ANEXO C: MANUAL TÉCNICO

C.1 INTRODUCCIÓN

La finalidad del manual técnico es proporcionar la lógica con que se ha desarrollado la aplicación

C.2 Objetivo

Proporcionar una guía de las funciones, clases y código de programación utilizado, para que se pueda en cualquier momento corregir errores o seguir implementando el Sistema.

C.3 Clases, Funciones y código de programación Utilizados para el desarrollo de la Aplicación

Para entender los procesos internos que realiza el sistema debemos entender cada uno de los objetos de nuestra clase generada; como se nota a continuación.

Función de conexion.php.- permitirá el enlace de la base de datos con la ***clasegeneral.php*** función de nombre ***conexion.php.***

```
$conex = pg_connect("host=localhost port=5432 dbname=".$_SESSION["basenactivadatrabajo"]."
user=xxxx  password=xxxx");
```

Cuadro 1. Función de conexión

La función ***pg_connect( )*** recibe los datos del puerto de entrada-salida 5432 que le permitirá la relación del interprete o lenguaje de ***.php*** con la base de datos ***base_patentes*** de ***postgres*** mediante la variable ***basenactivadatrabajo*** y los otros dos datos son el usuario y el password o clave, que permitirá conjugar al resto de objetos que explicaremos más adelante.

Clase clasegeneral.php.- la clase principal denominada ***class mismetodos*** es la que nos permitirá realizar funciones como: ingreso, actualización, eliminación, conexión, traer datos, cálculo de tiempos, auditoria, captura de rutas de ingreso al sistema.

Objeto *mismetodos($cone)*.- permitirá la actualización automática de la base de datos en la tabla ***tbl_historial_pago***, mediante un recorrido de la tabla, ***tbl_local*** donde se guarda las actividades o locales, esto permitirá saber el estado (no declarado, declarado, pagado, pago por alcance) de cada local, esto permitirá que el sistema actualice cada nuevo año cuales son las actividades que serán tomadas como coactiva.

```
function mismetodos($cone){
$this->cad_conexion=$cone;
$anio_actual=date("Y");
$estado='nodeclarado';
$datos = $this->traerdatos("select id_local from tbl_local where id_local not in (select id_local from
tbl_historial_pago where anio_apagar='$anio_actual')");
        for($i=0;$i<count($datos);$i++){
                $fila = $datos[$i];
                $id_local = $fila[0];
                $mi_ingreso="INSERT INTO tbl_historial_pago(id_local, anio_apagar,estado)
VALUES($id_local,$anio_actual,'$estado')";
                $this->ejecutarsql($mi_ingreso);
        }
```

Cuadro 2. Función de actualización automática para coactiva de actividades económicas

Objeto traerdatos($consulta).- permitirá realizar un recorrido a cualquier consulta que ingrese como parámetro, almacena en un *array* y retornará dichos datos consultados.

```
function traerdatos($consulta){
   global $conex;
   $datos = array();
   $result = @pg_query($conex,$consulta);
   while ($row = @pg_fetch_array($result)){
      $datos[] = $row;
   }
   return $datos;
        }
```

Cuadro 3. Función de recorrido y retorno de un array.

Objeto n_dias($txtFechInicio,$txtFechaFinal).- permitirá hacer los cálculos de fechas que serán recibidas mediante parámetros como fecha de inicio y fecha final, dato retornado será un número entero.

```
function n_dias($txtFechInicio,$txtFechaFinal)    {
 $txtDuracion =(strtotime($txtFechInicio)-strtotime($txtFechaFinal))/86400;
 $txtDuracion = abs($txtDuracion);
$txtDuracion = floor($txtDuracion);
   return  $txtDuracion;
   }
```

Cuadro 5. Función para cálculo de fechas**.**

Objeto Agregar($tabla ="",$campos="",$valores="").- nos permitirá el ingreso de datos en cualquiera de las tablas de la base de datos, datos que serán ingresados mediante parámetros como: nombre de la tabla, los campos de la tabla y los valores a ser ingresados.

```
function Agregar($tabla ="",$campos="",$valores="")
{
$mirestsol= "INSERT INTO $tabla($campos) VALUES($valores)";  “Inserción  de datos”
pg_set_client_encoding($this->cad_conexion,"LATIN1");
$result = @pg_query($this->cad_conexion,$mirestsol);
return true;
}
```

Cuadro 6. Función Agregar.

Objeto Modificar ($tabla,$micamp,$elvec).- permitirá la modificación de los datos almacenados en cualquier tabla de la base de datos, mediante los parámetros como: nombre de la tabla, los campos de la tabla y los valores a ser modificados.

```
function Modificar($tabla,$micamp,$elvec){
$numelementos = count($micamp);

// ver exitencia de datos
$seleccionodato = "select * FROM $tabla WHERE ".$micamp[0]." = '".$elvec[0]."'";
$resultaselec = @pg_query($this->cad_conexion,$seleccionodato);

// recorrido y verificación de existencia de datos iguales
$query = "UPDATE ".$tabla." SET ";
for($i=1;$i<$numelementos;$i++)   {
if($i==$numelementos-1)   {
$query .= $micamp[$i]."='".$elvec[$i]."'";
}
else
$query .= $micamp[$i]."='".$elvec[$i]."',";   }

// actualización de los nuevos datos en la tabla
$query .= " WHERE ".$micamp[0]."='".$elvec[0]."'";
pg_set_client_encoding($this->cad_conexion,"LATIN1");
$result = @pg_query($this->cad_conexion,$query);
// captura de error  en actualización
if ($result) {
return true;
}
else{
pg_last_error($this->cad_conexion);
return false;
}
```

Cuadro 7. Función Modificar de datos.

Objeto borrado ($tabla,$micamp,$elvec).- permitirá la eliminación de los datos almacenados en cualquier tabla de la base de datos, mediante los parámetros como: nombre de la tabla, los campos de la tabla y los valores a ser eliminados.

```
function borrado($tabla,$micamp,$elvec)
{

//comparación del datos y actualización de la tabla
 $Borrado = "DELETE FROM $tabla WHERE ".$micamp[0]." = '".$elvec[0]."'";
 $this->result = @pg_query($this->cad_conexion,$Borrado);
 return $this->result;

}
```

Cuadro 8. Función de eliminación de datos

Objeto Consultas($sql = "").- permitirá hacer consultas de los datos almacenados en cualquier tabla de la base de datos, mediante el parámetro de ***consulta sql*** o una ***vista***.

```
function Consultas($sql = "")
{
if($sql == ""){
$this->Error = "No se ha Realizado Ninguna Consulta";
return 0;
}
$this->result = @pg_query($this->cad_conexion,$sql);
//si hubo selección de datos
        return $this->result;
}
```

Cuadro 9. Función de consulta de datos.

Robot auditoria($nombreusuario,$sql).- permitirá la consulta y asignación de la acción que el usuario realice en los datos almacenados en cualquier tabla de la base de datos, mediante los parámetros como: nombre del usuario que haya ingresado al sistema y la ***consulta sql***. Que haya realizado al momento que haga un proceso; cada acción será registrado, como son:

- Usuario de ingreso
- Dirección IP del equipo de trabajo que acceda el usuario
- Fecha de ingreso
- Nombre de la tabla donde se está accediendo
- Acción que se realizó

- Datos que fueron modificados
- Hora del ingreso

Hay que notar que esta función ***robot*** de nombre ***auditoria($nombreusuario,$sql)*** se encuentra dentro del archivo ***clasegeneral.php*** pero no pertenece a la clase general.}

```
function auditoria($nombreusuario,$sql){
        $nombretabla = "";
        $accion = "";
        $transaccion = "";
        $sql = strtolower($sql);
        $ipos = strpos($sql,"tbl_");
        if($ipos===false){return;
        }
        else{   $nuevacad = substr($sql,$ipos,strlen($sql));
                $ipos = strpos($nuevacad," ");
                if($ipos===false){
                        $nombretabla = $nuevacad;
                }
                else{   $nuevacad = substr($nuevacad,0,$ipos);
                        $nombretabla = $nuevacad;}
        }
        $ipos = strpos($sql,"insert");
        if($ipos===false){
                $ipos = strpos($sql,"update");
                if($ipos===false){
                        $ipos = strpos($sql,"delete");
                        if($ipos===false){
                                $ipos = strpos($sql,"select");
                                if($ipos===false){
                                        return;
                                }
                                else{ $accion = "SELECCION";
                                }
                        }
                        else{   $accion = "ELIMINAR";
                        }}
                else{   $accion = "ACTUALIZAR";}       }
        else{   $accion = "INSERTAR";
        }
```

Cuadro 10. Función de auditoría.

Objeto InicioSesion($nombreusuario).- permitirá saber quién está ingresando al sistema, captura el lugar de ingreso (dirección IP), el tipo de usuario, fecha, hora, la transacción.

```
function InicioSesion($nombreusuario){
        $nombreusuario=$_SESSION["usuario_ingreso"];
        $conex = pg_connect("host=localhost   port=5432 dbname=base_patentes
user=postgres password=postgres");
        $ip = $_SERVER['REMOTE_ADDR'];
        $sql = str_replace("'", "", $sql);
        $sql = "insert into
tbl_seguridad_auditoria(usuario,ip,fecha,tabla,accion,transaccion,hora)
values('$nombreusuario','$ip',NOW(),'tbl_usuario','iniciosesion','inicio de sesion',NOW())";
        $result = @pg_query($conex,$sql);
}
```

Cuadro 11. Robot de Sesión del sistema**.**

Mediante el parámetro ***$_SESSION["usuario_ingreso"]*** de usuario capturamos quien ingresa al sistema, con la variable ***$_SERVER['REMOTE_ADDR']*** obtengo la dirección del host de ingreso, con estos datos escribo en la tabla de auditoria del sistema; de la misma forma para fin de sesión.

cronjobs.- tarea automática mediante la utilización del archivo ***actualiza_arriendo.bat*** para actualizar la tabla de arriendos; este archivo interactúa con la configuración de tareas de Windows, donde se asigna a una cierta hora y fechas para que se active tarea, en este caso activa al archivo ***actualiza_arriendo.php*** que hace un recorrido de la tabla ***tbl_arriendo***, como se nota a continuación.

```
<?php
require_once 'lib/conexion_admin.php';  //conexión a la clase
require_once 'lib/clasegeneral.php';
$miobj = new mismetodos($conex);

$datos_fecha=$miobj->Consultas("SELECT a.id_arriendo, a.id, a.sel_id as el_id, a.sel_id2,
a.id_tiempo, a.subtotal,a.id_factura_arriendo,a.fecha_desde, a.fecha_fin as
la_fechafin,se.id,se.opcion,se.estado  FROM tbl_arriendo a, select_2 as se where
a.sel_id=se.id and se.estado='NO' ");
while ($row = @pg_fetch_array($datos_fecha)){
        $id_arriendo = $row["el_id"];
        $fecha_fin = $row["la_fechafin"];
        $hoy = date("Y-m-d");
        if($hoy >= $fecha_fin){
        $cambio_estado='SI';
    $miobj->Consultas("update select_2 set estado='".$cambio_estado."' where id
='".$id_arriendo."' ");

        }
}
?>
```

Cuadro 10. cronjobs de actualización automática para arriendos del sistema**.**

Hay que notar que para poder ejecutar el ***cronjobs*** es necesario asignar la ruta física del archivo ***actualiza_arriendo.php*** y ejecutar el a***pache\cgi-bin\php.exe*** el cual se encuentra dentro del servidor ***mapserver***, como se nota:

```
F:\ms4w\Apache\cgi-bin\php.exe    //ruta física de php.exe y del  archivo .php

"F:\ms4w\Apache\htdocs\municipiomontufar\patentes\patentes_cronjobs\scripts\actualiza_a
rriendo.php"
```

Cuadro 11. Archivo .bat de llamada de tareas automáticas del S.O.

mapa_basepredios.map.- permitirá mostrar el mapa visor de los predios del área en estudio, donde podremos realizar configuraciones como:

- Color
- Iconos
- Coordenadas de posición

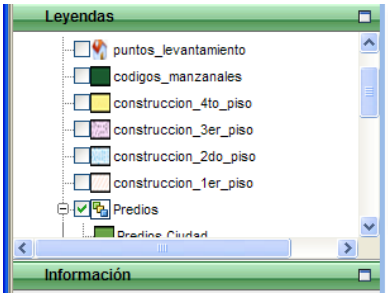
- Estilos de letra
- Transparencias
- Tamaño
- Capas o layers
- Conexión de la base de datos
- Tamaño
- Leyenda

```
LAYER  //inicio de capa
   NAME 'perfil_manzanero_n'   // nombre de la capa
   TYPE POLYGON     // soporte de datos tipo Polígono
   DUMP true // coordenadas  geográficas de la capa
  #EXTENT 183888.626751 10065971.890330 185055.817647 10066524.715330
   CONNECTIONTYPE postgis     // conexión a la base de datos
   CONNECTION "dbname='base_patentes' host=localhost port=5432  user=xxxxx
password=xxxxxx"
// consulta a la tabla en el esquema indicado cartografia
   DATA 'the_geom FROM "cartografia"."perfil_manzanero_n" USING UNIQUE gid USING
srid=23030'
   METADATA  // titulo de la capa
     'ows_title' 'perfil_manzanero_n'
   END
   STATUS OFF
   TRANSPARENCY 100  // manejo de transparencias
  PROJECTION
              "init=epsg:4326"   // tipo de proyección  asignado por PostSig
            END
   CLASS
      NAME 'perfil_manzanero_n'  // nombre layer  para el visor
     SYMBOL manzana  // icono representativo
      #STYLE
      #  WIDTH 0.91
        OUTLINECOLOR 255 66 0   // color de líneas
       #COLOR 8 25 248
           SIZE 15              // grosor de líneas
            TEMPLATE void
      #END
   END
  END  // fin de layer
```

Cuadro 12. Configuración de Visor de mapa

Hay que notar que en la parte de llamadas a símbolos se lo debe hacer mediante ***SYMBOL*** y se debe almacenar en la carpeta ***symbol*** para que sea reconocida

automáticamente, en este caso se manejara iconos (.***png***) para las capas de construcción y posicionamiento predial, como se nota en la imagen.



Listado de leyendas

Las capas del visor deben ser contenidas en la dirección física ***ms4w\apps\cartografia_patente}*** y exportadas en esa dirección mediante la utilización de las siguientes instrucciones del archivo ***.bat***.

mkdir cartografia_patentes // creación del contenedor for %%x in (../procesadoshape_final/*.shp) do shp2pgsql -s 23030 ../procesadoshape_final/%%~nx cartografia.%%~nx > cartografia_patentes/%%~nx.sql // Ciclo de volcado de archivos .***shp*** a extenciones .***sql***; esto evitara tener errores al subir a la base de datos ***postgres*** y será mucho más rápido.

Cuadro 13. Volcamiento de datos .shp a extensión .sql

Luego de haber transformado los archivos ***.shp*** a ***.sql*** debemos subir a la base de datos de ***postgres.*** Con la siguiente instrucción del archivo ***.bat.***

SET PGPASSWORD=xxxx // clave de acceso para postgres for %%x in (../procesadoshape_final/*.shp) do psql -U postgres -d base_patentes -f cartografia_patentes/%%~nx.sql // ciclo para enviar los datos ***.sql*** a ***postgres***. En el esquema ***cartografia_patentes***

Cuadro 14. Volcamiento de datos .sql a la base de datos base_patentes en postgres

Terminado de configurar el servidor de mapas y enviado a la base de datos debemos agregar en el aplicativo el archivo del visor, en este caso sería la siguiente ruta.

visor_geografico/templates/mimapservidor/estandard/index.php // ruta del visor

// Configuración del archivo Mapabasepatentes.xml

<MapFile>/ms4w/apps/cartografia_patentes/mapfile/mapa_basepredios.map</MapFile>

// ruta del todos los archivos .shp

Cuadro 15. Ruta del visor geográfico

- **Tratamiento de la base de datos base_patentes**

Para que la base de datos soporte datos geo-espaciales deberemos crearla de la siguiente manera.

```
CREATE DATABASE base_patentes WITH ENCODING='UTF8' OWNER=postgres
TEMPLATE=template_postgis CONNECTION LIMIT=-1  TABLESPACE=pg_default;
```

Cuadro 16.creación de la base de datos geo-espacial de nombre base_patentes

Funciones, vistas y disparadores dentro ***base_patentes***

view v_ciudades.- permitirá visualizar la lista de ciudades ingresadas
view v_ciudadano.- permitirá visualizar la lista de usuarios
view v_usuariosadmin.- permitirá visualizar la lista de usuario del sistema
view v_actividad.- permitirá visualizar la lista de actividades económicas activas
view v_catastral.- permitirá visualizar la lista de todos los predios con su respectiva clave catastral.
view v_local.- permitirá visualizar la lista de actividades económicas asignadas a un usuario activo para pago de patente.
view v_datos_localtodo.- permitirá visualizar la lista de locales activos
view v_todo_local.- permitirá visualizar la lista de locales de cada mercado
view v_mercado_local.- permitirá visualizar la lista de locales asignadas a un mercado
view v_local_valor.- permitirá visualizar la lista de precios por local de mercado
view v_imprime_fac_arriendo.- permitirá visualizar los datos para impresión
view v_factura_arriendo.- permitirá visualizar la lista de arriendos realizados a la fecha
view v_impuesto_patenteobligado.- permitirá visualizar la lista de patentes generadas para cada usuario.

view v_impuesto_patenteobligado_idciudadano.- permitirá visualizar los datos del usuario relacionados con cada actividad activa.

view v_impuesto_patenteobligado_idciudadanoxmil.- permitirá visualizar los datos del usuario relacionados con los datos del impuesto a los activos totales.

view v_pago_patenteobligado_formulario.- permitirá visualizar datos del formulario para la declaración de la patente para cada usuario.

view v_pago_patenteobligado.- permitirá visualizar la lista de usuarios obligados a llevar contabilidad que deben pagar el valor de patente.

view v_pago_patenteobligado.- permitirá visualizar la lista de usuarios no obligados a llevar contabilidad que deben pagar el valor de patente.

view v_mostrar_formulariopatente.- permitirá visualizar datos del formulario para su verificación antes del pago del valor de patente.

view v_mostrarformuxMil.- permitirá visualizar los datos del formulario para ser verificados antes del pago del valor impuesto a activos totales.

view v_actividad_especial.- permitirá visualizar la lista de actividades especiales de usuarios de personería natural.

view v_grupos_actividad.- permitirá visualizar la lista de categorías de actividades existentes y activas.

view v_master.- permitirá visualizar datos en la interfaz previa al pago de una patente.

view v_anio_descuento.- permitirá visualizar la lista de años donde se aplicarán descuentos para pagos.

view v_master_coactiva.- permitirá visualizar datos para cálculos de actividades atrasadas de pago.

view v_declara_usuario_formu.- permitirá visualizar la lista de usuarios que han declarado hasta la fecha.

view v_coactivas.- permitirá visualizar la lista de usuarios que adeudan más d un año.

function function_tempo_procesadopatente( ).- permitirá copiar los datos del formulario ingresados por el usuario previos al pago de la patente.

trigger trigger_pasoparametrospatente.- permitirá ejecutar la copia de datos a la tabla definitiva cuando sea revisado el formulario previo al pago de patente.

function function_tempo_procesadxmil( ).- permitirá copiar los datos del formulario ingresado por el usuario previo al pago de patente.

trigger trigger_pasoparametrosxmil.- permitirá ejecutar la copia de datos a la tabla definitiva cuando sea revisado el formulario previo al pago de patente.

function function_tempo_procesadonocontable( ).-permitirá copiar los datos del formulario ingresados por el usuario no obligados a llevar contabilidad previos al pago de la patente.

trigger trigger_pasoparametronocontable.- permitirá ejecutar la copia de datos a la tabla definitiva cuando sea revisado el formulario previo al pago de patente de los usuarios no obligados a llevar contabilidad.

function function_tempo_repartoactivos( ).- permitirá copiar los datos del formulario ingresados por el usuario previos al pago de la patente.

trigger trigger_pasoparametrosreparto.- permitirá ejecutar la copia de datos a la tabla definitiva cuando sea revisado el formulario previo al pago de patente de los usuarios.