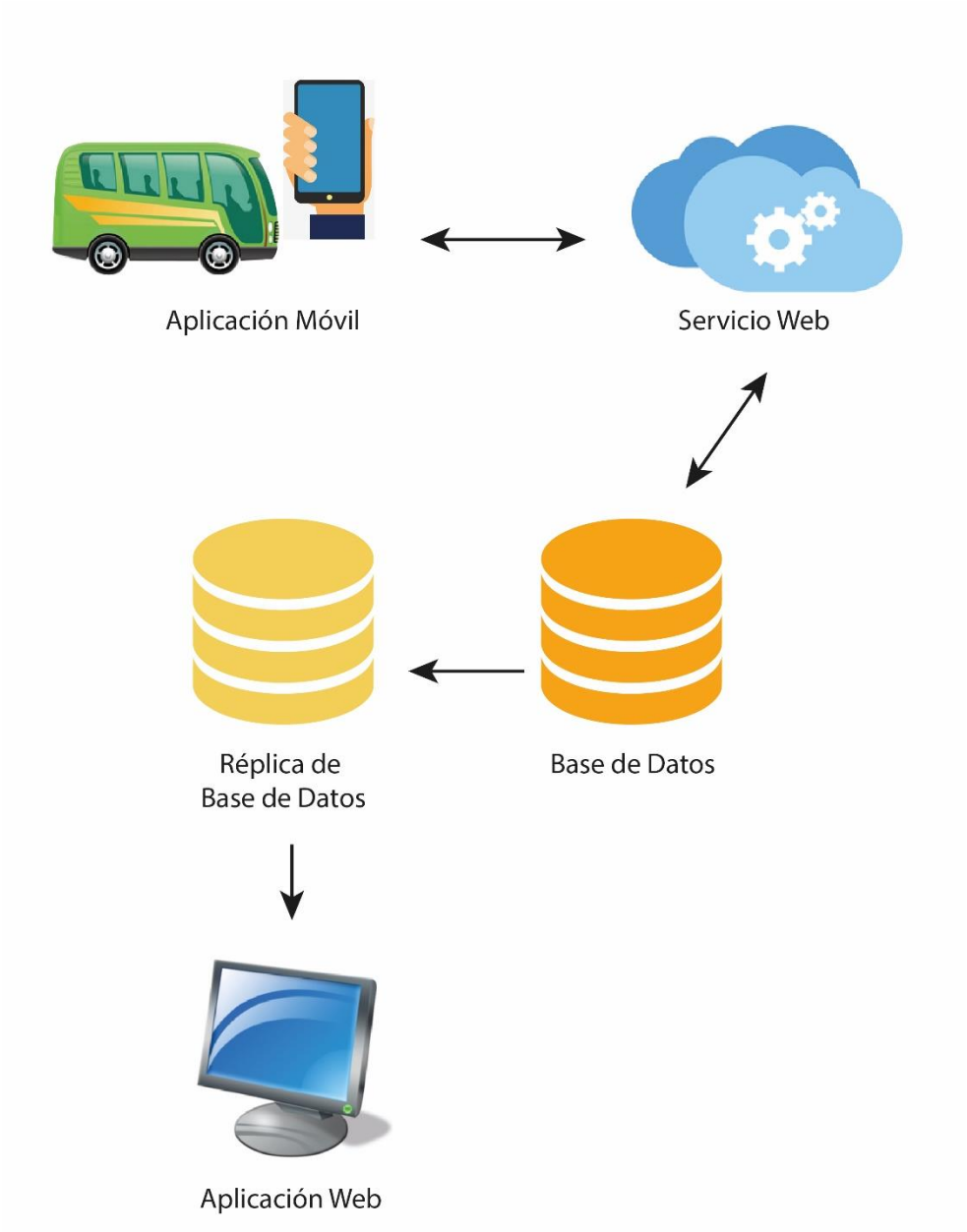
Por otro lado, el sistema permitirá que el personal encargado del Sistema de Transporte Público sea capaz de revisar de manera detallada el recorrido de las unidades de camiones urbanos monitoreadas para verificar que sigan de manera adecuada las rutas establecidas y con esto, tener un mayor control de los camiones que se encuentran operando.

La finalidad de este proyecto será probar las tecnologías de geolocalización de Android y las herramientas que nos brindan las APIs de Google Maps para conocer sus ventajas, desventajas y limitaciones y de esta manera saber si es una opción viable para utilizarla completamente en el Trabajo Terminal, solo una parte o descartarla totalmente.

Descripción General del Proyecto

Este proyecto es un sistema distribuido que almacena información sobre la fecha y hora en la que un camión urbano pasa por una estación de una determinada ruta. Esto lo hace mediante una aplicación móvil para dispositivos Android que consume mediante SOAP un servicio web alojado en un servidor Ubuntu; a su vez, este web service consulta y almacena información en una base de datos MySQL que está replicada en otro servidor, esta réplica es consultada por una aplicación web que utiliza las APIs de Google Maps para mostrar las visitas que realizan los camiones a cada una de las estaciones de las rutas establecidas.

Lo anterior se puede ver de manera gráfica en el siguiente diagrama:



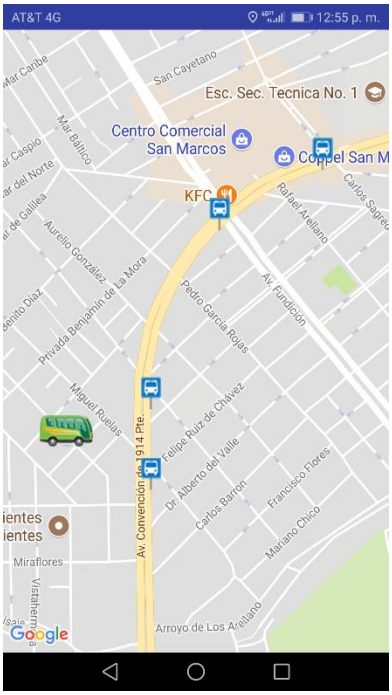
Componentes

Aplicación Móvil

Esta aplicación desarrollada para dispositivos con sistema operativo Android tiene como finalidad ir detectando la ubicación del dispositivo utilizando las APIs de Google Maps y aprovechando el GPS de este y así, obtener las coordenadas de la ubicación actual. Cada que hay un movimiento de posición, obtiene las nuevas coordenadas y hace una llamada al web service para compararlas con las coordenadas de las estaciones de la ruta que se seleccionó en la pantalla de inicio y en caso de que el dispositivo se encuentre en un rango de 5 metros de una estación, registrar la fecha y hora de la visita.

En la pantalla de inicio, el usuario ingresa las placas del camión (esto con la finalidad de tener un identificador) y selecciona la ruta por la cual va a pasar, al dar clic en aceptar, la aplicación hace una llamada al web service para obtener la ubicación de todas las estaciones de la ruta y se visualiza la pantalla principal, donde se observa la ubicación de las estaciones de la ruta, al dar clic en alguna se muestra su nombre y dirección, y la ubicación actual del camión (al dar clic en él se muestran sus placas y la dirección actual), esta se irá actualizando cada que haya un cambio de ubicación y al estar dentro del rango de 5 metros de una estación, se mostrará un cuadro de texto que indica “Estoy en la Estación x”.

Como se mencionó en la descripción del proyecto, esta aplicación consume el servicio web mediante el protocolo SOAP, este protocolo establece la comunicación mediante un archivo llamado WSDL (ver anexos) donde se establecen los métodos que contiene el servicio web, así como los parámetros que requiere cada uno de ellos.



Servicio Web

El servicio web fue programado en Java y está alojado en una instancia de Amazon Web Services corriendo con Ubuntu. Está constituido por dos métodos: getStations y checkIn, (ver código en anexos) a continuación, se explican con mayor detalle.

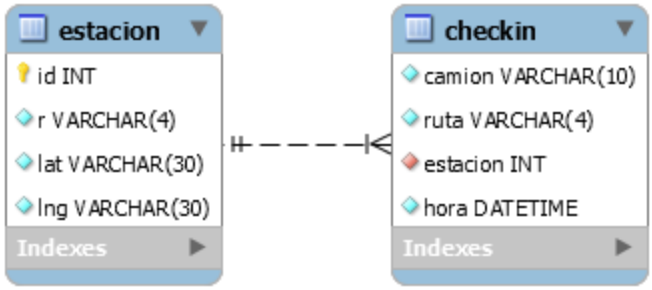
El método getStations recibe únicamente como parámetro un String con el número de la ruta y retorna un String con formato JSON que contiene toda la información referente a las estaciones que conforman la ruta que recibió como parámetro. El formato JSON que retorna es el siguiente: `{“estaciones”:[{“id”:“”, “latitud”:“”,“longitud”:“”}]}`. Lo que hace principalmente este método es conectarse a la base de datos y realizar una consulta, hecha la consulta, construye la cadena JSON que será devuelta.

El método checkIn es más complejo, recibe como parámetros tres Strings: placas, ruta y estación y devuelve un String con un “OK” o un “NO” dependiendo si el registro a la base de datos se realizó de manera correcta o no. Este método primero obtiene la fecha y hora actual, después hace una consulta sobre el último registro que existe en la base de datos con esas placas y esa estación, con el fin de que si el camión permanece parado en una estación por un tiempo no se genere una gran cantidad de registros, una vez hecha esa consulta, compara las fechas y horas y en caso de que hayan pasado más de 3 minutos del último registro o no exista registro anterior, registra en la base de datos la visita del camión a esa estación con la hora y fecha actuales.

Base de Datos

La base de datos está construida en MySQL, está alojada en la misma instancia que el Servicio Web, sin embargo, cuenta con una replicación maestro – esclavo en otra instancia de Ubuntu dentro de Amazon Web Services.

Está constituida únicamente por dos tablas: estación y checkin, las cuales están relacionadas mediante el id de la estación (ver script en anexos) como se puede observar en el siguiente diagrama:

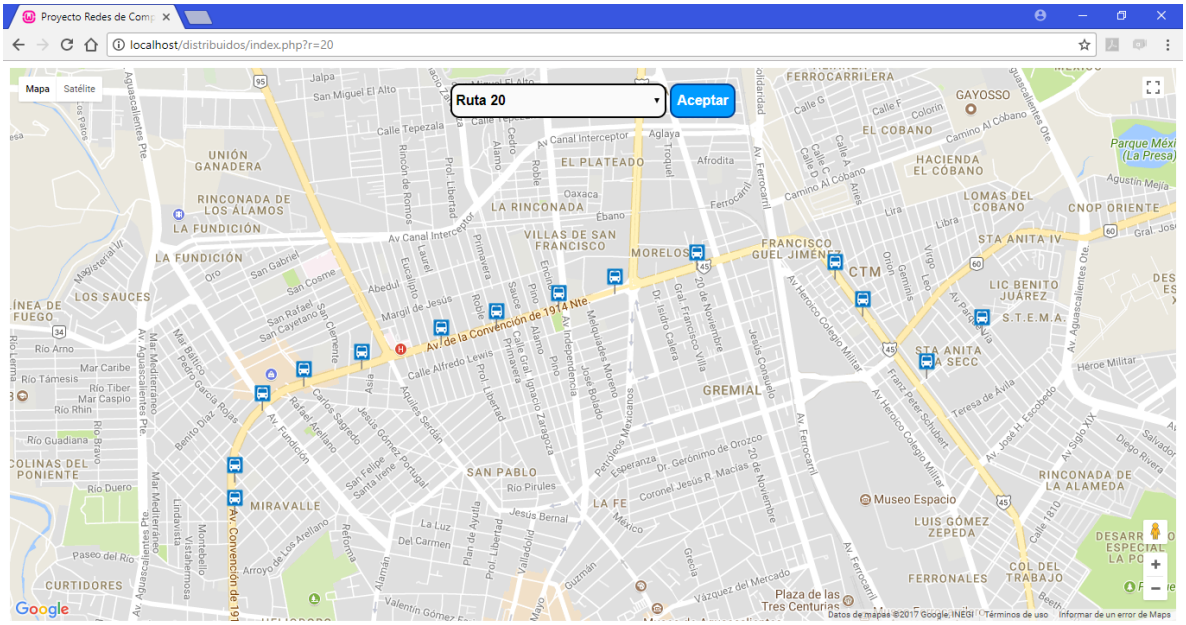


Esta base de datos tiene la única finalidad de almacenar la información para ser consultada después por la aplicación web. En la tabla estación se almacena las coordenadas (latitud y longitud) de las estaciones que conforman las rutas y en checkin la fecha y hora en las que los camiones pasan por las estaciones.

Aplicación Web

La aplicación web realiza con PHP dos consultas, la primera sobre la ubicación de todas las estaciones de la ruta selecciona por el usuario y con la segunda obtiene las visitas registradas que tiene cada estación.

Una vez que se tiene esta información, con la ayuda de las APIs de Google Maps, se carga un mapa en el cual se muestran las estaciones de la ruta seleccionada por el usuario y al hacer clic en alguna se muestra un cuadro de texto con los últimos registros de visitas. (ver código en anexos)



Ejecución y pruebas

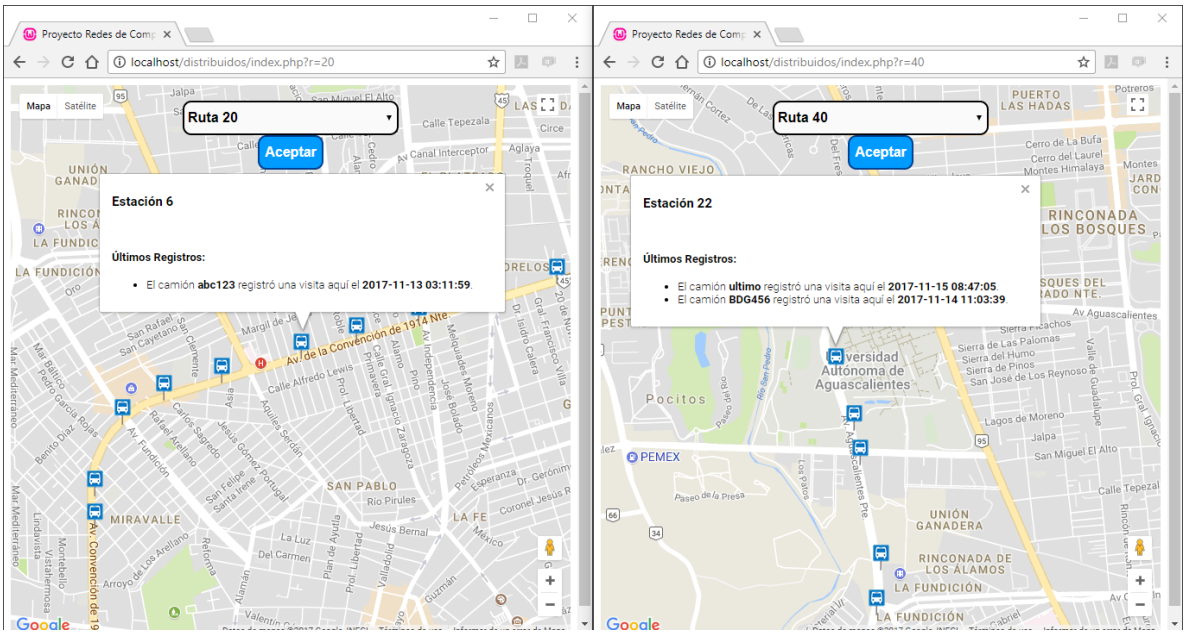
Para las pruebas se utilizaron únicamente algunas estaciones de las rutas 20 y 40:

```
mysql> select * from estacion;
+----+------+--------------------+---------------------+
| id | r    | lat                | lng                 |
+----+------+--------------------+---------------------+
|  1 | 20   | 21.890015411373476 | -102.312390966684   |
|  2 | 20   | 21.89150869859405  | -102.312390966684   |
|  3 | 20   | 21.894833695291315 | -102.31101767566838 |
|  4 | 20   | 21.89594864688259  | -102.30895773914494 |
|  5 | 20   | 21.896745035537563 | -102.30606095340886 |
|  6 | 20   | 21.897840062673513 | -102.30211274173894 |
|  7 | 20   | 21.89859662196283  | -102.29938761737981 |
|  8 | 20   | 21.8994527236842   | -102.29627625492253 |
|  9 | 20   | 21.900189365235494 | -102.29350821521916 |
| 10 | 20   | 21.90132418395853  | -102.2894097998444  |
| 11 | 20   | 21.900886184820525 | -102.28254334476628 |
| 12 | 20   | 21.899164038795124 | -102.28117005375066 |
| 13 | 20   | 21.896287112601403 | -102.27800504711308 |
| 14 | 20   | 21.89834775421563  | -102.27527992275395 |
| 15 | 40   | 21.891439012203044 | -102.31682197597661 |
| 16 | 40   | 21.894485271129156 | -102.31688634899297 |
| 17 | 40   | 21.899024673464357 | -102.31693999317326 |
| 18 | 40   | 21.90194136227556  | -102.31631772068181 |
| 19 | 40   | 21.90402183078874  | -102.31612460163274 |
| 20 | 40   | 21.908899365152138 | -102.3171438410584  |
| 21 | 40   | 21.910482039399803 | -102.31744424846806 |
| 22 | 40   | 21.91308992967481  | -102.31835619953313 |
| 23 | 40   | 21.916205400118173 | -102.31476203945317 |
+----+------+--------------------+---------------------+
23 rows in set (0.00 sec)
```

Se realizaron en total tres viajes para las pruebas, uno en la ruta 20 (camión abc123) y dos en la ruta 40 (camiones BDG456 y ultimo), al realizar la consulta en la base de datos se pueden ver los registros

```
mysql> select * from checkin;
+--------+------+----------+---------------------+
| camion | ruta | estacion | hora                |
+--------+------+----------+---------------------+
| abc123 | 20   |        2 | 2017-11-13 03:04:53 |
| abc123 | 20   |        3 | 2017-11-13 03:07:11 |
| abc123 | 20   |        4 | 2017-11-13 03:09:13 |
| abc123 | 20   |        5 | 2017-11-13 03:09:55 |
| abc123 | 20   |        6 | 2017-11-13 03:11:59 |
| abc123 | 20   |        7 | 2017-11-13 03:14:01 |
| abc123 | 20   |        8 | 2017-11-13 03:15:29 |
| abc123 | 20   |        9 | 2017-11-13 03:17:21 |
| abc123 | 20   |       10 | 2017-11-13 03:19:03 |
| abc123 | 20   |       11 | 2017-11-13 03:21:39 |
| abc123 | 20   |       12 | 2017-11-13 03:22:03 |
| abc123 | 20   |       13 | 2017-11-13 03:24:09 |
| abc123 | 20   |       14 | 2017-11-13 03:25:05 |
| BDG456 | 40   |       15 | 2017-11-14 10:57:13 |
| BDG456 | 40   |       16 | 2017-11-14 10:57:49 |
| BDG456 | 40   |       17 | 2017-11-14 11:00:51 |
| BDG456 | 40   |       18 | 2017-11-14 11:01:13 |
| BDG456 | 40   |       19 | 2017-11-14 11:01:47 |
| BDG456 | 40   |       20 | 2017-11-14 11:02:19 |
| BDG456 | 40   |       21 | 2017-11-14 11:03:19 |
| BDG456 | 40   |       22 | 2017-11-14 11:03:39 |
| BDG456 | 40   |       23 | 2017-11-14 11:04:47 |
| ultimo | 40   |       16 | 2017-11-15 08:44:18 |
| ultimo | 40   |       17 | 2017-11-15 08:44:55 |
| ultimo | 40   |       18 | 2017-11-15 08:45:21 |
| ultimo | 40   |       19 | 2017-11-15 08:45:39 |
| ultimo | 40   |       20 | 2017-11-15 08:46:15 |
| ultimo | 40   |       21 | 2017-11-15 08:46:45 |
| ultimo | 40   |       22 | 2017-11-15 08:47:05 |
+--------+------+----------+---------------------+
29 rows in set (0.00 sec)
```

Al abrir la aplicación web y seleccionar cada una de las rutas y una estación aleatoria se puede observar la información sobre los viajes realizados



Conclusiones

Tomando este proyecto como prueba de concepto para la realización de mi trabajo terminal pude realizar las siguientes observaciones:

- La geolocalización de Android, así como las APIs de Google Maps son relativamente fáciles de implementar sin embargo no considero óptima su utilización por el tipo y magnitud de proyecto que se busca desarrollar. Principalmente porque el GPS siempre está cambiando de coordenadas aunque el celular no se mueva, lo que aumentaría el nivel de procesamiento cuando aumente el número de peticiones al servidor.
- Si bien no las descarto totalmente, en mi opinión convendría más utilizar algún tipo de sensor en las estaciones para monitorear a los camiones y registrar la visita de los camiones, lo que evitaría dejarle al conductor la responsabilidad de tener una aplicación abierta todo el camino gastando sus datos móviles, podría ocasionar accidentes por usar el celular mientras conduce y se tendría mayor control de las peticiones al servidor.
- Observando el comportamiento de la base de datos, convendrá definir bien qué información es útil que se almacene de cada registro y la cantidad de registros y el tiempo que estarán almacenados, ya que al ser un proyecto que pretende tener gran magnitud, la cantidad de información puede crecer en gran medida.
- La versión de las APIs de Google Maps para Javascript es una muy buena opción para la parte de la aplicación web para la concesionaria, faltaría revisar si permite el monitoreo en tiempo real de las unidades para aceptarla como herramienta de desarrollo, así como investigar más a fondo los costos por utilizarla en producción.
- El protocolo SOAP y el formato JSON nos podría ser de gran utilidad al momento de transmitir información de un lado a otro.

Referencias

[1] Instituto Nacional de Estadística, Geografía e Informática, “Transporte urbano de pasajeros”, 2017. [En línea] Disponible en: http://www.inegi.org.mx/est/contenido/proyectos/registros/economicas/transporte/default.aspx [Accedido: 03-feb-2017]

[2] Dorantes García, B., Rivera Pérez, L. A., “Sistema de programación, logística, monitoreo e información para el transporte público de la ciudad de Aguascalientes, Aguascalientes, México”. 2017, Protocolo de Trabajo Terminal, Escuela Superior de Cómputo, Instituto Politécnico Nacional.

[3] F. González Serna, *Propuesta de transporte público para Aguascalientes*, Ciudad de México: Central Ciudadano y Consumidor: 2014

Anexos

Código de las llamadas al Servicio Web desde la Aplicación Móvil

```
1.  private class getStations extends AsyncTask<String, String, String>{
2.      static final String NAMESPACE = "http://WS/";
3.      static final String METHODNAME = "getStations";
4.      static final String URL = "http://18.217.115.39:8080/DistribuidosWS/DistribuidosWS?wsdl"
    ;
5.      static final String SOAP_ACTION = NAMESPACE + METHODNAME;
6.
7.      protected String doInBackground(String... params) {
8.          SoapObject request = new SoapObject(NAMESPACE, METHODNAME);
9.          request.addProperty("route", params[0]);
10.         SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER1
    1);
11.         envelope.dotNet = false;
12.         envelope.setOutputSoapObject(request);
13.         HttpTransportSE transporte = new HttpTransportSE(URL);
14.         try{
15.             transporte.call(SOAP_ACTION, envelope);
16.             SoapPrimitive response = (SoapPrimitive) envelope.getResponse();
17.             System.out.println(response.toString());
18.             stationsJSON =  response.toString();
19.             JSONObject obj = new JSONObject(stationsJSON);
20.             JSONArray stationArray = obj.getJSONArray("estaciones");
21.             for (int i = 0; i < stationArray.length(); i++) {
22.                 JSONObject stationDetail = stationArray.getJSONObject(i);
23.                 Station auxST = new Station();
24.                 auxST.setId(stationDetail.getString("id"));
25.                 auxST.setLatitude(stationDetail.getString("latitud"));
26.                 auxST.setLongitude(stationDetail.getString("longitud"));
27.                 stations.add(auxST);
28.             }
29.         } catch (Exception e) {
30.             System.out.println(e.toString());
31.         }
32.         return "HAY " + stations.size() + " ESTACIONES :D";
33.     }
34.
35.     protected void onPostExecute(String result) {
36.         System.out.println(result);
37.         for(int i = 0; i < stations.size(); i++) {
38.             Station st = new Station();
39.             st = stations.get(i);
40.             LatLng coordenadas = new LatLng(Double.parseDouble(st.getLatitude()), Double.par
    seDouble(st.getLongitude()));
41.             try {
42.                 mMap.addMarker(new MarkerOptions().position(coordenadas).title("Estacion " +
     st.getId()).snippet(setLocation(Double.parseDouble(st.getLatitude()), Double.parseDouble(st
    .getLongitude()))).icon(BitmapDescriptorFactory.fromResource(R.drawable.st)));
43.             } catch (IOException e) {
44.                 e.printStackTrace();
45.             }
46.         }
47.     }
48. }
```

```
1.  private class checkIn extends AsyncTask<String, String, String>{
2.      static final String NAMESPACE = "http://WS/";
3.      static final String METHODNAME = "checkIn";
4.      static final String URL = "http://18.217.115.39:8080/DistribuidosWS/DistribuidosWS?wsdl"
    ;
5.      static final String SOAP_ACTION = NAMESPACE + METHODNAME;
6.
7.      @Override
8.      protected String doInBackground(String... params) {
9.          SoapObject request = new SoapObject(NAMESPACE, METHODNAME);
10.         request.addProperty("placas", params[0]);
11.         request.addProperty("ruta", params[1]);
12.         request.addProperty("estacion", params[2]);
13.         SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER1
    1);
14.         envelope.dotNet = false;
15.         envelope.setOutputSoapObject(request);
16.         HttpTransportSE transporte = new HttpTransportSE(URL);
17.         try{
18.             transporte.call(SOAP_ACTION, envelope);
19.             SoapPrimitive response = (SoapPrimitive) envelope.getResponse();
20.             Log.d("response", response.toString());
21.         } catch (Exception e) {
22.             e.printStackTrace();
23.         }
24.         return null;
25.     }
26. }
```

Código de los métodos del Servicio Web

```
1.  public String getStations(@WebParam(name = "route") String route) {
2.          System.out.println("GET STATIONS :) Route= "+route);
3.          Conexion bd = new Conexion("localhost", "distribuidos", "root", "root");
4.          bd.conectar();
5.          String sql = "SELECT * FROM estacion WHERE r='"+route+"';";
6.          String sxml = "";
7.          ResultSet rs = bd.consulta(sql);
8.          LinkedList<String> auxL = new LinkedList<String>();
9.          try {
10.
11.             while(rs.next()){
12.                 String auxS = "{\"id\":"+rs.getString("id")+", \"latitud\":"+rs.getString("l
    at")+", \"longitud\":"+rs.getString("lng")+"}";
13.                 auxL.add(auxS);
14.             }
15.
16.         } catch (SQLException ex) {
17.             Logger.getLogger(DistribuidosWS.class.getName()).log(Level.SEVERE, null, ex);
18.         }
19.         sxml += "{\"estaciones\": [";
20.         String coma = ",";
21.         for(int i = 0; i < auxL.size(); i++){
22.             if(i == auxL.size() - 1)
23.                 coma = "";
24.             sxml += auxL.get(i) + coma;
25.         }
26.         sxml += "]}";
27.         System.out.println(sxml);
28.         return sxml;
29. }
```

```
1.  public String checkIn(@WebParam(name = "placas") String placas, @WebParam(name = "ruta") Str
    ing ruta, @WebParam(name = "estacion") String estacion) {
2.          Locale l = new Locale("es","MX");
3.          Calendar cal = Calendar.getInstance(TimeZone.getTimeZone("America/Mexico_City"));
4.          String datetime = "";
5.          datetime = (cal.get(Calendar.YEAR) + "-
    " + String.format("%02d",cal.get(Calendar.MONTH)) + "-
    " + String.format("%02d",cal.get(Calendar.DAY_OF_MONTH)));
6.          datetime += " " + (String.format("%02d",cal.get(Calendar.HOUR))+":"+String.format("%
    02d",cal.get(Calendar.MINUTE))+":"+String.format("%02d",cal.get(Calendar.SECOND)));
7.          System.out.println(datetime);
8.          Conexion bd = new Conexion("localhost", "distribuidos", "root", "root");
9.          bd.conectar();
10.         String sql = "SELECT MAX(hora) as hora FROM checkin WHERE camion ='"+placas+"' AND r
    uta = '" + ruta + "' AND estacion = " + estacion + ";";
11.         ResultSet rs = bd.consulta(sql);
12.         String horaMax = null;
13.         try {
14.             rs.next();
15.             horaMax = rs.getString("hora");
16.             System.out.println(horaMax);
17.         } catch (SQLException ex) {
18.             Logger.getLogger(DistribuidosWS.class.getName()).log(Level.SEVERE, null, ex);
19.         }
20.         int minutes = 0;
21.         if(horaMax != null){
22.             horaMax = horaMax.split(" ")[1];
23.             String auxDT = datetime.split(" ")[1];
24.             LocalTime max = LocalTime.parse(horaMax);
25.             LocalTime act = LocalTime.parse(auxDT);
26.             minutes = (int) ChronoUnit.MINUTES.between(max, act);
27.             System.out.println(minutes);
28.         }
29.         if(horaMax == null || minutes > 3){
30.             sql = "INSERT INTO checkin(camion, ruta, estacion, hora) VALUES('"+placas+"','"+
    ruta+"',"+estacion+",'"+datetime+"');";
31.             System.out.println(sql);
32.             if(bd.abc(sql))
33.                 return "OK";
34.             else
35.                 return "NO";
36.             }
37.         else
38.             return "OK";
39. }
```

Archivo WSDL

```
1.  <?xml version="1.0"?>
2.  <!--
3.   Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2-hudson-
    740-.
4.  -->
5.  <!--
6.   Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2-hudson-
    740-.
7.  -->
8.  <definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
    utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-
    policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w
    3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:
    tns="http://WS/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.
    org/wsdl/" targetNamespace="http://WS/" name="DistribuidosWS">
9.    <types>
10.     <xsd:schema>
11.       <xsd:import namespace="http://WS/" schemaLocation="http://18.217.115.39:8080/Distribui
    dosWS/DistribuidosWS?xsd=1"/>
```

```
    </xsd:schema>
  </types>
  <message name="getStations">
    <part name="parameters" element="tns:getStations"/>
  </message>
  <message name="getStationsResponse">
    <part name="parameters" element="tns:getStationsResponse"/>
  </message>
  <message name="checkIn">
    <part name="parameters" element="tns:checkIn"/>
  </message>
  <message name="checkInResponse">
    <part name="parameters" element="tns:checkInResponse"/>
  </message>
  <portType name="DistribuidosWS">
    <operation name="getStations">
      <input wsam:Action="http://WS/DistribuidosWS/getStationsRequest" message="tns:getStations"/>
      <output wsam:Action="http://WS/DistribuidosWS/getStationsResponse" message="tns:getStationsResponse"/>
    </operation>
    <operation name="checkIn">
      <input wsam:Action="http://WS/DistribuidosWS/checkInRequest" message="tns:checkIn"/>
      <output wsam:Action="http://WS/DistribuidosWS/checkInResponse" message="tns:checkInResponse"/>
    </operation>
  </portType>
  <binding name="DistribuidosWSPortBinding" type="tns:DistribuidosWS">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="getStations">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
    <operation name="checkIn">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="DistribuidosWS">
    <port name="DistribuidosWSPort" binding="tns:DistribuidosWSPortBinding">
      <soap:address location="http://18.217.115.39:8080/DistribuidosWS/DistribuidosWS"/>
    </port>
  </service>
</definitions>
```

Aplicación Web

```
<script>
    <?php
        require("conexion.php");
        if(emptyempty($_GET["r"]))
            $ruta = "20";
        else
            $ruta = $_GET["r"];
        $sql = "SELECT lat, lng FROM estacion WHERE id = (SELECT MIN(id) FROM estacion WHERE r='".$ruta."');";
        $consulta = $conn->query($sql);
```

```
        while($row = $consulta->fetch_assoc()) {
            $lat = $row["lat"];
            $lng = $row["lng"];
        }
      ?>
      function initMap() {
        var lat = <?php echo $lat; ?>;
        var lng = <?php echo $lng; ?>;
        var map = new google.maps.Map(document.getElementById('map'), {
          zoom: 15,
          center: new google.maps.LatLng(lat, lng)
        });
        <?php
            $sql = "SELECT * FROM estacion WHERE r='".$ruta."';";
            $consulta = $conn->query($sql);
            $i = 0;
            while($row = $consulta->fetch_assoc()) {

        ?>
                var lat = <?php echo $row["lat"]; ?>;
                var lng = <?php echo $row["lng"]; ?>;
                var img = "st.png";
                var <?php echo "marker$i"; ?> = new google.maps.Marker({
                    position: new google.maps.LatLng(lat, lng),
                    map: map,
                    title: <?php echo "'Estación ".$row["id"]."'"; ?>,
                    icon: img
                });
        <?php
                $sql = "SELECT * FROM checkin WHERE estacion = ".$row["id"]." ORDER BY hora DESC;";
                $info  = "<div id='content'><h3>Estación ".$row["id"]."</h3><br>";
                $info .= "<h4><b>Últimos Registros: </b></h4>";
                $cons = $conn->query($sql);
                if($cons->num_rows <= 0)
                $info .= "<center>No hay registros en esta estación</center>";
                else{
                    $info .= "<ul>";
                    while($r = $cons->fetch_assoc()){
                        $info .= "<li>El camión <b>".$r["camion"]."</b> registró una visita aquí el <b>".$r["hora"]."</b>.</li>";
                    }
                    $info .= "</ul>";
                }
        ?>
                var contentString = "<?php echo $info; ?>";
                var <?php echo "infowindow$i"; ?> = new google.maps.InfoWindow({
                    content: contentString
                });
                <?php echo "marker$i"; ?>.addListener('click', function() {
                    <?php echo "infowindow$i"; ?>.open(map, <?php echo "marker$i"; ?>);
                });
        <?php
            $i++;
            }
        ?>
      }
    </script>
    <script async defer
    src="https://maps.googleapis.com/maps/api/js?key=[CLAVE DE API]&callback=initMap">
    </script>
```

Script para creación de la base de datos

```
CREATE SCHEMA IF NOT EXISTS `distribuidos` DEFAULT CHARACTER SET utf8 ;
USE `distribuidos` ;

CREATE TABLE IF NOT EXISTS `distribuidos`.`estacion` (
  `id` INT NOT NULL,
  `r` VARCHAR(4) NOT NULL,
  `lat` VARCHAR(30) NOT NULL,
  `lng` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `distribuidos`.`checkin` (
  `camion` VARCHAR(10) NOT NULL,
  `ruta` VARCHAR(4) NOT NULL,
  `estacion` INT NOT NULL,
  `hora` DATETIME NOT NULL,
  INDEX `estacion_idx` (`estacion` ASC),
  CONSTRAINT `estacion`
    FOREIGN KEY (`estacion`)
    REFERENCES `distribuidos`.`estacion` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Repositorio en Github

https://github.com/benjamin024/Proyecto-Final-Redes-III