UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Diseño del backoffice de un panel de control de un ayuntamiento

---

MEMORIA REALIZADA POR:
Carlos Bosch Tur

Grado de ingeniería informática
Curso 2016/2017

Tutor:
Manuel Llorca Alcón

Índice:

Índice de figuras:

1.-Introducción

El proyecto se basa en ofrecer de forma cómoda y agradable información al usuario que pueda servirle en su día a día al desplazarse por su ciudad.

Dentro del proyecto se abarcan distintos puntos de información que son:

- Tráfico.
- Meteorología.
- Autobuses.

Estos datos son recogidos en tiempo real y analizados para exponerlos al usuario para su uso y disfrute cuando los necesite.

1.1.-¿Qué es Cátedra Smart City?

Será la colaboración y desarrollo del proyecto de ciudad inteligente. La principal iniciativa de colaboración estará en el desarrollo de plataformas de servicios que, apoyándose en las principales tecnologías disponibles en la actualidad, logren definir un sistema de base de datos integrada en todos los procesos claves de la ciudad. Este objetivo principal se subdivide en objetivos específicos para la realización de proyectos que consigan un beneficio a la ciudad y a los ciudadanos. Todo ello en el marco del objeto y finalidades específicas de la Universidad.

1.2.-Problemas de la ciudad

Dentro de una ciudad nos podemos encontrar con innumerables problemas que pueden complicar el día a día de sus residentes, como pueden ser:

- Tráfico.
- Meteorología.
- Transportes urbanos.
- Desconocimiento de ubicaciones o de negocios.

1.3.-Necesidades de una ciudad moderna

Con la constante evolución de las tecnologías una ciudad también tiene que adaptarse a los nuevos tiempos, pudiendo ofrecer así información útil y en tiempo real al ciudadano de a pie para poder facilitarle su día a día, como pueden ser:

- Sensores climáticos.
- Cámaras de tráfico para ver la evolución de este.
- Precisión del horario del transporte público.
- Información relevante de la ciudad.

1.4.-Motivación

La motivación principal por la cual me decante a realizar este proyecto está dividida en dos partes:

- La principal motivación es el gran aprecio que le tengo a mi ciudad natal. Cuando vi que tenía la oportunidad de aportar algo a esta ciudad no dude en emprender este proyecto.
- En segundo lugar esta mi intención de ayudar y acercar el conocimiento a todos los ciudadanos ya que cuanta más información disponemos a nuestro alcance nuestras decisiones serán más acertadas.

2.-Objetivo

El objetivo principal del proyecto es el desarrollo de una página web informativa donde quedará englobada toda la información que pudiera ser relevante para el usuario y que a través de una interfaz agradable, este pudiera disponer de todos los datos en tiempo real para así poder tomar decisiones más acertadas a través de la información que recibe de ésta.

3.-Iniciando el Proyecto

Al iniciar el proyecto la primera duda que nos asalta es en qué lenguajes de programación y en qué herramientas vamos a dar forma al proyecto.

3.1.-Lenguajes de programación

Dividiremos esta parte en tres. La web donde vemos la información y se mueven algunos de los scripts de recuperación de datos, la base de datos y la aplicación de las cámaras.

3.1.1.-La web

La página web fue diseñada en **html** para la parte que ve el ciudadano y **css** para la parte que le da sus propiedades gráficas. Se optó por éstos ya que son los más estandarizados y los que menos problemas les darían al final a los ciudadanos ya que casi en la totalidad de dispositivos es compatible al 100%.

En la parte del servidor de la web se optó por **php** por su gran comunidad de usuarios de internet. Al ser una plataforma libre que cualquiera puede utilizar y al estar tan extendido sería muy fácil ampliarlo y extenderlo aún más.

3.1.2.-Base de datos

Para la base de datos se utilizó **Mysql** por su forma de trabajar con los datos y su facilidad de manipularlos y mostrarlos de forma correcta.

3.1.3.-Aplicación de las cámaras

Para recibir los datos de un socket y enviar los datos a nuestra base de datos se utilizó **java** por ser multiplataforma. También puede adaptarse a cualquier equipo en el que se encuentre para así poder enviar y recibir los datos de forma confiable.

4.-Recogida de datos

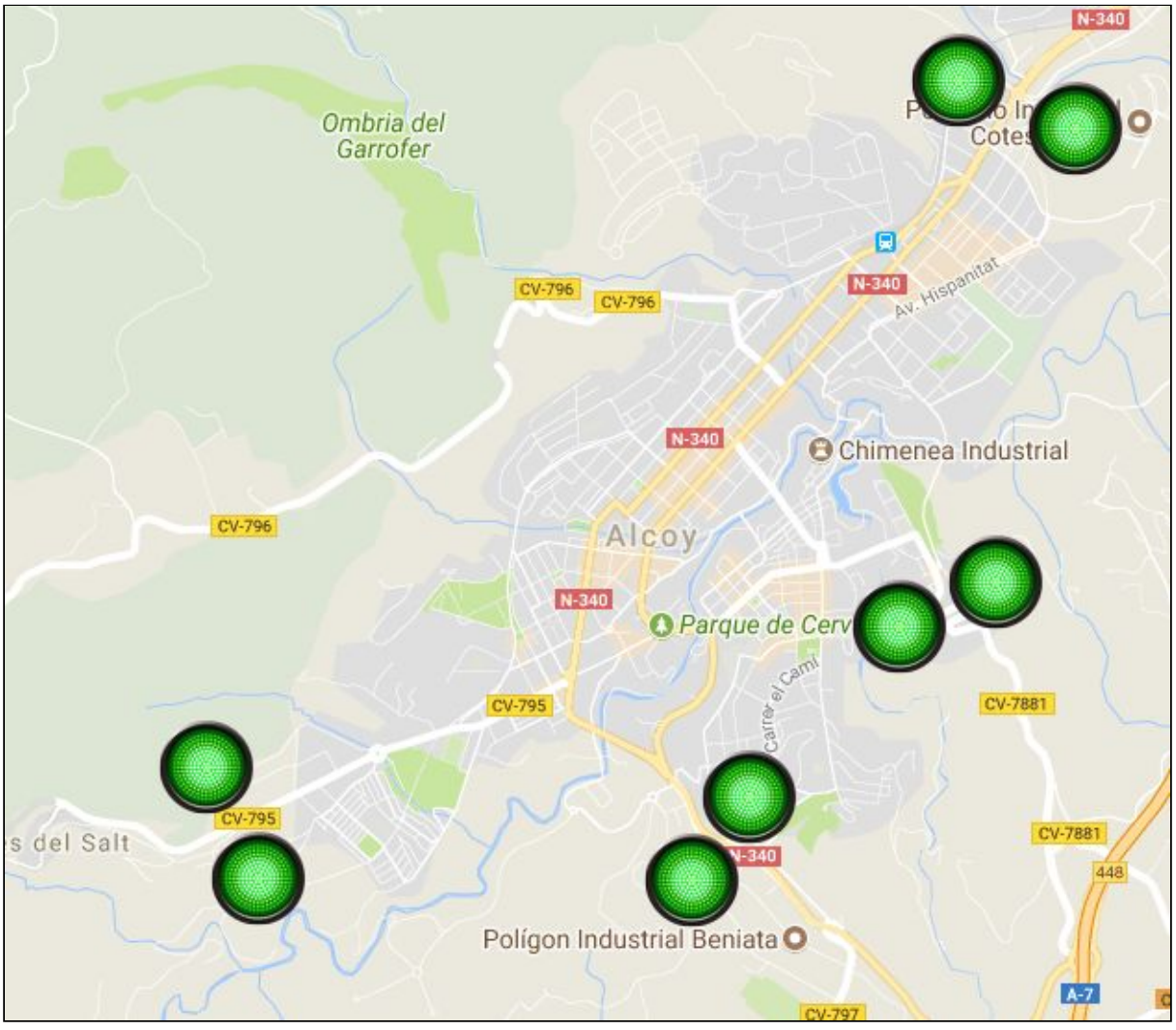
En el proyecto se han tenido que obtener los datos de formas muy diversas:

- ❖ Recogida de datos desde un socket a una base de datos.
- ❖ Acceder a los datos desde el código fuente de otra web.
  - ➢ Filtrado de datos y posteriormente mostrarlo de forma correcta.
- ❖ Obtención de los datos de un archivo xml externo.
  - ➢ Tras su filtrado se visualizan los datos en la web de forma agradable para el ciudadano.

Vamos a dividir esta sección en tres apartados para desglosar su recogida.

4.1.-Cámaras de tráfico

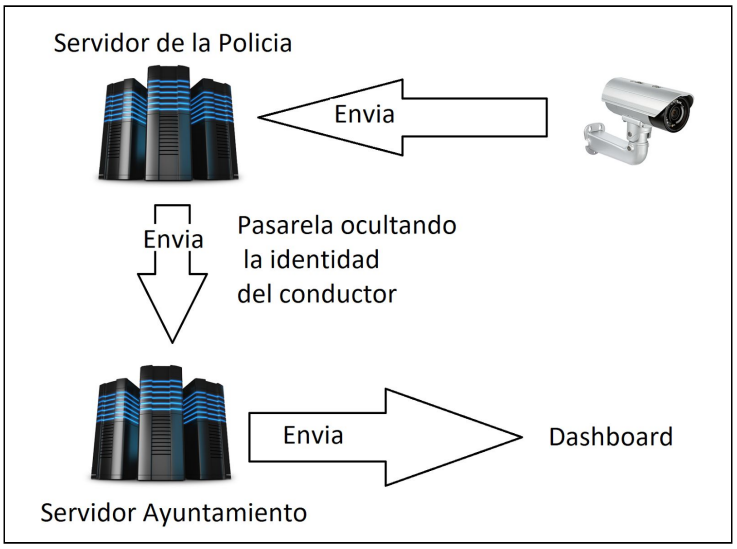
Al realizar este proyecto se han tenido a disposición cinco cámaras. En el siguiente mapa se puede observar la posición de éstas:



1.- Foto aérea de las cámaras que hay actualmente en Alcoy.

Estas cámaras mandan la información al servidor de la policía de Alcoy. Éstas recogen entre otros datos las matrículas de los vehículos que entran y salen de la ciudad. Al ser datos confidenciales se tuvo que crear un programa en java que ocultara la matrícula del conductor y que enviara la información después a la base de datos del ayuntamiento.

4.1.1.-Esquema de la recogida de datos



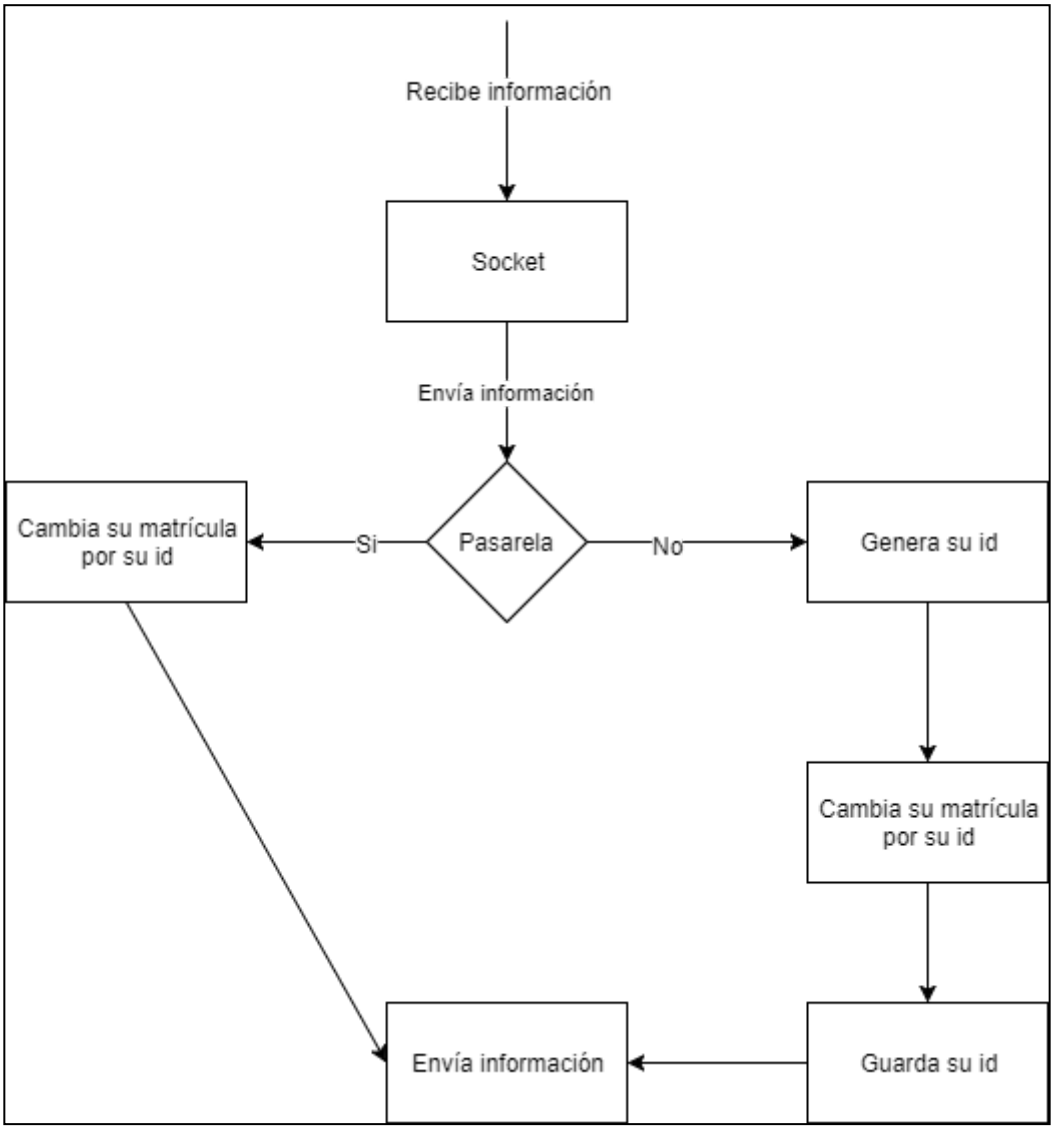
2.- Esquema de recogida de datos de las cámaras.

4.1.2.-Pasarela que oculta la identidad del conductor

Esta pasarela tenía que hacer tres cosas:

- Escuchar de un socket como la información iba entrando(matrícula y resto de datos).
- Comprobar si ya existía esta matrícula antes en la base de datos de la policía.
- Tras la comprobación pueden ocurrir dos cosas:
  - Que la matrícula **exista** en la base de datos, por lo tanto se enviará a la base de datos del ayuntamiento cambiando su matrícula por su id.
  - Que la matrícula **no exista** en la base de datos, por lo que se genera una id para ocultar la identidad del conductor. Posteriormente se almacena para su uso futuro y se envían los datos al servidor del ayuntamiento.

4.1.3.-Diagrama de flujo



3.- Diagrama de flujo de cómo se transforma la información de las cámaras.

4.1.4.-Código fuente de la Pasarela

La pasarela se resume en:

- **Servidor.java**: Este archivo es el que contiene toda la parte de las consultas y los accesos a la base de datos como la puesta en marcha del servidor.
- **Conexion.java**: Este archivo tiene las variables necesarias para que el servidor arranque y quede a la escucha de un puerto y de una ip.
- **ConexionMySQL.java**: Este archivo tiene todo lo relacionado con la conexión a la base de datos.
- **MainServidor.java**: Pone en marcha el servidor.

4.1.5.-Código fuente Servidor.java

En el siguiente fragmento de código vemos elementos comunes que vamos a ir necesitando como son las librerías que vamos a utilizar, y las clases públicas que envuelven todo el código.

```
1  package sockets;
2
4  *  Author: Carlos Bosch Turi
15 import java.io.BufferedReader;
24
25
26 public class Servidor extends Conexion //Se hereda de conexión para hacer uso de los sockets
27 {
28     public Servidor() throws IOException{super("servidor");} //Se usa el constructor para servidor de Conexion
29
30     public void startServer()//Método para iniciar el servidor
31     {
32         try
33         {
```

Podemos ver a continuación como el servidor se pone en espera de algún cliente. Una vez el socket recibe la conexión se le comunica que fue aceptada la conexión y se empiezan a recibir los datos.

```
34             System.out.println("Esperando..."); //El servidor se pone a la espera de conexiones
35             cs = ss.accept(); //Acepta y comienza el socket y espera una conexión desde un cliente
36             System.out.println("Cliente en linea");
37             //Se obtiene el flujo de salida del cliente para enviarle mensajes
38             salidaCliente = new DataOutputStream(cs.getOutputStream());
39             //Se le envía un mensaje al cliente usando su flujo de salida
40             salidaCliente.writeUTF("Peticion recibida y aceptada");
41             //Se obtiene el flujo entrante desde el cliente
42             BufferedReader entrada = new BufferedReader(new InputStreamReader(cs.getInputStream()));
43
```

En el bucle recoge la cadena de texto, se separa la cadena de texto por el campo identificador “**,**” y se crean dos variables, una que servirá como contador que será “**i**” y el array que almacenará la cadena que será “**datos**”.

```
44             while((mensajeServidor = entrada.readLine()) != null) //Mientras haya mensajes desde el cliente
45             {
46                 //Se muestra por pantalla el mensaje recibido
47                 System.out.println(mensajeServidor);
48
49                 // Pasamos a separar el cadena de texto por comas y volcarlo en una array
50                 StringTokenizer st = new StringTokenizer(mensajeServidor,",");
51
52                 //Variables
53                 int i=0; // Contador
54                 String[] datos=new String[18]; //Array que almacenara la cadena
55
```

En el bucle ya se pasa la cadena a un array para así poder manejar los datos de forma más organizada. Se establece conexión con la base de datos y se crea la variable "**aux_control**" de control que se inicializa a -1 para saber si la matrícula existe.

```
59              while (st.hasMoreTokens())
60              {
61
62              String str=st.nextToken();
63              datos[i]=str;
64
65              //System.out.println("i: "+i+ " datos: "+datos[i]);
66              i++;
67              }
68
69              //Conectamos con la base de datos
70              ConexionMySQL mysql9 = new ConexionMySQL();
71              java.sql.Connection cn9= mysql9.Conectar();
72              Statement sql9=null; //todo
73
74              //Variables para recogida de los datos de sql
75              ResultSet rs9=null; //listar todo
76              int aux_control = -1; // Variable de control para ver si existe o no en la bbdd la matricula
77              if(cn9!=null){
```

Con la conexión ya establecida con la base de datos, se crea la consulta a esta para posteriormente en el bucle, recibir el código de la matrícula que se estaba buscando y es almacenado el código en "**aux_control**".

```
78
79      try {
80          //System.out.println("Conexión establecida");
81          sql9=(Statement) cn9.createStatement();
82          rs9=sql9.executeQuery("SELECT * FROM matricula WHERE matricula='"+datos[3]+"'");
83          //System.out.println("CONSULTA EJECUTADA");
84
85          boolean r9=rs9.next();
86          while (r9) {
87              //System.out.println(rs9.getInt("codigo") + " - " + rs9.getString("matricula"));
88              aux_control = rs9.getInt("codigo");
89              r9=rs9.next();
90          }
```

Si se recoge el código de la matrícula, significa que existe, por lo tanto no tiene que insertar la matrícula. Si no se recoge, entonces la insertará en la base de datos, se le generará un código a esa matrícula y se cerrará la conexión de la base de datos.

```
92      if(aux_control>0)
93      {
94
95      }else
96      {
97
98              try {
99
100                 Statement statement3=(Statement) cn9.createStatement();
101                 //System.out.println("Conexión establecida1");
102                 //System.out.println("insert into matricula (matricula) "+ "values('"+datos[3]+"')");
103
104                  statement3.execute("insert into matricula (matricula) "+ "values('"+datos[3]+"')");
105                 //System.out.println("CONSULTA EJECUTADA2");
106                  statement3.close();
107                 cn9.close();
108                 //System.out.println("CERRADA LA CONEXION");
109             } catch (SQLException e) {
110                 System.out.println(e+"ERROR AL EJECUTAR LA SENTENCIA SQL");
111             }
112
113     }
114
115         cn9.close();
116         //System.out.println("CERRADA LA CONEXION");
117     } catch (SQLException e) {
118         System.out.println(e+"ERROR AL EJECUTAR LA SENTENCIA SQL");
119     }
120
121 }
```

Se prepara la conexión a la base de datos y se busca nuevamente el código de la matrícula ya teniendo seguro que sí existirá en la base de datos.

```
123  ConexionMySQL mysql = new ConexionMySQL();
124
125  java.sql.Connection cn= mysql.Conectar();
126  Statement sql=null; //todo
127
128  //Variables para recogida de los datos de sql
129  ResultSet rs=null; //listar todo
130
131  if(cn!=null){
132      try {
133          //System.out.println("Conexión establecida");
134          sql=(Statement) cn.createStatement();
135          rs=sql.executeQuery("SELECT * FROM matricula WHERE matricula='"+datos[3]+"'");
136          //System.out.println("CONSULTA EJECUTADA");
```

Recuperamos el código que se le asigna a la matrícula y lo cambiamos en el array de los datos que nos envió el socket.

```
137
138      // Recuperamos el codigo que se le asigna a la matricula y lo cambiamos en la arry
139      boolean r=rs.next();
140      while (r) {
141          //System.out.println(rs.getInt("codigo") +  " - " + rs.getString("matricula"));
142          datos[3] = String.valueOf(rs.getInt("codigo"));
143          r=rs.next();
144      }
145
```

Se cierra la conexión y se crea nuevamente la conexión.

```
157      cn.close();
158      //System.out.println("CERRADA LA CONEXION");
159  } catch (SQLException e) {
160      System.out.println(e+"ERROR AL EJECUTAR LA SENTENCIA SQL");
161  }
162
163  java.sql.Connection cn2= mysql.Conectar();
```

Se insertan los datos en la tabla "**datos**" para almacenarlos en un futuro uso y cerramos la conexión a la base de datos.

```
165      try {
166
167          Statement statement2=(Statement) cn2.createStatement();
168
169          //System.out.println("Conexión establecida");
170
171
172          // Insertamos la matricula en la nueva tabla con la id en vez de la matricula original
173           statement2.execute("insert into datos (id_registro,numero_reconocedor_lpr,numero_camara_lpr,"
174                  + "matricula,fecha,hora,numero_de_mascara,velocidad,direccion,coeficiente_reconocimiento) "
175                  + "values("+datos[0]+","+datos[1]+","+datos[2]+",'"+datos[3]+"','"+datos[4]+"','"
176                  +datos[5]+"','"+datos[6]+","+datos[7]+",'"+datos[8]+"','"+datos[9]+")");
177          //System.out.println("CONSULTA EJECUTADA");
178
179           statement2.close();
180          cn2.close();
181
182          //System.out.println("CERRADA LA CONEXION");
183      } catch (SQLException e) {
184          System.out.println(e+"ERROR AL EJECUTAR LA SENTENCIA SQL");
185      }
186
187      try{
188          cn.close();
189      }catch(SQLException ex){
190
191          System.out.println("Error al desconectar "+ex);
192      }
193      }
194      }
195      System.out.println("Fin de la conexión");
```

```
196             ss.close();//Se finaliza la conexión con el cliente
197         }
198         catch (Exception e)
199         {
200             System.out.println(e.getMessage());
201         }
202     }
203 }
204
```

4.1.6.-Código fuente Conexion.java

Lo primero que hacemos es cargar las librerías que posteriormente se utilizarán. Se definen las variables de:

- **Puertos.**
- **HOST.**
- **mensajeServidor.**
- **ss.**
- **cs.**
- **salidaServidor.**
- **salidaCliente.**

Éstas son utilizadas para conectar con el socket y recibir información de éste. También aquí se crea el constructor del socket.

```
1  package sockets;
2
4  *  Author: Carlos Bosch Tur
13
14 import java.io.DataOutputStream;
18
19 public class Conexion
20 {
21     //Variables del socket
22     private final int PUERTO = 1234; //Puerto para la conexión
23     private final String HOST = "localhost"; //Host para la conexión
24     protected String mensajeServidor; //Mensajes entrantes (recibidos) en el servidor
25     protected ServerSocket ss; //Socket del servidor
26     protected Socket cs; //Socket del cliente
27     protected DataOutputStream salidaServidor, salidaCliente; //Flujo de datos de salida
28
29     public Conexion(String tipo) throws IOException //Constructor
30     {
31         if(tipo.equalsIgnoreCase("servidor"))
32         {
33             ss = new ServerSocket(PUERTO);//Se crea el socket para el servidor en puerto 1234
34             cs = new Socket(); //Socket para el cliente
35         }
36         else
37         {
38             cs = new Socket(HOST, PUERTO); //Socket para el cliente en localhost en puerto 1234
39         }
40     }
```

4.1.7.-Código fuente ConexionMySQL.java

Aquí se definen las variables para conectar con la base de datos que son:

- **bd**
- **url**
- **user**
- **pass**

Se crea el constructor de la conexión a la base de datos para acceder a ésta desde cualquier parte del código.

```
1  package sockets;
2
4  *  Author: Carlos Bosch Tur
13
14 import java.sql.*;
16
17 public class ConexionMySQL {
18
19     //Varibles de acceso a la bbdd
20     public String db = "tfg";
21     public String url = "jdbc:mysql://localhost/"+db;
22     public String user = "carlos";
23     public String pass = "123";
24
25    public Connection Conectar(){
26
27        Connection link = null;
28
29        try{
30
31            Class.forName("org.gjt.mm.mysql.Driver");
32
33            link = DriverManager.getConnection(this.url, this.user, this.pass);
34
35        }catch(Exception ex){
36
37            JOptionPane.showMessageDialog(null, ex);
38
39        }
40        return link;
41    }
42 }
```

4.1.8.-Código fuente MainServidor.java

Aquí el servidor es lanzado para mantenerse en espera de que algún cliente conecte para recibir la información que le suministre.

```
1  package sockets;
2
4  *  Author: Carlos Bosch Tur
13
14 import java.io.IOException;
16
17 //Clase principal que hará uso del servidor
18 public class MainServidor
19 {
20     public static void main(String[] args) throws IOException
21     {
22         Servidor serv = new Servidor(); //Se crea el servidor
23
24         System.out.println("Iniciando servidor");
25         serv.startServer(); //Se inicia el servidor
26     }
27 }
```

4.1.9.-Base de datos del ayuntamiento

Esta base de datos está compuesta por tres tablas que son “**camaras**” , “**matriculas**” y “**trafico**”. Cada uno almacena cierta información de las cámaras.

La tabla trafico es idéntica a la de matrículas con la particularidad que solo almacena los datos de las últimas 24 horas para que así el mapa que se verá más adelante cargue con mayor fluidez.

	Tabla	Filas	Tipo	Cotejamiento	Tamaño
☐	camaras	8	InnoDB	latin1_spanish_ci	16 KB
☐	matriculas	~5,734,007	InnoDB	latin1_spanish_ci	457.9 MB
☐	trafico	41,651	InnoDB	latin1_spanish_ci	4 MB
	3 tablas	~5,775,666	InnoDB	latin1_spanish_ci	461.9 MB

4.- Visión general de la estructura de la base de datos

Estructura de la tabla “**camaras**”:

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
☐	1	codigo 🔑	int(11)			No	*Ninguna*		AUTO_INCREMENT
☐	2	camara	varchar(255)	latin1_spanish_ci		No	*Ninguna*		
☐	3	coord	varchar(255)	latin1_spanish_ci		No	*Ninguna*		
☐	4	text1	varchar(255)	latin1_spanish_ci		No	*Ninguna*		
☐	5	text2	varchar(255)	latin1_spanish_ci		No	*Ninguna*		

5.- Estructura tabla **camaras**

Estructura de la tabla “**matriculas**”:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	matricula	varchar(10)	latin1_spanish_ci		No	*Ninguna*		
3	fecha	datetime(3)			No	*Ninguna*		
4	hora	time			No	*Ninguna*		
5	direccion	varchar(255)	latin1_spanish_ci		No	*Ninguna*		
6	velocidad	float			No	*Ninguna*		
7	id_sistema	varchar(255)	latin1_spanish_ci		No	*Ninguna*		

6.- Estructura tabla **matriculas**

Estructura de la tabla “**trafico**”:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	matricula	varchar(10)	latin1_spanish_ci		No	*Ninguna*		
3	fecha	datetime(3)			No	*Ninguna*		
4	hora	time			No	*Ninguna*		
5	direccion	varchar(255)	latin1_spanish_ci		No	*Ninguna*		
6	velocidad	float			No	*Ninguna*		
7	id_sistema	varchar(255)	latin1_spanish_ci		No	*Ninguna*		

7.- Estructura tabla **trafico**

4.1.10.-Sql de la tabla

Aquí veremos información general sobre la base de datos y parámetros sobre la configuración, como puede ser la franja horaria de ésta.

```
-- phpMyAdmin SQL Dump
-- version 4.7.0
-- https://www.phpmyadmin.net/
--
-- Servidor: localhost
-- Tiempo de generación: 31-07-2017 a las 18:14:33
-- Versión del servidor: 5.6.34
-- Versión de PHP: 7.1.4

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de datos: `qwx136`
--
```

Aquí se muestra la estructura de las tablas de "**camaras**" y de "**matriculas**".

```
-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `camaras`
--

CREATE TABLE `camaras` (
  `codigo` int(11) NOT NULL,
  `camara` varchar(255) COLLATE latin1_spanish_ci NOT NULL,
  `coord` varchar(255) COLLATE latin1_spanish_ci NOT NULL,
  `text1` varchar(255) COLLATE latin1_spanish_ci NOT NULL,
  `text2` varchar(255) COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `matriculas`
--

CREATE TABLE `matriculas` (
  `id` int(11) NOT NULL,
  `matricula` varchar(10) COLLATE latin1_spanish_ci NOT NULL,
  `fecha` datetime(3) NOT NULL,
  `hora` time NOT NULL,
  `direccion` varchar(255) COLLATE latin1_spanish_ci NOT NULL,
  `velocidad` float NOT NULL,
  `id_sistema` varchar(255) COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;

```

El disparador hace que en la tabla trafico solo se almacenen los datos de las últimas 24 horas para que el mapa de las cámaras cargue más rápido.

```
--
-- Disparadores `matriculas`
--
DELIMITER $$
CREATE TRIGGER `limite` BEFORE INSERT ON `matriculas` FOR EACH ROW DELETE FROM trafico
WHERE fecha = NOW() - INTERVAL 1 DAY
$$
DELIMITER ;
DELIMITER $$
CREATE TRIGGER `nuevalinea` AFTER INSERT ON `matriculas` FOR EACH ROW INSERT INTO trafico
    SELECT * FROM `matriculas` ORDER BY `id` DESC LIMIT 1
$$
DELIMITER ;

-- --------------------------------------------------------
```

Estructura de la tabla de “**trafico**”.

```
--
-- Estructura de tabla para la tabla `trafico`
--

CREATE TABLE `trafico` (
  `id` int(11) NOT NULL,
  `matricula` varchar(10) COLLATE latin1_spanish_ci NOT NULL,
  `fecha` datetime(3) NOT NULL,
  `hora` time NOT NULL,
  `direccion` varchar(255) COLLATE latin1_spanish_ci NOT NULL,
  `velocidad` float NOT NULL,
  `id_sistema` varchar(255) COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;

```

Se asigna las claves primarias de las tablas.

```
--
-- Índices para tablas volcadas
--

--
-- Indices de la tabla `camaras`
--
ALTER TABLE `camaras`
  ADD PRIMARY KEY (`codigo`);

--
-- Indices de la tabla `matriculas`
--
ALTER TABLE `matriculas`
  ADD PRIMARY KEY (`id`);

--
-- Indices de la tabla `trafico`
--
ALTER TABLE `trafico`
  ADD PRIMARY KEY (`id`);

```

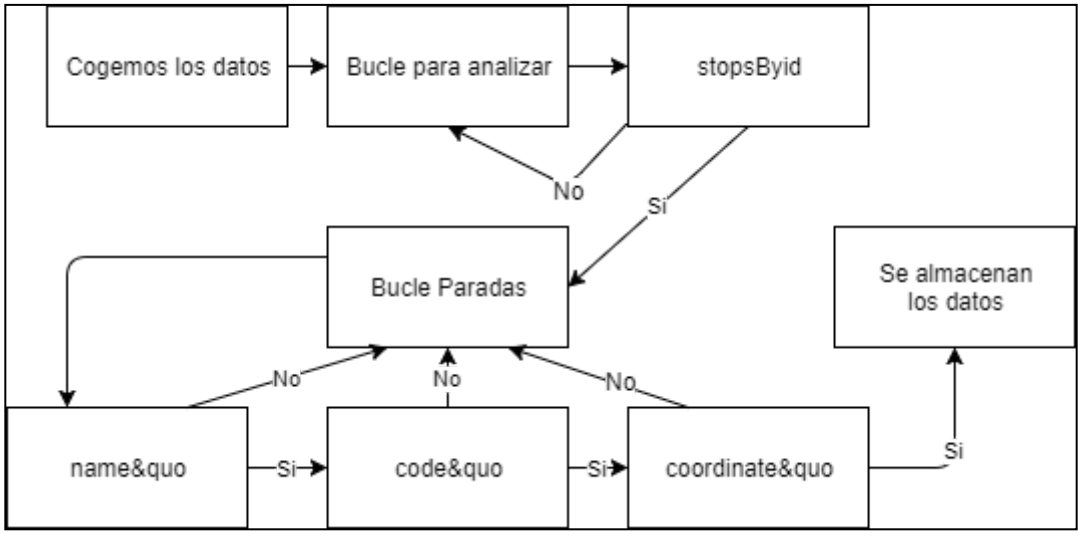
4.2.-Autobuses

Para poder recoger los datos en tiempo real de los autobuses teníamos que hacer que el servidor se descargara el código fuente de la web de los autobuses donde estaban los datos que se necesitaban, los analizara ,filtrara y mostrara en un mapa el resultado.

4.2.1.-Esquema de recogida de datos

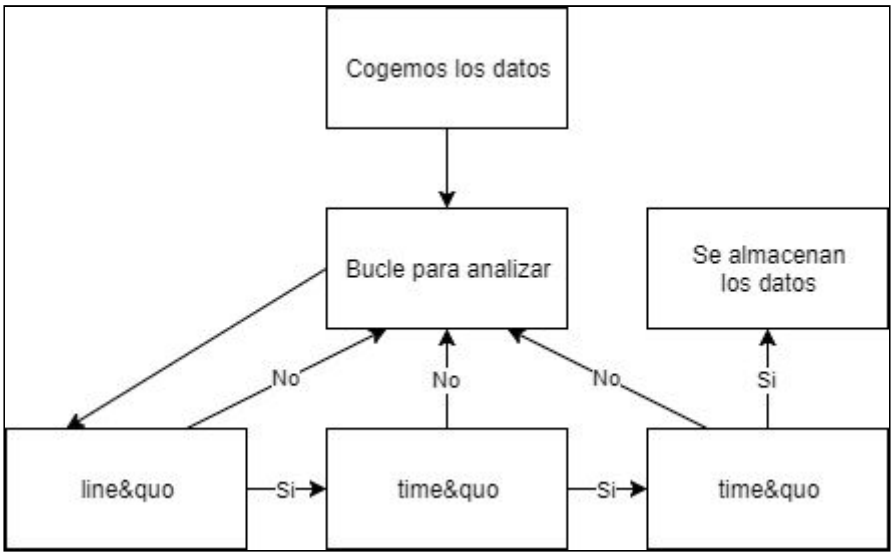
8.- Esquema recogida de datos de los autobuses

4.2.2.-Diagrama de flujo autob.php



9.- Diagrama de flujo de autob.php

4.2.3.-Diagrama de flujo autob2.php



10.- Diagrama de flujo autob2.php

4.2.4.-Código fuente de la recogida de datos

Constara de dos archivos:

- **autob.php:** Contiene el script que se utiliza para acceder a la web de los autobuses y recoger todas las paradas de autobuses.
- **autob2.php:** Se encarga de mostrar la información relacionada de la parada solicitada.

4.2.4.1.-autob.php

Se define la url desde donde se accederá para reunir la información que necesitaremos, se carga cada letra en un posición del array y se definen variables de control que serán utilizadas durante la ejecución del script.

```php
<?php
$url = "http://www.alcoi.vectalia.es/";
$url = $url."linea/sant-vicent-centre-batoi-eixample/";
$url = $url."#linea=2";
$data = file_get_contents("$url");
$web =  htmlentities($data, ENT_QUOTES);
$tamano = strlen($web);
$contador = 0;
$control_1 = 0;
$aux_code_parada = "";
```

Se definen los nombres de los array que almacenará el nombre, la coordenada y el código de cada parada de autobuses.

```
12  // Arrays
13
14  $contador_array = 0;
15  $nombre;
16  $cord;
17  $codigo;
18
19  $contador = 0;
```

El bucle empieza a correr descartando todo el texto hasta encontrar la palabra "**stopsById**", y a partir de aquí ya se empieza a guardar información para mostrarla más adelante.

```
21  while($contador!=$tamano)
22  {
23
24  if($web[$contador]=="s")
25    {$contador++;
26   if($web[$contador]=="t")
27     {$contador++;
28    if($web[$contador]=="o")
29      {$contador++;
30     if($web[$contador]=="p")
31       {$contador++;
32      if($web[$contador]=="s")
33        {$contador++;
34       if($web[$contador]=="B")
35         {$contador++;
36        if($web[$contador]=="y")
37          {$contador++;
38         if($web[$contador]=="I")
39           {$contador++;
40          if($web[$contador]=="d")
41            {$contador++;
```

Se descarta todo hasta encontrar la palabra “**name&quo**”.

```
while($control_1==0)
{
if($web[$contador]=="n")
  {$contador++;
 if($web[$contador]=="a")
   {$contador++;
  if($web[$contador]=="m")
    {$contador++;
   if($web[$contador]=="e")
     {$contador++;
    if($web[$contador]=="&")
      {$contador++;
     if($web[$contador]=="q")
       {$contador++;
      if($web[$contador]=="u")
        {$contador++;
       if($web[$contador]=="o")
```

Cuando se encuentra la palabra “**name&quo**” se posiciona el array en la primera letra del nombre de la parada y se crea la variable “**aux_code_parada**”, que servirá de ayuda para más tarde reunir la información.

```
{
//echo "Contador: ".$contador. " Texto: ".$web[$contador]."<br/>";
$contador = $contador +10;
$aux_code_parada = "";
//echo "Nombre: ";
$contador_array++;
```

Se almacena el nombre de la parada en el array nombre.

```
while($web[$contador]!="&")
{
$aux_code_parada =  $aux_code_parada . $web[$contador];
//echo "encuentra code";
//echo "Contador: ".$contador. " Texto: ".$web[$contador]."<br/>";
//echo $web[$contador];
$nombre[$contador_array] = $nombre[$contador_array].$web[$contador];
$contador++;
}
```

Se descarta todo hasta encontrar la palabra “**code&quo**”.

```
84
85  if($web[$contador]=="c")
86    {$contador++;
87   if($web[$contador]=="o")
88     {$contador++;
89    if($web[$contador]=="d")
90      {$contador++;
91     if($web[$contador]=="e")
92       {$contador++;
93      if($web[$contador]=="&")
94        {$contador++;
95       if($web[$contador]=="q")
96         {$contador++;
97        if($web[$contador]=="u")
98          {$contador++;
99         if($web[$contador]=="o")
```

Cuando se encuentra la palabra “**code&quo**” se posiciona el array en la primera letra del código de la parada y se crea la variable “**aux_code_parada**”, que servirá de ayuda para más tarde reunir la información.

```
102  $contador = $contador +10;
103  $aux_code_parada = "";
```

Se almacena el código de la parada en el array código.

```
106 while($web[$contador]!="&")
107 {
108 $aux_code_parada =  $aux_code_parada . $web[$contador];
109 //echo "encuentra code";
110 //echo "Contador: ".$contador. " Texto: ".$web[$contador]."<br/>";
111 //echo $web[$contador];
112 $codigo[$contador_array] = $codigo[$contador_array].$web[$contador];
113 $contador++;
114 }
```

Se descarta todo hasta encontrar la palabra “**coordinates&quo**”.

```
124  if($web[$contador]=="c")
125    {$contador++;
126   if($web[$contador]=="o")
127     {$contador++;
128    if($web[$contador]=="o")
129      {$contador++;
130     if($web[$contador]=="r")
131       {$contador++;
132      if($web[$contador]=="d")
133        {$contador++;
134       if($web[$contador]=="i")
135         {$contador++;
136        if($web[$contador]=="n")
137          {$contador++;
138         if($web[$contador]=="a")
139           {$contador++;
140          if($web[$contador]=="t")
141            {$contador++;
142           if($web[$contador]=="e")
143             {$contador++;
144            if($web[$contador]=="s")
145              {$contador++;
146             if($web[$contador]=="&")
147               {$contador++;
148              if($web[$contador]=="q")
149                {$contador++;
150               if($web[$contador]=="u")
151                 {$contador++;
152                if($web[$contador]=="o")
153                  {
```

Cuando se encuentra la palabra “**coordinates&quo**” se posiciona el array en la primera letra de las coordenadas de la parada y se crea la variable “**aux_code_parada**”, que servirá de ayuda para más tarde reunir la información.

```
155  $contador = $contador +10;
156  $aux_code_parada = "";
```

Se almacenan las coordenadas de la parada en el array cord.

```
159  while($web[$contador]!="&")
160  {
161  $aux_code_parada =  $aux_code_parada . $web[$contador];
162  //echo "encuentra code";
163  //echo "Contador: ".$contador. " Texto: ".$web[$contador]."<br/>";
164  //echo $web[$contador];
165  $cord[$contador_array] = $cord[$contador_array].$web[$contador];
166  $contador++;
167  }
```

4.2.4.2.-autob2.php

Recogemos el código que se envía a través de la url del navegador y accedemos a la web de donde se saca la información con el código que antes hemos recogido para sacar los datos de la parada elegida, una vez hecho se pasa a un array y se empieza analizar.

```php
<?php
$aux_code_parada = $_GET['codigo'];
$url_parada = "http://www.alcoi.vectalia.es/ajax/";
$url_parada = $url_parada."microsite/";
$url_parada = $url_parada."isae-estimate-by-stop?lang=es&__internal__=1";
$url_parada = $url_parada."&code=$aux_code_parada";
$data_parada = file_get_contents("$url_parada");
$web_parada =  htmlentities($data_parada, ENT_QUOTES);
$tamano_parada = strlen($web_parada);
$contador_parada = 0;
$nojayautobus = 0;
```

Se descarta todo hasta llegar a “**line&quo**”.

```php

while($contador_parada!=$tamano_parada)
{
if($web_parada[$contador_parada]=="l")
    {$contador_parada++;
  if($web_parada[$contador_parada]=="i")
    {$contador_parada++;
   if($web_parada[$contador_parada]=="n")
     {$contador_parada++;
    if($web_parada[$contador_parada]=="e")
      {$contador_parada++;
      if($web_parada[$contador_parada]=="&")
        {$contador_parada++;
       if($web_parada[$contador_parada]=="q")
         {$contador_parada++;
        if($web_parada[$contador_parada]=="u")
          {$contador_parada++;
         if($web_parada[$contador_parada]=="o")
           {
```

Se posiciona el array en la posición correcta y se muestra que línea está por llegar.

```php
            $contador_parada = $contador_parada +10;
            echo "</br>Linea: ".$web_parada[$contador_parada]."</br>";
            $aux_control_dos = 0;
```

Se descarta todo hasta llegar a “**time&quo**”.

```
while($aux_control_dos!=2)
{
if($web_parada[$contador_parada]=="t")
  {$contador_parada++;
 if($web_parada[$contador_parada]=="i")
   {$contador_parada++;
  if($web_parada[$contador_parada]=="m")
    {$contador_parada++;
   if($web_parada[$contador_parada]=="e")
     {$contador_parada++;
    if($web_parada[$contador_parada]=="&")
      {$contador_parada++;
     if($web_parada[$contador_parada]=="q")
       {$contador_parada++;
      if($web_parada[$contador_parada]=="u")
        {$contador_parada++;
       if($web_parada[$contador_parada]=="o")
```

Se posiciona el array en la posición correcta y se prepara para mostrar cuánto tiempo falta para que llegue el siguiente autobús de esa línea.

```
{$contador_parada = $contador_parada +4;
$aux_control_dos++;
echo "</br>Tiempo de espera: ";
```

Se muestra el tiempo que falta para que llegue el siguiente autobús y se actualiza la variable de control de autobuses para que el script sepa si pasa algún autobús.

```
while($web_parada[$contador_parada]!=",")
{
echo $web_parada[$contador_parada];
$contador_parada++;
}
echo " minutos";
$nojayautobus = 1;
```

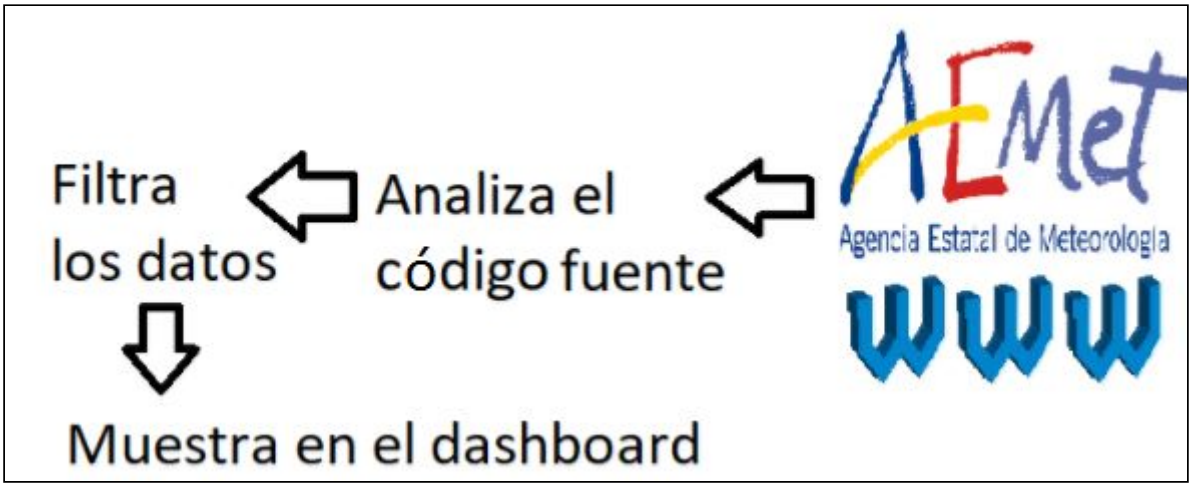
Si no hubiera pasado ningún autobús, con la siguiente comprobación se mostrará un mensaje avisando al usuario que no pasara ningún autobús.

```
if($nojayautobus == 0)
{
    echo "No hay autobuses disponibles para esta parada por el momento.";
}
?>
```

4.3.-Datos Meteorológicos

Para poder recoger los datos en tiempo real de los datos meteorológicos teníamos que hacer que el servidor se descargara el xml de la web de aemet donde estaban los datos que necesitábamos, los analizara ,filtrara y mostrara en un mapa el resultado.

4.3.1-Esquema de recogida de datos



11.- Esquema de recogida de datos de aemet

4.3.2.-Código fuente de la recogida de datos

4.3.2.1.-tiempo.php

Lo primero que se hace es cargar el archivo xml y entrar en un bucle donde se desglosan todas sus partes.

```php
<?php
// muestra lo del nodo origen
foreach ($xml->origen  as $elemento)
{
echo "Campo: " .htmlentities($elemento->productor). "<br>";
echo "Campo: " .htmlentities($elemento->web). "<br>";
echo "Campo: " .htmlentities($elemento->enlace). "<br>";
echo "Campo: " .htmlentities($elemento->language). "<br>";
echo "Campo: " .htmlentities($elemento->copyright). "<br>";
echo "Campo: " .htmlentities($elemento->nota_legal). "<br>";
}
echo "Campo: " .htmlentities($xml->elaborado). "<br>";
echo "Campo: " .htmlentities($xml->nombre). "<br>";
echo "Campo: " .htmlentities($xml->provincia). "<br>";
```

Muestra el nodo “**prob_precipitacion”** al completo.

```
$contador_aux = 0;
while($contador_aux!=$aux_control_de_los_bucles)
{
foreach($xml->prediccion->dia[$bucle_principal]->
prob_precipitacion[$contador_aux]->attributes() as $a => $b)
{
echo "Campo: ".$b."<br>";
echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
prob_precipitacion[$contador_aux]."<br>";
}
$contador_aux++;
}
```

Muestra el nodo “**cota_nieve_prov**” al completo.

```
$contador_aux = 0;
while($contador_aux!=$aux_control_de_los_bucles)
{
foreach($xml->prediccion->dia[$bucle_principal]->
cota_nieve_prov[$contador_aux]->attributes() as $a => $b)
{
echo "Campo: ".$b."<br>";
echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
cota_nieve_prov[$contador_aux]."<br>";
}
$contador_aux++;
}
```

Muestra el nodo “**estado_cielo**” al completo.

```
$contador_aux = 0;
while($contador_aux!=$aux_control_de_los_bucles)
{
foreach($xml->prediccion->dia[$bucle_principal]->
estado_cielo[$contador_aux]->attributes() as $a => $b)
{
echo "Campo: ".$b."<br>";
echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
estado_cielo[$contador_aux]."<br>";
}
$contador_aux++;
}
```

Muestra el nodo “**viento**” al completo.

```
$contador_aux = 0;
while($contador_aux!=$aux_control_de_los_bucles)
{
foreach($xml->prediccion->dia[$bucle_principal]->
viento[$contador_aux]->attributes() as $a => $b)
{
echo "Campo: ".$b."<br>";
echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
viento[$contador_aux]->direccion."<br>";
echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
viento[$contador_aux]->velocidad."<br>";
}
$contador_aux++;
}
```

Muestra el nodo “**racha_max**” al completo.

```
$contador_aux = 0;
while($contador_aux!=$aux_control_de_los_bucles)
{
foreach($xml->prediccion->dia[$bucle_principal]->
racha_max[$contador_aux]->attributes() as $a => $b)
{
echo "Campo: ".$b."<br>";
echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
racha_max[$contador_aux]."<br>";
}
$contador_aux++;
}
```

Muestra el nodo “**temperatura**” al completo.

```
echo "muestra el nodo temperatura completo</br>";
echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
temperatura->maxima."<br>";
echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
temperatura->minima."<br>";
$contador_aux = 0;
if($aux_control_de_los_bucles==6)
{
while($contador_aux!=4)
{
foreach($xml->prediccion->dia[$bucle_principal]->
temperatura->dato[$contador_aux]->attributes() as $a => $b)
{
echo "Campo: ".$b."<br>";
echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
temperatura->dato[$contador_aux]."<br>";
}
$contador_aux++;
}
}
```

Muestra el nodo “**sens_termica**” al completo.

```
120  echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
121  sens_termica->maxima."<br>";
122  echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
123  sens_termica->minima."<br>";
124  $contador_aux = 0;
125  if($aux_control_de_los_bucles==6)
126  {
127  while($contador_aux!=4)
128  {
129  foreach($xml->prediccion->dia[$bucle_principal]->
130  sens_termica->dato[$contador_aux]->attributes() as $a => $b)
131  {
132  echo "Campo: ".$b."<br>";
133  echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
134  sens_termica->dato[$contador_aux]."<br>";
135  }
136  $contador_aux++;
137  }
138  }
```

Muestra el nodo “**humedad_relativa**” al completo.

```
140  echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
141  humedad_relativa->maxima."<br>";
142  echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
143  humedad_relativa->minima."<br>";
144  $contador_aux = 0;
145  if($aux_control_de_los_bucles==6)
146  {
147  while($contador_aux!=4)
148  {
149  foreach($xml->prediccion->dia[$bucle_principal]->
150  humedad_relativa->dato[$contador_aux]->attributes() as $a => $b)
151  {
152  echo "Campo: ".$b."<br>";
153  echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
154  humedad_relativa->dato[$contador_aux]."<br>";
155  }
156  $contador_aux++;
157  }
158  }
159  echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->uv_max."";
160  $bucle_principal++;
161  }
162  echo "numero de nodo ".$bucle_principal."</br>";
163  foreach($xml->prediccion->dia[$bucle_principal]->
164  attributes() as $a => $b)
165  {
166  echo "Campo: ".$b."<br>";
167  }
```

Muestra el nodo “**prob_precipitacion**” al completo.

```
169  $contador_aux = 0;
170  while($contador_aux!=3)
171  {
172  foreach($xml->prediccion->dia[$bucle_principal]->
173  prob_precipitacion[$contador_aux]->attributes() as $a => $b)
174  {
175  echo "Campo: ".$b."<br>";
176  echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
177  prob_precipitacion[$contador_aux]."<br>";
178  }
179  $contador_aux++;
180  }
```

Muestra el nodo “**cota_nieve_prov**” al completo.

```
183  $contador_aux = 0;
184  while($contador_aux!=3)
185  {
186  foreach($xml->prediccion->dia[$bucle_principal]->
187  cota_nieve_prov[$contador_aux]->attributes() as $a => $b)
188  {
189  echo "Campo: ".$b."<br>";
190  echo "Campo: ".$xml->prediccion->dia[$bucle_principal]->
191  cota_nieve_prov[$contador_aux]."<br>";
192  }
193  $contador_aux++;
194  }
     ?>
```

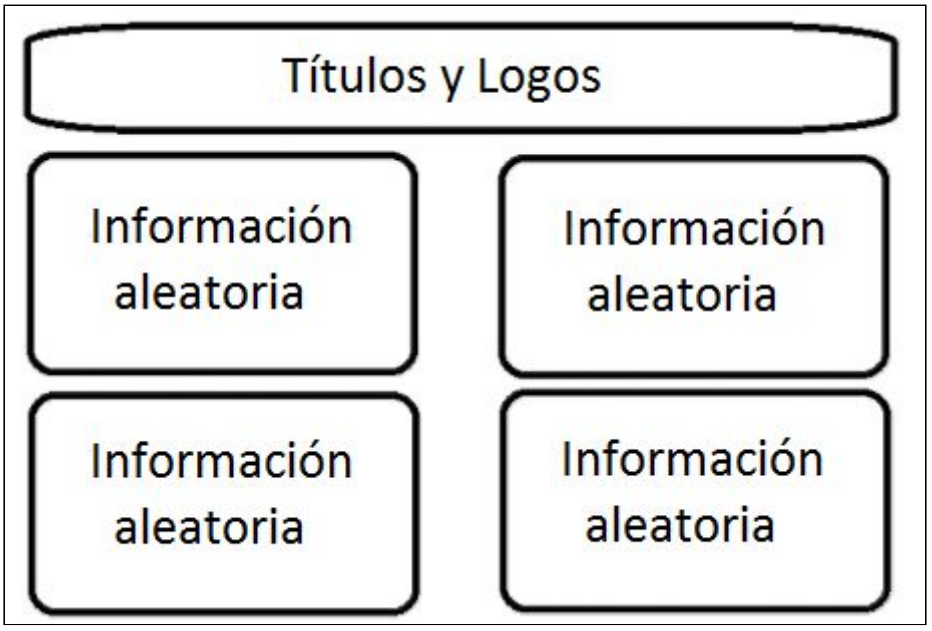
5.-Estructura principal

Con los datos recogidos y filtrados podemos empezar a montar una página web donde acercar al usuario esta información de forma práctica y lógica para su uso diario.

5.1.-Idea general

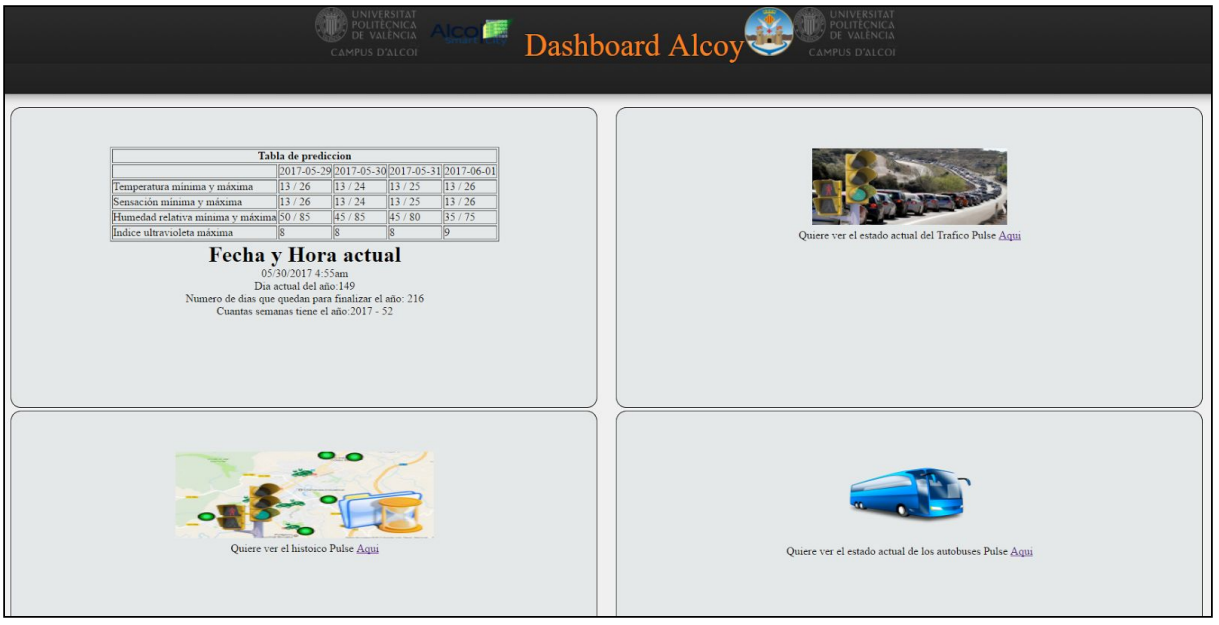
Pensamos en un sistema de una barra fija superior donde colgar el nombre del proyecto y los logos de las empresas implicadas y debajo de ésta un panel de cuatro bloques que saldrán de forma aleatoria cada vez que el usuario actualice.

5.1.1.-Imagen



12.- Idea general del inicio del proyecto

5.1.2.-Vista previa



13.- Vista final de la página web

5.1.3.-Código fuente index.php

En las primeras líneas de código se define el lenguaje de la web en español y algunas de las características de la web, como son los parámetros de comportamiento de los robots a la hora de indexar la web, palabras claves y el autor entre otras.

```
<!DOCTYPE html>
<html lang="es">
<head>
<!-- librerias -->
<meta name="autor" content="Carlos Bosch Tur" />
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<meta name="copyright" content="#" />
<meta name="robots" content="index, follow" />
<meta name="keywords" content="dashboard, alcoy" />
<meta name="description" content="Dashboard Alcoy" />
<!--<link rel="icon" type="image/png" href="imagenes/iconoweb.png" />-->
<link href="estilo/estilo.css" rel="stylesheet" type="text/css"><!-- hoja de estilo -->
<!-- Scripts -->

<!-- fin librerias -->
<title>Dashboard Alcoy</title><!-- titulo -->

</head>
```

Se define un array con las cuatro secciones que forman la página principal y un número aleatorio que marcará desde donde empieza el array.

```
<?php
$nombres = array("buss.php","tiempo.php","Trafico.php","historico.php");
$aleatorio = rand(0, 3);
?>
```

Podemos ver ahora la forma en que los divs crean el menú central, los logos y la distribución de este.

```
<body>
    <a name="arriba"></a>
    <div><!-- contenedor web global -->

        <div class="aux_div"><!-- Parte superior -->
            <div class="separador_sup_izq"> </div>
            <div class="central_superior"><div class="titulo">
                <img src="imagenes/logo_uni_b.png" width="150px" height="70px"/>
                <img src="imagenes/smart.png" width="150px" height="70px"/>
                Dashboard Alcoy<img src="imagenes/alcoy_logo.png" width="70px" height="70px"/>
                <img src="imagenes/logo_uni_b.png" width="150px" height="70px"/></div></div>
            <div class="separador_sup_der"> </div>
        </div><!-- fin Parte superior -->
```

Este trozo de código genera la separación del menú superior de los bloques dinámicos.

```
<div class="aux_div_dos"><!-- Parte superior segunda barra -->
    <div class="separador_cent_izq"> </div>
    <div class="central_medio"> </div
    <div class="separador_cent_der"> </div>
</div><!-- fin Parte superior segunda barra -->

<div class="separador_blanco"> </div><!-- degradado de negro a color web -->
```

En este trozo de código podemos observar como se monta la parte dinámica de la web y cómo se trata el número aleatorio para montar las secciones de la web.

```
        <div class="aux_div_tres"><!-- cuerpo de la web -->
            <div class="separador_noticias_izq"> </div>
            <div class="central_noticias" >

            <div class="arriba_izq"><?php include $nombres[$aleatorio]; $aleatorio++;?></div>
            <div class="arriba_dere"><?php if ($aleatorio > 3 ){$aleatorio = 0; }
            include $nombres[$aleatorio]; $aleatorio++;?></div>
            <div class="bajo_izq"><?php if ($aleatorio > 3 ){$aleatorio = 0; }
            include $nombres[$aleatorio]; $aleatorio++;?></div>
            <div class="bajo_dere"><?php if ($aleatorio > 3 ){$aleatorio = 0; }
            include $nombres[$aleatorio]; $aleatorio++;?></div>

            </div>
            <div class="separador_noticias_der"> </div>
        </div><!-- fin cuerpo de la web -->

    </div><!-- fin contenedor web global -->

</body>
</html>
```

5.2.-Mostrando datos del tráfico

Esta informacion está dividida en dos partes:

- Un mapa donde ver el estado actual del tráfico.
- Un menú donde poder ver el historial del tráfico.

5.2.1.-Estado actual del tráfico

La idea general es poder ver en un mapa en tiempo real el estado actual de las carreteras para saber qué vía sería la más adecuada para tomar. Se mostrará con semáforos de diferentes colores la densidad del tráfico.

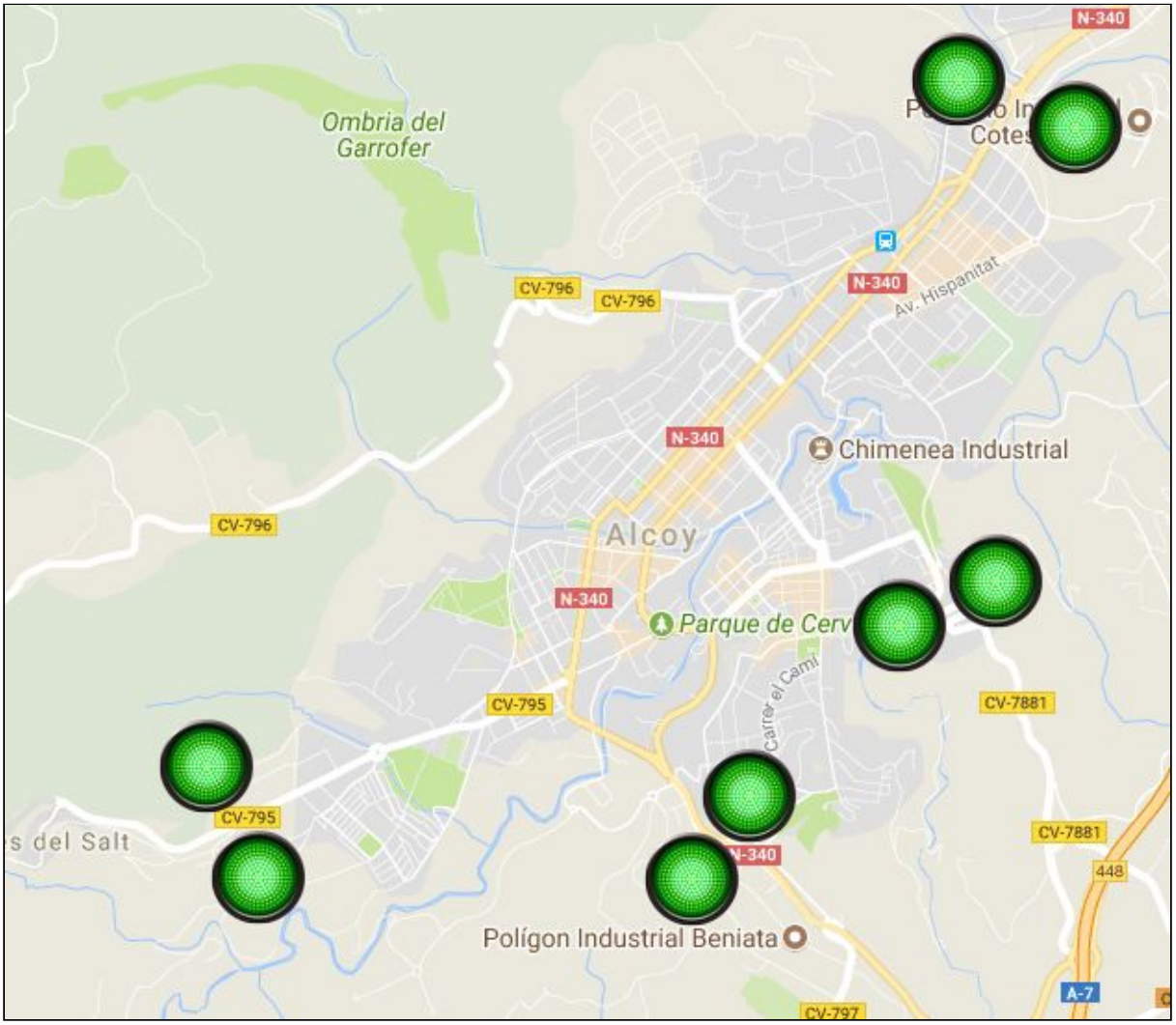
Regla:

 si hay más de 20 coches/min.

 si hay más de 10 coches/min.

 10 o menos coches/min.

5.2.1.1.-Vista previa



14.- Vista de la ubicación actual de las cámaras.

Al pulsar encima de alguno de los semáforos podemos sacar información más detallada.

15.- Vista de una cámara seleccionada

Podemos observar que nos da la media de los últimos coches que han pasado por ese punto en el último minuto. Si pulsamos en +info podremos ver el historial del tráfico de ese día hasta la hora en la que nos encontramos.

16.- Gráfica de la cámara de Cocentaina

5.2.2.-Código fuente trafico.php

En las primeras líneas de código se definen los datos para crear la conexión con la base de datos. Estos son: el nombre de la base de datos, la dirección del servidor, el usuario de la base de datos y la contraseña de éste.

```
<?php
// Conectamos base de datos
// Datos del servidor
$nombrebbdd = "XXXXXXXXXX";
$servidor = "XXXXXXXXX";
$userbbdd = "XXXXXXXX";
$passbbdd = "XXXXXXXXXX";
```

En las siguientes líneas se crea la conexión a la base de datos y se selecciona la base de datos.

```
$conexion = mysql_connect($servidor,$userbbdd,$passbbdd)
or die("Error:".mysql_error());
mysql_select_db($nombrebbdd,$conexion)
or die("Error:".mysql_error());
```

En las siguientes líneas imprimimos un mensaje comentando la regla que actúa en los colores de los semáforos y empezamos escribiendo el código html de la página y definimos ya el título de ésta.

```
echo "MEDIA (10min) de coches que pasan por cada Acceso cada minuto
 pasado, se muestra icono ROJO si mas de 20
 coches/min, AMARILLO si mas de 10 coches/min y VERDE 10 o menos";
echo "<html>";
echo "<head>";
echo "<title>Trafico actual Entradas/Salidas Alcoy</title>;";
```

En el siguiente script vemos como se hace la precarga del mapa y se coloca una imagen para que el usuario vea interacción en la web.

```
echo"<script type=\"text/javascript\">";
echo"var int=self.setInterval(\"refresh()\",123000);";
echo"function refresh()";
echo"{";
echo"location.reload(true);";
echo"}";
echo"</script>";
echo "<script type='text/javascript'
src='http://ajax.googleapis.com/ajax/libs/jquery
/1.4.2/jquery.min.js'></script>";
echo "<script type=\"text/javascript\">";
echo "function efecto(){";
echo "$('#fotocargando').hide();";
echo "$('#map').fadeIn(1500);";
echo "}";
echo "</script>";
```

Aquí vemos como en el body se carga de inicio la imagen de la precarga de la web para que el usuario vea interacción en la web.

```
41  echo "</head>";
42  echo "<body>";
43  echo "<body onload=\"efecto();\">";
44  echo "<div id=\"fotocargando\" style=\"width:100%;
45  height:100%;text-align: center;\">";
46  echo "<img src=\"./loading.gif\">";
47  echo "</div>";
48  echo "<div id=\"map\" style=\"width:100%;height:
49  95%;text-align: center;\"></div>";
50  echo "</body>";
```

En el siguiente fragmento de código recuperamos la información de la tabla "**camaras**", creamos un contador para contar el número de cámaras y en el siguiente bucle se guarda en un array el número de vehículos que pasan por delante de cada cámara.

```
53  $camaras = "SELECT * FROM camaras";
54  $result = mysql_query($camaras,$conexion)
55  or die("Error:".mysql_error());
56
57  $contador_numero_camaras = 1;
58  while ($row = mysql_fetch_array($result))
59          {
60  $consulta_camara[$contador_numero_camaras] =
61  "SELECT count(matricula) as coches FROM trafico
62  WHERE id_sistema = ".$row['camara']." AND fecha
63   between DATE_SUB(now(), interval 10 minute) and now()";
64  $contador_numero_camaras++;
65          }
```

Se crea un nuevo contador para que el bucle recorra el número de cámaras. En el bucle se conecta a la base de datos, se selecciona la base de datos y se saca la media de cada una de las cámaras.

```
67  $contador_obtener_datos_camara = 1;
68  $array_datos_camaras = array();
69  $aux_finfo = "";
70  while($contador_obtener_datos_camara!=$contador_numero_camaras)
71  {
72  $conexion = mysql_connect($servidor,$userbbdd,$passbbdd)
73  or die("Error:".mysql_error());
74  mysql_select_db($nombrebbdd,$conexion)or die("Error:".mysql_error());
75  $aux_result = "result"."$contador_obtener_datos_camara";
76  $aux_result = mysql_query($consulta_camara
77  [$contador_obtener_datos_camara],$conexion)or die("Error:".mysql_error());
78  $aux_finfo = mysql_fetch_array($aux_result, MYSQL_NUM);
79  mysql_free_result($aux_result);
80  mysql_free_result($consulta_camara[$contador_obtener_datos_camara]);
81  mysql_close($conexion);
82  $aux_finfo[0]=$aux_finfo[0]/10;
83  $array_datos_camaras[$contador_obtener_datos_camara] = $aux_finfo[0];
84  $contador_obtener_datos_camara++;
85  }
```

En el siguiente código se crean las coordenadas de la vista inicial al cargar el mapa por primera vez, se recogen los datos de las cámaras y se marca en el mapa las coordenadas de cada cámara.

```
86
87    echo "<script>";
88    echo "function myMap(){";
89    echo "var myCenter = new google.maps.LatLng(38.702438,-0.481251);";
90
91  $camaras = "SELECT * FROM camaras";
92  $conexion = mysql_connect($servidor,$userbbdd,$passbbdd)
93  or die("Error:".mysql_error());
94  mysql_select_db($nombrebbdd,$conexion)or die("Error:".mysql_error());
95  $result = mysql_query($camaras,$conexion)or die("Error:".mysql_error());
96  $contador_coord_camara = 1;
97  while ($row = mysql_fetch_array($result))
98  {
99    $estacion = "estacion" . "$contador_coord_camara";
100   echo "var $estacion = new google.maps.LatLng(".$row['coord'].");";
101   $contador_coord_camara++;
102 }
```

Se crean las opciones de control del mapa al inicializarlo.

```
104   echo "var mapCanvas = document.getElementById(\"map\");";
105   echo "var mapProp ={";
106   echo "center: myCenter,";
107   echo "scrollwheel: false,";
108   echo "zoom: 14,";
109   echo "zoomControl: false,";
110   echo "rotateControl : false,";
111   echo "mapTypeControl: false,";
112   echo "streetViewControl: false,";
113   echo "scaleControl: false,";
114   echo "disableDefaultUI: true,";
115   echo "};";
116   echo "var map = new google.maps.Map(mapCanvas, mapProp);";
117
```

En el siguiente fragmento de código podemos observar que tipo de color de semáforo se dibuja en el mapa dependiendo de la regla de los semáforos anteriormente citada.

```
118  $contador_datos_mapa_camara = 1;
119  while($contador_datos_mapa_camara!=$contador_numero_camaras)
120  {
121
122    $aux_marker = "marker"."$contador_datos_mapa_camara";
123    echo "if($array_datos_camaras[$contador_datos_mapa_camara]>20){";
124    echo "var $aux_marker = new google.maps.Marker({";
125    echo "position:estacion$contador_datos_mapa_camara,";
126    echo "icon: \"semaforo-rojo-opcion-8.png\",";
127    echo "animation: google.maps.Animation.DROP";
128    echo "});";
129    echo "$aux_marker.setMap(map);";
130    echo "}else if($array_datos_camaras[$contador_datos_mapa_camara]>10){";
131    echo "var $aux_marker = new google.maps.Marker({";
132    echo "position:estacion$contador_datos_mapa_camara,";
133    echo "icon: \"semaforo-amarillo-opcion-8.png\",";
134    echo "animation: google.maps.Animation.DROP";
135    echo "});";
136    echo "$aux_marker.setMap(map);";
137    echo "}else{";
138    echo "var $aux_marker = new google.maps.Marker({";
139    echo "position:estacion$contador_datos_mapa_camara,";
140    echo "icon: \"semaforo-verde-opcion-8.png\",";
141    echo "animation: google.maps.Animation.DROP";
142    echo "});";
143    echo "$aux_marker.setMap(map);";
144    echo "}";
145    $contador_datos_mapa_camara++;
146   }
```

Recuperamos los datos de las cámaras y se crean los mensajes de cada cámara con su nombre y con el promedio de los coches que pasan por ésta.

```
148  //preparamos la consulta
149  $camaras = "SELECT * FROM camaras";
150  $result = mysql_query($camaras,$conexion)or die("Error:".mysql_error());
151
152  $contador_mensaje_dos_camaras = 1;
153  while ($row = mysql_fetch_array($result))
154          {
155  echo "var infowindow$contador_mensaje_dos_camaras = new google.maps.InfoWindow({";
156  echo "content: \"
157  ".$row[text1].$array_datos_camaras[$contador_mensaje_dos_camaras].$row[text2]."\"";
158  echo "});";
159  $contador_mensaje_dos_camaras++;
160          }
```

En el siguiente fragmento de código vemos cómo se crea el mensaje emergente al pulsar click encima de cada cámara.

```
161
162  $contador_click_mapa_camara = 1;
163  while($contador_click_mapa_camara!=$contador_numero_camaras)
164  {
165  echo "google.maps.event.addListener
166  (marker$contador_click_mapa_camara,'click',function() {";
167  echo "infowindow$contador_click_mapa_camara.open
168  (map, marker$contador_click_mapa_camara);";
169  echo "});";
170  $contador_click_mapa_camara++;
171  }
```

Aquí declaramos la key de la api del mapa de google con el que funciona todo el script en la página web.

```
172  echo "}";
173  echo "</script>";
174  echo "<script src=\"https://maps.googleapis.com/maps/
175  api/js?key=AIzaSyAwItg2TfpiKBdHdrBKUKTRiLYjgUy3sGs&callback=myMap\"></script>";
176  echo "</html>";
177  mysql_close($conexion);
178  ?>
```

5.2.3.-Código fuente trafico2.php

Preparamos la conexión a la base de datos, un contador de las horas y creamos las variables de la fecha para poder actuar con ellas más adelante.

```
$conexion = mysqli_connect($servidor,$userbbdd,$passbbdd,$nombrebbdd)
or die("Error:".mysqli_error());

//variables
$aux_contador_horas=0;
$ano = date("Y");
$mes = date("m");
$dia = date("d");
```

Se prepara la consulta del gráfico seleccionado y posteriormente se realiza la consulta.

```
$SQLDatos = "SELECT t1.hora, t1.Entran, COALESCE(t2.Salida,0) AS Salen
FROM (SELECT HOUR(fecha) as hora, COUNT(matricula) as Entran
FROM matriculas WHERE (id_sistema = 'AF_Acceso_Banyeres')
AND (direccion ='Acercamiento') AND fecha LIKE ('%".$ano."-".$mes."-".$dia."%')
group by HOUR(fecha)) t1 LEFT JOIN (SELECT HOUR(fecha) as hora,
COUNT(matricula) as Salida FROM matriculas WHERE
(id_sistema = 'AF_Acceso_Banyeres') AND (direccion ='Alejamiento')
AND fecha LIKE ('%".$ano."-".$mes."-".$dia."%') group by HOUR(fecha))
 t2 ON t1.hora = t2.hora";

$result = mysqli_query($conexion,$SQLDatos)or die("Error:".mysqli_error());
$datosObtenidos=obtenerDatos($result);
$datosJSON= json_encode($datosObtenidos,JSON_NUMERIC_CHECK);
mysqli_close($conexion);
```

Se obtiene el número de filas, de columnas y el nombre de los campos de las columnas y se empieza a preparar la tabla.

```
function obtenerDatos($result){
      $numFilas = mysqli_num_rows($result);   //obtenemos número de filas
      $numCols = mysqli_num_fields($result);  //obtenemos número de columnas
      $columnas = array();
      //preparamos las cabeceras de tabla de datos
      while ($finfo = mysqli_fetch_field($result)) {
          //obtenemos diferente informacion del resultado de la consulta
          $columnas = array_merge($columnas, array($finfo->name));
          //obtenemos el nombre del campo de cada columna
      }
      $datos[0] =$columnas;  //pasa el nombre de todas la columnas
    //  a la primera fila de los datos
      $i=1;
```

Se prepara la tabla y se devuelve el array completo con los datos y las cabeceras.

```
  while ($filas = $result->fetch_array(MYSQLI_NUM)) {
      $columnas = array();
      $j=0;
      while ($j<$numCols)
      {
        $columnas = array_merge($columnas, array($filas[$j]));
        $j++;
      }
      $datos[$i] = $columnas;
      //pasamos todas la columnas a una fila de datos
      $i++;
  }
  mysqli_free_result($result);  //liberamos el resultset
  //print_r ($datos);
  return $datos; //devolvemos el array completo con cabecera y datos
}
```

Se cargan las librerías externas que se necesitan para dibujar la tabla en la página web y se cargan los datos que anteriormente se solicitaron a la base de datos.

```
echo "<html>";
echo "<head>";
echo "<meta http-equiv=\"Content-Type\"
content=\"text/html; charset=utf-8\" />";
echo "<script type=\"text/javascript\"
src=\"https://www.gstatic.com/charts/loader.js\"></script>";
echo "<script src=\"https://ajax.googleapis.com/ajax
/libs/jquery/2.2.0/jquery.min.js\"></script>";
echo "<script type=\"text/javascript\">";
echo "google.charts.load('current', {packages: ['corechart']});";
echo "google.charts.setOnLoadCallback(drawBasic);";
echo "";
echo "function drawBasic() {";
echo "";
echo "var respuesta=$datosJSON;";
echo "var data = new google.visualization.arrayToDataTable(respuesta);";
echo "";
```

Configuramos diversas opciones de la tabla como el título, mínimos y máximos de la tabla.

```
echo "var options = {";
echo "title: 'Coches Acceso Banyeres / Horas del dia',";
echo "hAxis: {";
echo "title: 'Horas del Día',";
echo "format: '',";
echo "viewWindow: {";
echo "min: [00,, ],";
echo "max: [23,, ]";
echo "}";
echo "},";
echo "vAxis: {";
echo "title: 'Numero de coches'";
```

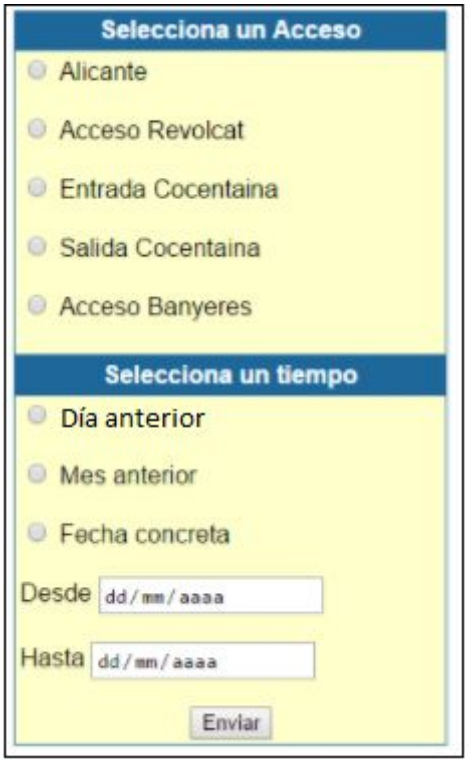
Se dibuja la tabla en la página web para que se pueda visualizar.

```
        echo "title: 'Numero de coches'";
        echo "}";
        echo "};";

        echo "var chart = new google.visualization.ColumnChart
        (document.getElementById('chart_div'));";
        echo "";
        echo "chart.draw(data, options);";
        echo "}";
        echo "</script>";
        echo "</head>";
        echo "<body>";
        echo "<div id=\"chart_div\"></div>";
        echo "</body>";
        echo "</html>";


    ?>
```

5.3.-Estadísticas sobre el tráfico

En este apartado se ofrece la posibilidad al usuario de poder consultar datos históricos sobre la densidad del tráfico en Alcoy.

5.3.1.-Vista previa



17.- Formulario del historial del tráfico.

5.3.2.-Código Fuente historial.php

El siguiente fragmento de código que se mostrará será un ejemplo de "**AF_Entrada_Alicante**" ya que el código es el mismo cambiando la cámara seleccionada.

Lo primero que hace es comprobar que el botón "Enviar" haya sido pulsado y acto seguido comprueba la cámara seleccionada. Ahora comprueba si se pulsó sobre el día de ayer y te redirecciona a la web en cuestión.

```
if (isset($_POST['enviar'])) {
   //Si el checkbox condiciones tiene valor
     if (isset($_POST['v3']) && $_POST['v3'] == 'AF_Entrada_Alicante')

       if (isset($_POST['fecha']) && $_POST['fecha'] == 'dia'){

       header("Location: his_ayer.php?v3=$v3");
?>
<script language="JavaScript">
    top.location.href = "his_ayer.php?v3=$v3";
</script>
```

Este es el código para seleccionar el mes pasado.

```
       if (isset($_POST['fecha']) && $_POST['fecha'] == 'mes'){
      header("Location: his_mes.php?v3=$v3");
?>
<script language="JavaScript">
    top.location.href = "his_mes.php?v3=$v3";
</script>
```

A continuación vemos el código para montar la tabla entre dos fechas concretas.

```
       if (isset($_POST['fecha']) && $_POST['fecha'] == 'fechax'){
           header("Location: his_fecha.php?v1=$v1&v2=$v2&v3=$v3");
?>
<script language="JavaScript">
    top.location.href = "his_fecha.php?v1=$v1&v2=$v2&v3=$v3";
</script>
<?php

       }
   }
```

Si no se pulsan los parámetros en la tabla de forma correcta salta una advertencia.

```
   }
   else
      echo '<div style="color:red" align="center" >
      Debes seleccionar una salida y una fecha.</div>';
}
?>
```

Se empieza a crear la información básica de la web que engloba la tabla.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Historial de Trafico</title>
</head>

<body>
```

Código de la parte superior de la tabla donde se define su aspecto.

```
<table width="280" cellspacing="1" cellpadding="3"
border="0" bgcolor="#1E679A" align="center">
<tr>
   <td><font color="#FFFFFF" face="arial, verdana, helvetica">
   <div style="text-align:center;">
<b>Selecciona un Acceso</b>
</div>
   </font></td>
</tr>
<tr>
   <td bgcolor="#ffffcc">
   <font face="arial, verdana, helvetica">
  <form action="historial.php" method="post">
<input type="radio" name="v3" value="AF_Entrada_Alicante">
 Alicante<br><br>
<input type="radio" name="v3" value="AF_Acceso_Revolcat">
 Acceso Revolcat<br><br>
<input type="radio" name="v3" value="AF_Entrada_Cocentaina">
 Entrada Cocentaina<br><br>
<input type="radio" name="v3" value="AF_Salida_Cocentaina">
 Salida Cocentaina<br><br>
<input type="radio" name="v3" value="AF_Acceso_Banyeres">
 Acceso Banyeres<br><br>
   </font>
   </td>
</tr>
</table>
```

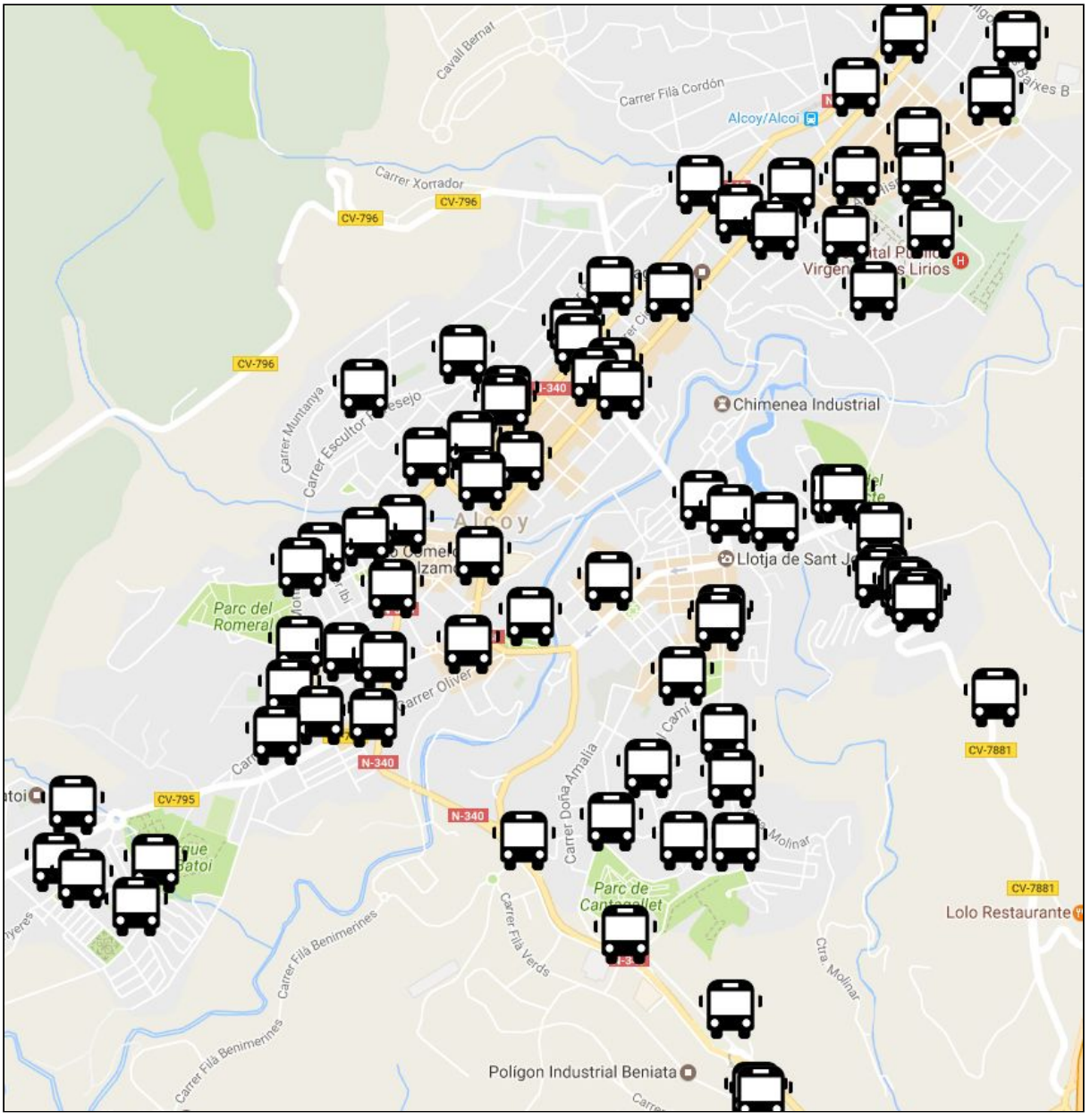
Parte inferior de la tabla donde se define su aspecto.

```
<table width="280" cellspacing="1" cellpadding="3" border="0"
 bgcolor="#1E679A" align="center">
<tr>
   <td><font color="#FFFFFF" face="arial, verdana, helvetica">
    <div style="text-align:center;">
<b>Selecciona un tiempo</b>
</div>
   </font></td>
</tr>
<tr>
   <td bgcolor="#ffffcc">
   <font face="arial, verdana, helvetica">
<input type="radio" name="fecha" value="dia"> Dia anterior<br><br>
<input type="radio" name="fecha" value="mes"> Mes anterior<br><br>
<input type="radio" name="fecha" value="fechax"> Fecha concreta<br><br>
Desde <input type="date" name="v1" size="20"><br><br>
Hasta <input type="date" name="v2" size="20"><br><br>
<div style="text-align:center;">
<input type="submit" name="enviar" value="Enviar"/>
</div>
</form>
 </font>
   </td>
</tr>
</table>
```

5.4.-Parada de autobuses

El usuario dispondrá en un mapa todas las paradas de autobuses para poder ver información de éstas y poder conocer los siguientes autobuses en pasar por allí.

5.4.1.-Vista previa

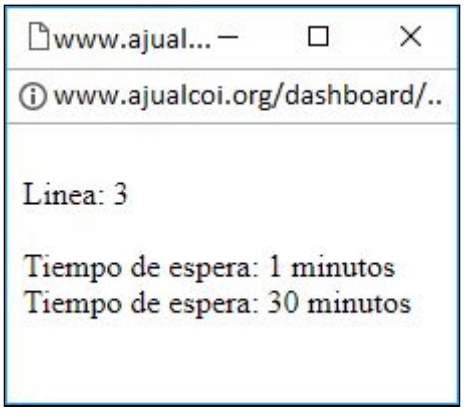


18.- Vista de la ubicación actual de las paradas de autobús.

Al acercar más la vista y pulsar sobre uno de ellos veremos más información de la parada y podremos acceder a un nuevo menú.

19.- Vista de una de las paradas de autobús.

Al pulsar en “Aquí” se abrirá un “**popup**” en el cual veremos el tiempo que le quedan a los autobuses para llegar a esa parada.



20.- Vista de la llegada de un autobús.

5.4.2.-Código Fuente autob.php

Código inicial de la web que dibuja las paradas de autobús encima del mapa de Alcoy. Se establece la posición del mapa con la variable "**myCenter**".

```
echo "<html>";
echo "<style>";
echo "*{ margin: 0px; padding: 0px; }";

echo "</style>";

echo "<body>";
echo "<div id=\"map\" style=\"width:100%;height:100%\"></div>";

echo "<script>";


  echo "function myMap(){";
  echo "var myCenter = new google.maps.LatLng(38.701552, -0.478876);";
```

En esta sección se crea un bucle en el que se generan las coordenadas de todas las paradas de autobuses que hay en Alcoy.

```
  while($contador_mapa_1<=81)
  {
  echo "var estacion$contador_mapa_1 =
  new google.maps.LatLng($cord[$contador_mapa_1]);";
  $contador_mapa_1++;
  }
```

Se generan las opciones que trae el mapa por defecto para que ya esté todo configurado para el usuario.

```
  echo "var mapCanvas = document.getElementById(\"map\");";
  echo "var mapProp ={";
  echo "center: myCenter,";
  //echo "scrollwheel: false,";
  echo "zoom: 14,";
  //echo "zoomControl: false,";
  //echo "rotateControl : false,";
 // echo "mapTypeControl: false,";
  //echo "streetViewControl: false,";
  echo "};";
  echo "var map = new google.maps.Map(mapCanvas, mapProp);";
```

Se crea un bucle con todas las paradas de autobús para poner su icono en el lugar que corresponde encima del mapa.

```
$contador_mapa_1=1;
while($contador_mapa_1<=81)
{
echo "var marker$contador_mapa_1 = new google.maps.Marker({";
echo "position:estacion$contador_mapa_1,";
echo "icon: \"imagenes/autobuss.png\",";
echo "animation: google.maps.Animation.DROP";
echo "});";
echo "marker$contador_mapa_1.setMap(map);";

$contador_mapa_1++;
}
```

Se crea el mensaje que se mostrará en el desplegable de cada parada de autobús.

```
$contador_mapa_1=1;
while($contador_mapa_1<=81)
{
echo "var infowindow$contador_mapa_1 = new google.maps.InfoWindow({";
echo "content: \"<center>$nombre[$contador_mapa_1] <br> Cuando llega el
siguiente autobus <br> Pulsa
<a href='./autob2.php?codigo=$codigo[$contador_mapa_1]'
target='popup' onClick='window.open(this.href, this.target,
width=300,height=400); return false;'>Aqui</a></center>\"";
echo "});";
$contador_mapa_1++;
}
```

Se crea la opción de click en la imagen de cada parada de autobús para que tenga función y aparezca el desplegable.

```
$contador_mapa_1=1;
while($contador_mapa_1<=81)
{
echo "google.maps.event.addListener(marker$contador_mapa_1,'click',function() {";
echo "infowindow$contador_mapa_1.open(map, marker$contador_mapa_1);";
echo "});";
$contador_mapa_1++;
}
```

Código con las clave de la api que nos permite dibujar encima de los mapas de google.

```
echo "}";
echo "</script>";
echo "<script src=\"https://maps.googleapis.com/maps/api/
js?key=AIzaSyAwItg2TfpiKBdHdrBKUKTRiLYjgUy3sGs&callback=myMap\"></script>";
echo "</body>";
echo "</html>";
?>
```

5.4.3.-Código Fuente autob2.php

Aquí recogemos el código que se envía a través de la url del navegador y accedemos a la web, de donde se saca la información con el código que antes hemos recogido para sacar los datos de la parada elegida. Una vez hecho se pasa a un array y se empieza a analizar.

```
<?php
$aux_code_parada = $_GET['codigo'];
$url_parada = "http://www.alcoi.vectalia.es/ajax/";
$url_parada = $url_parada."microsite/";
$url_parada = $url_parada."isae-estimate-by-stop?lang=es&__internal__=1";
$url_parada = $url_parada."&code=$aux_code_parada";
$data_parada = file_get_contents("$url_parada");
$web_parada =  htmlentities($data_parada, ENT_QUOTES);
$tamano_parada = strlen($web_parada);
$contador_parada = 0;
$nojayautobus = 0;
```

Se descarta todo hasta llegar a “**line&quo**”.

```

while($contador_parada!=$tamano_parada)
{
if($web_parada[$contador_parada]=="l")
    {$contador_parada++;
  if($web_parada[$contador_parada]=="i")
    {$contador_parada++;
   if($web_parada[$contador_parada]=="n")
     {$contador_parada++;
    if($web_parada[$contador_parada]=="e")
      {$contador_parada++;
      if($web_parada[$contador_parada]=="&")
        {$contador_parada++;
       if($web_parada[$contador_parada]=="q")
         {$contador_parada++;
        if($web_parada[$contador_parada]=="u")
          {$contador_parada++;
         if($web_parada[$contador_parada]=="o")
           {
```

Se posiciona el array en la posición correcta y se muestra que línea está por llegar.

```
            $contador_parada = $contador_parada +10;
            echo "</br>Linea: ".$web_parada[$contador_parada]."</br>";
            $aux_control_dos = 0;
```

Se descarta todo hasta llegar a "**time&quo**".

```
while($aux_control_dos!=2)
{
if($web_parada[$contador_parada]=="t")
  {$contador_parada++;
 if($web_parada[$contador_parada]=="i")
   {$contador_parada++;
  if($web_parada[$contador_parada]=="m")
    {$contador_parada++;
   if($web_parada[$contador_parada]=="e")
     {$contador_parada++;
    if($web_parada[$contador_parada]=="&")
      {$contador_parada++;
     if($web_parada[$contador_parada]=="q")
       {$contador_parada++;
      if($web_parada[$contador_parada]=="u")
        {$contador_parada++;
       if($web_parada[$contador_parada]=="o")
```

Se posiciona el array en la posición correcta y se prepara para mostrar cuánto tiempo falta para que llegue el siguiente autobús de esa línea.

```
{$contador_parada = $contador_parada +4;
$aux_control_dos++;
echo "</br>Tiempo de espera: ";
```

Se muestra el tiempo que falta para que llegue el siguiente autobús y se actualiza la variable de control de autobuses para que el script sepa que sí pasa algún autobús.

```
while($web_parada[$contador_parada]!=",")
{
echo $web_parada[$contador_parada];
$contador_parada++;
}
echo " minutos";
$nojayautobus = 1;
```

Si no hubiera pasado ningún autobús con la siguiente comprobación se mostrará un mensaje avisando al usuario que no pasará ningún autobús.

```
if($nojayautobus == 0)
{
    echo "No hay autobuses disponibles para esta parada por el momento.";
}
?>
```

5.5.-Datos meteorológicos

El usuario tendrá a su alcance conocer el clima que hará hoy y los días sucesivos. De este modo podrá saber de antemano qué tiempo va a hacer antes de salir de casa.

5.5.1.-Vista previa

Tabla de predicción				
	2017-05-29	2017-05-30	2017-05-31	2017-06-01
Temperatura mínima y máxima	13 / 26	13 / 24	13 / 25	13 / 26
Sensación mínima y máxima	13 / 26	13 / 24	13 / 25	13 / 26
Humedad relativa mínima y máxima	50 / 85	45 / 85	45 / 80	35 / 75
Indice ultravioleta máxima	8	8	8	9

21.- Vista final tabla de temperaturas.

5.5.2.-Código fuente tiempo.php

Cargamos el archivo xml con la información del tiempo desde la web de aemet y se empieza a generar el código de la tabla que mostrará los datos de forma organizada.

```
<?php
$xml = simplexml_load_file
('http://www.aemet.es/xml/municipios/localidad_03009.xml');
?>
</br></br></br>
<center>
<table border="1">
<tr>
<th colspan="5">Tabla de prediccion</th>
</tr>
<tr>
<td> </td>
<td>
```

Se puede observar como se escribe automáticamente la fecha actual y la de días sucesivos en la tabla para saber a qué día pertenece cada información.

```
<?php
foreach($xml->prediccion->dia[0]->attributes() as $a => $b)
{
echo $b;
}
?>
</td>
<td>
<?php
foreach($xml->prediccion->dia[1]->attributes() as $a => $b)
{
echo $b;
}
?>
</td>
<td><?php
foreach($xml->prediccion->dia[2]->attributes() as $a => $b)
{
echo $b;
}
?></td>
<td>
<?php
foreach($xml->prediccion->dia[3]->attributes() as $a => $b)
{
echo $b;
}
```

Se escribe de forma automática la temperatura mínima y máxima que habrá en el día de hoy y en los posteriores.

```
</td>
</tr>
<tr>
<td>Temperatura m&iacute;nima y m&aacute;xima</td>
<td>
<?php echo "".$xml->prediccion->dia[0]->temperatura->minima."
 / ".$xml->prediccion->dia[0]->temperatura->maxima.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[1]->temperatura->minima."
 / ".$xml->prediccion->dia[1]->temperatura->maxima.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[2]->temperatura->minima."
 / ".$xml->prediccion->dia[2]->temperatura->maxima.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[3]->temperatura->minima."
 / ".$xml->prediccion->dia[3]->temperatura->maxima.""; ?>
</td>
</tr>
```

Se escribe de forma automática la sensación mínima y máxima que habrá en el día de hoy y en los posteriores.

```
<tr>
<td>Sensaci&oacute;n m&iacute;nima y m&aacute;xima</td>
<td>
<?php echo "".$xml->prediccion->dia[0]->sens_termica->minima."
 / ".$xml->prediccion->dia[0]->sens_termica->maxima.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[1]->sens_termica->minima."
 / ".$xml->prediccion->dia[1]->sens_termica->maxima.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[2]->sens_termica->minima."
 / ".$xml->prediccion->dia[2]->sens_termica->maxima.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[3]->sens_termica->minima."
 / ".$xml->prediccion->dia[3]->sens_termica->maxima.""; ?>
</td>
</tr>
```

Se escribe de forma automática la humedad relativa mínima y máxima que habrá en el día de hoy y en los posteriores.

```
<tr>
<td>Humedad relativa m&iacute;nima y m&aacute;xima</td>
<td>
<?php echo "".$xml->prediccion->dia[0]->humedad_relativa->minima."
 / ".$xml->prediccion->dia[0]->humedad_relativa->maxima.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[1]->humedad_relativa->minima."
 / ".$xml->prediccion->dia[1]->humedad_relativa->maxima.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[2]->humedad_relativa->minima."
 / ".$xml->prediccion->dia[2]->humedad_relativa->maxima.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[3]->humedad_relativa->minima."
 / ".$xml->prediccion->dia[3]->humedad_relativa->maxima.""; ?>
</td>
</tr>
```

Se escribe de forma automática el índice ultravioleta máximo que habrá en el día de hoy y en los posteriores.

```
<tr>
<td>Indice ultravioleta m&aacute;xima</td>
<td>
<?php echo "".$xml->prediccion->dia[0]->uv_max.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[1]->uv_max.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[2]->uv_max.""; ?>
</td>
<td>
<?php echo "".$xml->prediccion->dia[3]->uv_max.""; ?>
</td>
</tr>
</table>
```

Se genera diversa información útil para el ciudadano como es la fecha actual y la hora, el día actual en número del año en que nos encontramos y los días que quedan del año.

```
<?php
echo date('m/d/Y g:ia');
$fecha = date('m/d/Y');
echo "</br>Dia actual del a&ntilde;o:";
echo date('z',strtotime($fecha));
$numero = date('z',strtotime($fecha));
$resta = 365 - $numero;
echo "</br>Numero de dias que quedan para finalizar el a&ntilde;o: ".$resta;
```

6.-Bibliografía

Php:
http://php.net/

- Web oficial del lenguaje php.

java + socket:
https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html

- Web oficial del tratamiento de los sockets en java.

Mysql:
https://dev.mysql.com/doc/

- Web oficial del lenguaje mysql.

Método Get:
https://www.w3schools.com/tags/att_form_method.asp

- Documentación del método GET. Cómo funciona el metodo GET en profundidad.

Diferencias método GET y POST
https://www.w3schools.com/tags/ref_httpmethods.asp

- Documentación sobre las diferencias del método get y post. Aclaraciones sobre los distintos usos del GET y del POST.

Google Maps:
https://developers.google.com/maps/documentation/javascript/?hl=es-419

- Documentación oficial de la api de los mapas de google. Donde se obtuvo su “api” para utilizar su sistema de mapas para implementarlos en la página web.

Foro api maps:
https://productforums.google.com/forum/#!forum/maps

- Foro oficial de la api de los mapas de google. Donde se resolvieron dudas sobre su implementación.

Web Scraping
https://sitelabs.es/web-scraping-introduccion-y-herramientas/

- Explicación detallada de web scraping. Se utilizaron sus técnicas para recoger la información para posteriormente mostrarla en la página web.

Web sobre css
https://www.w3schools.com/css/

- Web detallada con documentación sobre css.

Web aemet
http://www.aemet.es/es/portada

- Web oficial sobre la meteorología de España. De aquí se extraen los datos que posteriormente son utilizados en la página web.

Web del ayuntamiento de Alcoy
http://ajualcoi.org

- Web oficial del ayuntamiento de Alcoy. El proyecto es alojado en este servidor.

Web de los autobuses de Alcoy
https://alcoi.vectalia.es/

- Web oficial de los autobuses de Alcoy. De aquí se extraen los datos que posteriormente son utilizados en la página web.

7.-Conclusiones finales

Una vez finalizado el proyecto creo que será de mucha utilidad para el ciudadano en su día a día ya que le acerca una información que sin este dashboard estaría muy esparcida por muchos sitios de difícil acceso y aquí queda recogida para mostrarla de forma rápida y muy intuitiva, como son:

- Transporte urbano.
- Climatología.
- Tráfico en tiempo real e histórico.

7.1.-Posibles ampliaciones

Para poder seguir ayudando al ciudadano este proyecto puede seguir ampliándose en muchos aspectos para intentar ofrecer más información que le pueda ser útil como por ejemplo:

- Puntos de interés.
- Teléfonos de contacto (Policía, ayuntamiento, etc).
- Información sobre la ciudad.
- App para dispositivos móviles.

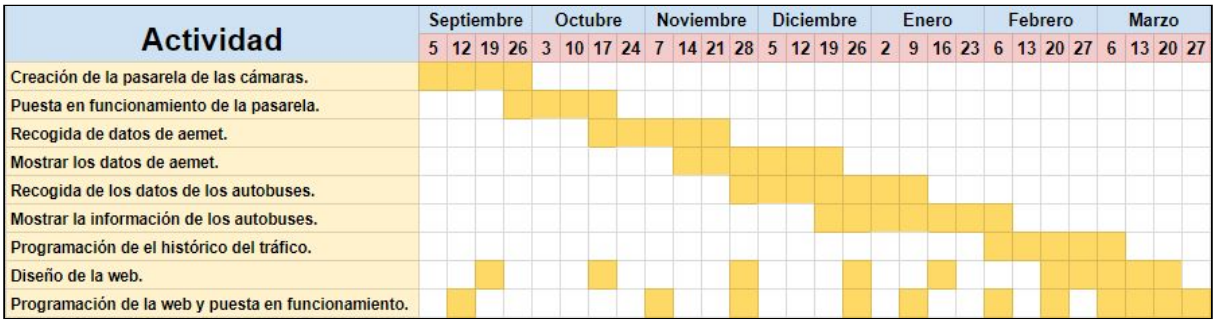
8.- Planificación

Se emprende el proyecto en el mes de septiembre de 2016 y se estima su finalización alrededor de marzo de 2017.

Dentro del proyecto había una serie de fases a realizar:

1.- Creación de la pasarela de las cámaras.
2.- Puesta en funcionamiento de la pasarela.
3.- Recogida de datos de aemet.
4.- Mostrar los datos de aemet.
5.- Recogida de los datos de los autobuses.
6.- Mostrar la información de los autobuses.
7.- Programación de el histórico del tráfico.
8.- Diseño de la web.
9.- Programación de la web y puesta en funcionamiento.

En la siguiente tabla se muestran los periodos de realización de cada una de las fases del proyecto.



22.- Imagen del tiempo en que se realizó el proyecto.