Contenido

[Controllers 2](#_Toc200973337)

[Agenda.php 2](#_Toc200973338)

[Archivos.php 5](#_Toc200973339)

[Autos.php 7](#_Toc200973340)

[Bitacora.php 21](#_Toc200973341)

[Cafeteria.php 22](#_Toc200973342)

[Clientes.php 26](#_Toc200973343)

[Compras.php 28](#_Toc200973344)

[Configuracion.php 92](#_Toc200973345)

[Correo.php 106](#_Toc200973346)

[Cotizaciones.php 109](#_Toc200973347)

# Controllers

## Agenda.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

// Controlador Agenda para manejar eventos y reuniones

class Agenda extends CI\_Controller {

    // Constructor: carga modelos y librerías necesarias

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('agenda\_model','Modelo');          // Modelo para agenda

        $this->load->model('tickets\_IT\_model', 'ITModelo');   // Modelo para tickets IT

        $this->load->library('correos');                      // Librería para envío de correos

        $this->load->model('conexion\_model', 'Conexion');     // Modelo para conexiones DB

    }

    // Página principal: muestra el calendario

    public function index() {

        $this->load->view('header');                     // Carga el header

        $this->load->view('agenda/calendario');          // Carga la vista del calendario

    }

    // Obtiene eventos para mostrarlos en el calendario

    function getEventos()

    {

        $data = $this->Modelo->getEventos();  // Obtiene eventos del modelo

        echo json\_encode($data);              // Devuelve datos en formato JSON

    }

    // Crea un nuevo evento o reunión

    function crearEvento()

    {

        $reunion = $\_POST['reunion'];  // Determina si es reunión (1) o evento normal

        // Datos básicos del evento

        $datos = array(

            'usuario' => $this->session->id,       // ID del usuario de sesión

            'titulo' => $\_POST['titulo'],          // Título del evento

            'inicia' => $\_POST['inicia'],          // Fecha/hora de inicio

            'termina' => $\_POST['termina'],        // Fecha/hora de fin

            'descripcion' => $\_POST['descripcion'],// Descripción

            'sala' => $\_POST['sala'],              // Sala asignada

        );

        // Si es una reunión (valor 1)

        if ($reunion == 1) {

            $datos['equipo'] = 1;                  // Marca como reunión de equipo

            $datos['correos'] = $\_POST['tags\_1'];  // Correos de invitados

            echo $this->Modelo->insertEvent($datos); // Inserta el evento

            // Crea un ticket asociado a la reunión

            $data = array(

                'usuario' => $this->session->id,

                'tipo' => 'Reunion',               // Tipo de ticket

                'titulo' => $\_POST['titulo'] . ' -- Fecha/Hora: ' . $\_POST['inicia'] . ' hasta ' . $\_POST['termina'],

                'descripcion' => $\_POST['descripcion'] . ' -- Fecha/Hora: ' . $\_POST['inicia'] . ' hasta ' . $\_POST['termina'] . ' agregar correos: ' . $\_POST['tags\_1'],

                'estatus' => 'ABIERTO',            // Estado inicial

                'cierre' => '0',                  // No cerrado

            );

            $last\_id = $this->ITModelo->crear\_ticket($data); // Crea ticket y obtiene ID

            // Prepara datos para correo de notificación

            $datosCorreo = array(

                'id' => $last\_id,

                'prefijo' => 'IT',

                'titulo' => $data['titulo'],

                'fecha' => date('d/m/Y h:i A'),

                'usuario' => $this->session->nombre,

                'correo' => $this->session->correo,

            );

            $this->correos->creacionTicket($datosCorreo); // Envía correo

            // Redirecciona a la página de archivos del ticket

            redirect(base\_url('tickets\_IT/archivos/') . $last\_id);

        } else {

            // Si no es reunión, solo inserta el evento normal

            echo $this->Modelo->insertEvent($datos);

        }

    }

    // Elimina un evento

    function borrarEvento()

    {

        $id = $\_POST['id'];  // ID del evento a borrar

        // Intenta borrar y devuelve resultado (1=éxito, 0=fallo)

        if($this->Modelo->deleteEvent($id) > 0)

        {

          echo "1";

        }

        else {

          echo "0";

        }

    }

    // Valida disponibilidad de sala en un horario específico

    function validacion()

    {

        $inicia=$this->input->post('inicia');  // Hora de inicio

        $termina=$this->input->post('termina'); // Hora de fin

        $sala=$this->input->post('sala');      // Sala a verificar

        // Consulta eventos que se solapen con el horario solicitado

        $res = $this->Conexion->consultar("SELECT \* FROM `agenda` WHERE inicia <= '".$termina."' AND termina >= '".$inicia."' and sala=".$sala);

      if($res)

      {

        echo json\_encode($res);  // Devuelve eventos conflictivos

      }

    }

}

## Archivos.php

<?php defined('BASEPATH') OR exit('No direct script access allowed');

// Controlador para manejo de archivos (PDF, documentos y descargas)

class Archivos extends CI\_Controller {

    // Constructor: carga el modelo de conexión a base de datos

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('conexion\_model', 'Conexion');

    }

    // Muestra un archivo PDF en línea desde la base de datos

    function pdf() {

        // Verifica que existan los parámetros requeridos

        if(isset($\_POST['tabla']) && isset($\_POST['id'])) {

            $tabla = $this->input->post("tabla");  // Tabla donde está almacenado

            $id = $this->input->post("id");        // ID del registro

            // Obtiene el archivo y su nombre desde la base de datos

            $res = $this->Conexion->consultar("SELECT nombre, archivo from $tabla where id = $id", TRUE);

            $nombre = $res->nombre;  // Nombre original del archivo

            $file = $res->archivo;    // Contenido binario del PDF

            // Configura headers para visualización en navegador

            header('Content-type: application/pdf');

            header('Content-Disposition: inline; filename="' . $nombre . '"');

            header('Content-Transfer-Encoding: binary');

            header('Accept-Ranges: bytes');

            echo $file;  // Imprime el contenido del PDF

        } else {

            // Redirecciona si faltan parámetros

            redirect(base\_url('inicio'));

        }

    }

    // Muestra un documento genérico (similar a pdf() pero con campo personalizable)

    function documento() {

        // Verifica parámetros requeridos (incluyendo el campo específico)

        if(isset($\_POST['tabla']) && isset($\_POST['id']) && isset($\_POST['campo'])) {

            $id = $this->input->post("id");     // ID del registro

            $tabla = $this->input->post("tabla"); // Tabla de origen

            $campo = $this->input->post("campo"); // Campo que contiene el archivo

            // Obtiene el archivo desde la base de datos

            $res = $this->Conexion->consultar("SELECT $campo from $tabla where id = $id", TRUE);

            $file = $res->$campo;  // Contenido binario del documento

            // Configura headers para visualización en navegador

            header('Content-type: application/pdf');

            header('Content-Disposition: inline; filename="' . $campo . '"');

            header('Content-Transfer-Encoding: binary');

            header('Accept-Ranges: bytes');

            echo $file;  // Imprime el contenido del documento

        } else {

            // Redirecciona si faltan parámetros

            redirect(base\_url('inicio'));

        }

    }

    // Descarga un archivo XML desde la base de datos

    function descarga() {

        // Verifica parámetros requeridos

        if(isset($\_POST['tabla']) && isset($\_POST['id']) && isset($\_POST['campo'])) {

            $id = $this->input->post("id");     // ID del registro

            $tabla = $this->input->post("tabla"); // Tabla de origen

            $campo = $this->input->post("campo"); // Campo que contiene el XML

            // Obtiene el archivo XML desde la base de datos

            $res = $this->Conexion->consultar("SELECT $campo from $tabla where id = $id", TRUE);

            $file = $res->$campo;  // Contenido del XML

            // Configura headers para descarga

            header('Content-type: text/xml');

            echo $file;  // Imprime el contenido XML

        } else {

            // Redirecciona si faltan parámetros

            redirect(base\_url('inicio'));

        }

    }

}

## Autos.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

// Controlador para gestión de vehículos (autos)

class Autos extends CI\_Controller {

    // Constructor: carga modelos y librerías necesarias

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('autos\_model','Modelo');          // Modelo principal de autos

        $this->load->model('privilegios\_model');             // Modelo de privilegios

        $this->load->model('usuarios\_model');                // Modelo de usuarios

        $this->load->model('descargas\_model');               // Modelo para descargas

        $this->load->library('correos');                     // Librería de envío de correos

        $this->load->model('conexion\_model', 'Conexion');    // Modelo de conexión a DB

    }

    // Muestra el catálogo principal de vehículos

    function index() {

        $datos['autos'] = $this->Modelo->getCatalogo();  // Obtiene listado de autos

        $datos['titulo'] = "Autos";                     // Título para la vista

        $this->load->view('header');                     // Carga el header

        $this->load->view('autos/catalogo',$datos);      // Carga la vista principal

    }

    // Muestra detalles de un vehículo específico

    function ver($idauto) {

        // Actualiza responsable si se envió el formulario

        $responsable = $this->input->post('responsable');

        if(isset($responsable)) {

            $this->Modelo->updateAuto($idauto, array('responsable' => $responsable));

        }

        // Obtiene datos del auto y usuarios disponibles

        $datos['auto'] = $this->Modelo->getAuto($idauto);

        $datos['responsable'] = $this->usuarios\_model->getUsuario($datos['auto']->responsable);

        $datos['usuarios'] = $this->usuarios\_model->getUsuarios();

        $this->load->view('header');

        $this->load->view('autos/ver\_auto', $datos);

    }

    // Muestra vista de seguimiento GPS

    function gps() {

        $datos['autos'] = $this->Modelo->gpsAutos();  // Obtiene autos con GPS

        $datos['titulo'] = "GPS";                     // Título para la vista

        $this->load->view('header');

        $this->load->view('autos/gps',$datos);

    }

    // Muestra formulario para registrar GPS de un vehículo

    function alta\_gps($idauto) {

        $datosCoche = $this->Modelo->getAuto($idauto);

        $datos['auto\_foto'] = $datosCoche->foto;

        $datos['auto'] = $datosCoche->id;

        $datos['auto\_marca'] = $datosCoche->marca;

        $datos['auto\_combustible'] = $datosCoche->combustible;

        $datos['auto\_placas'] = $datosCoche->placas;

        $datos['gps'] = $this->Modelo->getGPS($idauto);

        $datos['titulo'] = "Autos";

        $this->load->view('header');

        $this->load->view('autos/alta\_gps',$datos);

    }

    // Muestra formulario para editar vehículo

    function editar($idauto) {

        $datos['auto'] = $this->Modelo->getAutos($idauto);

        $datos['usuarios'] = $this->privilegios\_model->listadoJefes();

        $this->load->view('header');

        $this->load->view('autos/editar',$datos);

    }

    // Muestra vista de uso de vehículos

    function uso\_autos() {

        $this->load->view('header');

        $this->load->view('autos/uso\_autos');

    }

    // Muestra formulario para registrar nuevo vehículo

    function alta\_autos() {

        $datos['usuarios'] = $this->privilegios\_model->listadoJefes();

        $this->load->view('header');

        $this->load->view('autos/alta\_autos', $datos);

    }

    // Muestra revisiones de un vehículo específico

    function revisiones\_ANT($auto) {

        $datos['auto'] = $auto;

        $datos['rev'] = $this->Modelo->getRevisiones($auto);

        $this->load->view('header');

        $this->load->view('autos/revisiones', $datos);

    }

    // Muestra opciones de revisiones

    function revisiones() {

        $this->load->view('header');

        $this->load->view('autos/revisiones\_opciones');

    }

    // Muestra revisiones pendientes

    function revisiones\_pendientes() {

        $datos['autos'] = $this->Modelo->getRevPendientes();

        $datos['titulo'] = "Revisiones Pendientes";

        $this->load->view('header');

        $this->load->view('autos/catalogo',$datos);

    }

    // Muestra revisiones programadas para hoy

    function revisiones\_hoy() {

        $datos['autos'] = $this->Modelo->getRevHoy();

        $datos['titulo'] = "Revisiones de Hoy";

        $this->load->view('header');

        $this->load->view('autos/catalogo',$datos);

    }

    // Muestra próximos mantenimientos

    function proximos\_mttos() {

        $head['url\_actual'] = current\_url();

        $datos['autos'] = $this->Modelo->getProxMttos();

        $datos['titulo'] = "Proximos Mantenimientos";

        $this->load->view('header', $head);

        $this->load->view('autos/proximos\_mttos',$datos);

    }

    // Muestra formulario para registrar revisión

    function registrar\_revision($auto) {

        $head['url\_actual'] = current\_url();

        $datos['iu'] = uniqid();

        $datos['auto'] = $auto;

        $res = $this->Modelo->getMotorPlacasKm($auto);

        $datos['combustible'] = $res->combustible;

        $datos['placas'] = $res->placas;

        $datos['kilometraje'] = $res->kilometraje;

        $this->load->view('header', $head);

        $this->load->view('autos/registrar\_revision',$datos);

    }

    // Muestra checklist de revisión en PDF

    function ver\_checklist($id) {

        $res = $this->Modelo->getChecklist($id);

        $pdf = $res->checklist;

        $datos['data'] = $pdf;

        $this->load->view('pdf', $datos);

    }

    // Muestra foto de hallazgo específico

    function hallazgo\_foto($id\_foto) {

        $photo = $this->Modelo->getHallazgoFoto($id\_foto);

        if($photo) {

            header("Content-type: image/png");

            echo $photo->foto;

        } else {

            echo "ERROR";

        }

    }

    // Muestra hallazgos de una revisión específica

    function hallazgos($id\_rev) {

        // Datos de la revisión

        $datosRevision = $this->Modelo->getRevision($id\_rev);

        $auto = $datosRevision->auto;

        $datos['rev\_id'] = $datosRevision->id;

        $datos['rev\_fecha'] = $datosRevision->fecha;

        $datos['rev\_User'] = $datosRevision->User;

        $datos['rev\_kilometraje'] = $datosRevision->kilometraje;

        $datos['rev\_combustible'] = $datosRevision->combustible;

        $datos['rev\_placas'] = $datosRevision->placas;

        $datos['rev\_vencimiento\_poliza'] = $datosRevision->vencimiento\_poliza;

        $datos['rev\_ecologico'] = $datosRevision->vencimiento\_ecologico;

        // Datos del vehículo

        $datosCoche = $this->Modelo->getAuto($auto);

        $datos['auto\_foto'] = $datosCoche->foto;

        $datos['auto\_marca'] = $datosCoche->marca;

        $datos['auto\_combustible'] = $datosCoche->combustible;

        $datos['auto\_placas'] = $datosCoche->placas;

        // Hallazgos encontrados

        $datos['carroceria'] = $this->Modelo->getHallazgosCarroceria($id\_rev);

        $datos['otros'] = $this->Modelo->getHallazgosOtros($id\_rev);

        $this->load->view('header');

        $this->load->view('autos/hallazgos', $datos);

    }

    // Guarda fotos temporales de hallazgos

    function temp\_photos() {

        $datos['iu'] = $this->input->post('iu');

        $datos['tipo'] = 'REVISION';

        $datos['usuario'] = $this->session->id;

        $datos['objeto'] = $this->input->post('auto');

        $datos['texto'] = $this->input->post('texto');

        $fotoB64 = $this->input->post('archivo');

        $datos['archivo'] = base64\_decode($fotoB64);

        $res = $this->Modelo->saveTempPhotos($datos);

        if($res > 0) {

            $respuesta = array('id' => $res, 'file' => $fotoB64);

            echo json\_encode($respuesta);

        }

    }

    // Elimina fotos temporales

    function delete\_temp\_photos() {

        $id\_temp = $this->input->post('id');

        $res = $this->Modelo->deleteTempPhotos($id\_temp);

        if($res) {

            echo "1";

        }

    }

    // Procesa el registro de una revisión completa

    function registrarRev() {

        $ACIERTOS = array();

        $ERRORES = array();

        $VARIABLES = array();

        // Datos básicos de la revisión

        $descripcion = $this->input->post('texto');

        $usuario = $this->session->nombre;

        $AUTO = $this->input->post('auto');

        $IU = $this->input->post('iu');

        // Datos del vehículo

        $KILOMETRAJE = $this->input->post('kilometraje');

        $COMBUSTIBLE = $this->input->post('combustible');

        $PLACAS = $this->input->post('placas');

        $timePoliza = strtotime($this->input->post('vencimientoPoliza'));

        $VENCIMIENTO\_POLIZA = date("Y-m-d", $timePoliza);

        $timeEco = strtotime($this->input->post('vencimientoEcologico'));

        $VENCIMIENTO\_ECOLOGICO = date("Y-m-d", $timeEco);

        // Recolección de variables de revisión

        $VARIABLES['Kilometraje'] = array('Kilometraje' => $this->input->post('kilometraje'), 'comentario' => $this->input->post('com\_kilometraje'));

        $VARIABLES['Aceite del motor'] = array('Aceite del motor' => $this->input->post('aceiteMotor'), 'comentario' => $this->input->post('com\_aceiteMotor'));

        $VARIABLES['Condiciones del aceite'] = array('Condiciones del aceite' => $this->input->post('condicionesAceite'), 'comentario' => $this->input->post('com\_condicionesAceite'));

        $VARIABLES['Limpia parabrisas delantero'] = array('Limpia parabrisas delantero' => $this->input->post('llpbDelantero'), 'comentario' => $this->input->post('com\_llpbDelantero'));

        $VARIABLES['Limpia parabrisas trasero'] = array('Limpia parabrisas trasero' => $this->input->post('llpbTrasero'), 'comentario' => $this->input->post('com\_llpbTrasero'));

        $VARIABLES['Deposito de refrigerante'] = array('Deposito de refrigerante' => $this->input->post('refrigeranteDeposito'), 'comentario' => $this->input->post('com\_refrigeranteDeposito'));

        $VARIABLES['Radiador'] = array('Radiador' => $this->input->post('refrigeranteRadiador'), 'comentario' => $this->input->post('com\_refrigeranteRadiador'));

        $VARIABLES['Liquido de frenos'] = array('Liquido de frenos' => $this->input->post('liquidoFrenos'), 'comentario' => $this->input->post('com\_liquidoFrenos'));

        $VARIABLES['Direccion hidraulica'] = array('Direccion hidraulica' => $this->input->post('direccionH'), 'comentario' => $this->input->post('com\_direccionH'));

        $VARIABLES['Direccional izquierda'] = array('Direccional izquierda' => $this->input->post('direccionalIzq'), 'comentario' => $this->input->post('com\_direccionalIzq'));

        $VARIABLES['Direccional derecha'] = array('Direccional derecha' => $this->input->post('direccionalDer'), 'comentario' => $this->input->post('com\_direccionalDer'));

        $VARIABLES['Lampara izquierda'] = array('Lampara izquierda' => $this->input->post('lamparaIzq'), 'comentario' => $this->input->post('com\_lamparaIzq'));

        $VARIABLES['Lampara derecha'] = array('Lampara derecha' => $this->input->post('lamparaDer'), 'comentario' => $this->input->post('com\_lamparaDer'));

        $VARIABLES['Lampara trasera izquierda'] = array('Lampara trasera izquierda' => $this->input->post('traseraIzq'), 'comentario' => $this->input->post('com\_traseraIzq'));

        $VARIABLES['Lampara trasera derecha'] = array('Lampara trasera derecha' => $this->input->post('traseraDer'), 'comentario' => $this->input->post('com\_traseraDer'));

        $VARIABLES['Lampara de emergencia'] = array('Lampara de emergencia' => $this->input->post('emergencia'), 'comentario' => $this->input->post('com\_emergencia'));

        $VARIABLES['Reversa'] = array('Reversa' => $this->input->post('reversa'), 'comentario' => $this->input->post('com\_reversa'));

        $VARIABLES['Tapiceria'] = array('Tapiceria' => $this->input->post('tapiceria'), 'comentario' => $this->input->post('com\_tapiceria'));

        $VARIABLES['Controles'] = array('Controles' => $this->input->post('controles'), 'comentario' => $this->input->post('com\_controles'));

        $VARIABLES['Kit de herramientas'] = array('Kit de herramientas' => $this->input->post('kitHerramientas'), 'comentario' => $this->input->post('com\_kitHerramientas'));

        $VARIABLES['Tapetes'] = array('Tapetes' => $this->input->post('tapetes'), 'comentario' => $this->input->post('com\_tapetes'));

        $VARIABLES['Placa delantera'] = array('Placa delantera' => $this->input->post('placaDelantera'), 'comentario' => $this->input->post('com\_placaDelantera'));

        $VARIABLES['Placa trasera'] = array('Placa trasera' => $this->input->post('placaTrasera'), 'comentario' => $this->input->post('com\_placaTrasera'));

        $VARIABLES['Tarjeta de combustible'] = array('Tarjeta de combustible' => $this->input->post('tarjetaCombustible'), 'comentario' => $this->input->post('com\_tarjetaCombustible'));

        $VARIABLES['Tarjeta de circulacion'] = array('Tarjeta de circulacion' => $this->input->post('tarjetaCirculacion'), 'comentario' => $this->input->post('com\_tarjetaCirculacion'));

        $VARIABLES['Poliza'] = array('Poliza' => $this->input->post('poliza'), 'comentario' => $this->input->post('com\_poliza'));

        // Identificación de fallas

        $FALLAS = array();

        foreach ($VARIABLES as $key => $value) {

            if($value[$key] == "NG") {

                $FALLAS[$key] = $value['comentario'];

            }

        }

        // Guarda en base de datos

        $datos = array(

            'auto' => $AUTO,

            'usuario' => $this->session->id,

            'kilometraje' => $KILOMETRAJE,

            'combustible' => $COMBUSTIBLE,

            'placas' => $PLACAS,

            'vencimiento\_poliza' => $VENCIMIENTO\_POLIZA,

            'vencimiento\_ecologico' => $VENCIMIENTO\_ECOLOGICO

        );

        $db\_response = $this->Modelo->saveChecklist($datos);

        if($db\_response['EXITO']) {

            // Actualiza datos del vehículo

            $AutoData = array(

                'kilometraje' => $KILOMETRAJE,

                'vencimiento\_poliza' => $VENCIMIENTO\_POLIZA,

                'vencimiento\_ecologico' => $VENCIMIENTO\_ECOLOGICO

            );

            $this->Modelo->updateAuto($AUTO, $AutoData);

            // Registra hallazgos si existen

            if(isset($descripcion)) {

                $arreglo = array('revision' => $db\_response['ID'], 'iu' => $IU);

                $this->Modelo->saveHallazgosCarroceria($arreglo);

            }

            // Registra fallas encontradas

            if(isset($FALLAS)) {

                foreach ($FALLAS as $key => $value) {

                    $arreglo = array('revision' => $db\_response['ID'], 'tipo' => $key, 'descripcion' => $value);

                    $this->Modelo->saveHallazgos($arreglo);

                }

                $acierto = array('titulo' => 'Registro de Revisión', 'detalle' => 'Se ha registrado Revisión con Éxito');

                array\_push($ACIERTOS, $acierto);

            }

            // Notifica si está próximo el mantenimiento

            $prox = $this->Modelo->getMotorPlacasKm($AUTO);

            if(($prox->kilometraje + 200) >= $prox->proximo\_mtto) {

                $Auto = $this->Modelo->getAuto($AUTO);

                $Resp = $this->Modelo->getResponsable($Auto->responsable);

                $data['auto'] = $Auto->fabricante . ' ' . $Auto->marca . ' ' . $Auto->modelo;

                $data['placas'] = $Auto->placas;

                $data['serie'] = $Auto->serie;

                $data['kmActual'] = number\_format($KILOMETRAJE);

                $data['proxMtto'] = number\_format($Auto->proximo\_mtto);

                if($Resp) {

                    $data['correoResponsable'] = $Resp->correo;

                    $this->correos->proxMtto200($data);

                }

            }

        } else {

            $error = array('titulo' => 'ERROR', 'detalle' => 'Error al registrar revisión');

            array\_push($ERRORES, $error);

        }

        $this->session->aciertos = $ACIERTOS;

        $this->session->errores = $ERRORES;

        redirect(base\_url('inicio'));

    }

    // Obtiene lista de vehículos para AJAX

    function ajax\_getAutos() {

        $res = $this->Conexion->consultar("SELECT id, marca, placas from autos order by marca");

        if($res) {

            echo json\_encode($res);

        }

    }

    // Obtiene eventos de vehículos para calendario

    function ajax\_getEventos() {

        $res = $this->Conexion->consultar("SELECT BA.id, BA.inicio as start, BA.final as end, BA.usuario, BA.usuarios, BA.auto, concat(A.placas, ' - ', A.marca) as title, BA.destino, BA.visita, BA.rsi, BA.comentarios, BA.equipo, BA.color from bitacora\_autos BA inner join autos A on A.id = BA.auto");

        if($res) {

            echo json\_encode($res);

        }

    }

    // Crea evento para vehículo (AJAX)

    function ajax\_crearEvento() {

        $data = json\_decode($this->input->post('data'));

        $data->usuario = $this->session->id;

        $f = array('fecha' => 'CURRENT\_TIMESTAMP()');

        $res = $this->Conexion->insertar('bitacora\_autos', $data, $f);

        echo $res;

    }

    // Elimina evento de vehículo (AJAX)

    function ajax\_borrarEvento() {

        $where['id'] = $this->input->post('id');

        if($this->Conexion->eliminar('bitacora\_autos', $where)) {

            echo "1";

        }

    }

    // Muestra foto de vehículo

    function photo($id) {

        $res = $this->Conexion->consultar("SELECT ifnull(A.foto, '') as Photo from autos A where A.id = $id", TRUE);

        header("Content-type: image/png");

        echo $res->Photo;

    }

    // Registra o actualiza GPS de vehículo

    function registrarGPS() {

        $auto = intval($this->input->post('idAuto'));

        $res = $this->Conexion->consultar("SELECT \* from gpsAutos where idAuto = ".$auto);

        $data = array(

            'idAuto' => $auto,

            'marca' => $this->input->post('marca'),

            'imei' => $this->input->post('imei'),

            'modelo' => $this->input->post('modelo'),

            'chip' => $this->input->post('chip'),

            'telefono' => $this->input->post('nchip'),

        );

        if($res) {

            $id\_inserted = $this->Modelo->updateGps($auto, $data);

        } else {

            $id\_inserted = $this->Modelo->registrar\_GPS($data);

        }

        redirect(base\_url('autos/alta\_gps/'.$auto));

    }

    // Registra nuevo vehículo

    function registrar() {

        $foto = file\_get\_contents($\_FILES['foto']['tmp\_name']);

        $poliza = file\_get\_contents($\_FILES['poliza']['tmp\_name']);

        $data = array(

            'serie' => $this->input->post('serie'),

            'fabricante' => $this->input->post('fabricante'),

            'marca' => $this->input->post('marca'),

            'modelo' => $this->input->post('modelo'),

            'combustible' => $this->input->post('combustible'),

            'kilometraje' => $this->input->post('kilometraje'),

            'placas' => $this->input->post('placas'),

            'no\_poliza' => $poliza,

            'vencimiento\_poliza' => $this->input->post('vencimiento\_poliza'),

            'vencimiento\_ecologico' => $this->input->post('vencimiento\_ecologico'),

            'tarjeta\_combustible' => $this->input->post('tarjeta\_combustible'),

            'foto' => $foto,

            'responsable' => $this->input->post('responsable'),

        );

        $id\_inserted = $this->Modelo->registrar\_auto($data);

        redirect(base\_url('autos'));

    }

    // Actualiza datos de vehículo

    function updateAuto() {

        $idauto = $this->input->post('idAuto');

        $data = array(

            'serie' => $this->input->post('serie'),

            'fabricante' => $this->input->post('fabricante'),

            'marca' => $this->input->post('marca'),

            'modelo' => $this->input->post('modelo'),

            'placas' => $this->input->post('placas'),

            'vencimiento\_ecologico' => $this->input->post('vencimiento\_ecologico'),

            'tarjeta\_combustible' => $this->input->post('tarjeta\_combustible'),

            'vencimiento\_poliza' => $this->input->post('vencimiento\_poliza'),

            'activo' => $this->input->post('activo'),

            'responsable' => $this->input->post('responsable'),

        );

        $this->Modelo->updateAuto($idauto, $data);

        redirect(base\_url('autos/'));

    }

    // Obtiene vehículos filtrados (AJAX)

    function ajaxgetAutos() {

        $activo = $this->input->post('activo');

        $parametro = $this->input->post('parametro');

        $texto = $this->input->post('texto');

        $query = "SELECT A.id, A.fabricante, A.marca, A.serie, A.placas, A.responsable, A.modelo, A.activo,ifnull(concat(U.nombre, ' ', U.paterno), 'N/A') as Responsable, (SELECT max(fecha) from autos\_revisiones where auto=A.id) as Ultrev, (SELECT max(id) from autos\_revisiones where auto=A.id) as IdUltrev FROM autos A LEFT JOIN usuarios U ON A.responsable = U.id where 1=1 ";

        if($activo == "0") {

            $query .= " and A.activo = '1'";

        } else {

            $query .= " and A.activo = '0'";

        }

        if(!empty($texto)) {

            if($parametro == 'placas') {

                $query .=" and A.placas = '". $texto ."'";

            } elseif($parametro == 'serie') {

                $query .= " and A.serie ='". $texto ."'";

            }

        }

        $res = $this->Conexion->consultar($query);

        if($res) {

            echo json\_encode($res);

        }

    }

    // Muestra archivo de póliza

    function getFile($id) {

        $row = $this->descargas\_model->getFile($id, 'autos');

        $file = $row->no\_poliza;

        header('Content-type: application/pdf');

        header('Content-Transfer-Encoding: binary');

        header('Accept-Ranges: bytes');

        echo $file;

    }

    // Sube foto de vehículo

    function uploadFoto() {

        $id = $this->input->post('id');

        $datos['foto'] = file\_get\_contents($\_FILES['file']['tmp\_name']);

        $this->Modelo->updateAuto($id,$datos);

    }

    // Sube póliza de vehículo

    function uploadPoliza() {

        $id = $this->input->post('id');

        $datos['no\_poliza'] = file\_get\_contents($\_FILES['file']['tmp\_name']);

        if(!$this->Modelo->updateAuto($id,$datos)) {

            trigger\_error("Error al subir archivo", E\_USER\_ERROR);

        }

    }

}

## Bitacora.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

// Controlador Bitacora para gestionar registros de software

class Bitacora extends CI\_Controller {

    // Constructor del controlador

    function \_\_construct() {

        parent::\_\_construct();

    }

    // Página principal: carga la vista de la bitácora de software

    public function index() {

        $this->load->view('header');                   // Carga el encabezado común

        $this->load->view('bitacora/software');        // Carga la vista específica de bitácora de software

    }

    // Obtiene los registros de la bitácora (vía Ajax)

    function ajax\_getLog() {

        $res = $this->Conexion->consultar("SELECT \* from bitacora\_software"); // Consulta todos los registros

        echo json\_encode($res);                                                // Devuelve los datos en formato JSON

    }

}

## Cafeteria.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

// Controlador Cafeteria para gestionar incidencias relacionadas con el área de cafetería

class Cafeteria extends CI\_Controller {

    // Constructor: carga modelos y librerías necesarias

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('cafeteria\_model','Modelo');   // Modelo principal para cafetería

        $this->load->library('correos');                  // Librería para enviar correos

        $this->load->library('AOS\_funciones');            // Librería adicional (propietaria)

    }

    // Carga la vista principal para generar tickets de cafetería

    public function generar() {

        $this->load->view('header');                      // Encabezado

        $this->load->view('cafeteria/cafeteria');         // Vista principal del formulario

        // $this->load->view('test/chat');                // Vista de prueba (comentada)

    }

    // Registra un nuevo ticket

    function registrar() {

        $data = array(

            'id\_user' => $this->session->id,                                      // Usuario que registra

            'tipo' => $this->input->post('opCategoria'),                          // Categoría seleccionada

            'titulo' => $this->input->post('titulo'),                             // Título del ticket

            'descripcion' => $this->input->post('descripcion'),                   // Descripción del problema

            'estatus' => 'ABIERTO',                                               // Estado inicial

            'cierre' => '0',                                                      // Aún no cerrado

            'fecha\_incidencia' => $this->input->post('fecha\_incidencia'),         // Fecha de la incidencia

        );

        $last\_id = $this->Modelo->crear\_comentario($data);                         // Guarda el ticket y obtiene ID

        // Datos para notificación por correo

        $datosCorreo['id'] = $last\_id;

        $datosCorreo['prefijo'] = substr($this->router->fetch\_class(), 8);

        $datosCorreo['titulo'] = $data['titulo'];

        $datosCorreo['fecha'] = date('d/m/Y h:i A');

        $datosCorreo['usuario'] = $this->session->nombre;

        $datosCorreo['correo'] = $this->session->correo;

        $datosCorreo['categoria'] = $this->input->post('opCategoria');

        $datosCorreo['fecha\_incidencia'] = $this->input->post('fecha\_incidencia');

        // $this->correos->comentarios\_cafeteria($datosCorreo);  // Envío de correo (comentado)

        redirect(base\_url('cafeteria/archivos/') . $last\_id);  // Redirecciona a la carga de archivos

    }

    // Carga vista para subir archivos relacionados a un ticket

    function archivos($id\_ticket) {

        $datos['id\_ticket'] = $id\_ticket;

        // $datos['controlador'] = 'tickets\_IT';                // Referencia a otro controlador (comentado)

        $this->load->view('header');

        $this->load->view('cafeteria/subir\_archivos', $datos);

    }

    // Procesa subida de archivos al ticket

    function subir\_archivos() {

        $idTicket = $this->input->post('id\_ticket');

        for ($i = 0; $i < count($\_FILES['file']['tmp\_name']); $i++) {

            $datos = array(

                'id\_comentario' => $idTicket,

                'nombre' => $\_FILES['file']['name'][$i],

                'archivo' => file\_get\_contents($\_FILES['file']['tmp\_name'][$i])

            );

            if (!$this->Modelo->subir\_archivos($datos)) {

                trigger\_error("Error al subir archivo", E\_USER\_ERROR);

            }

        }

    }

    // Muestra detalles del ticket, comentarios y archivos

    public function ver($id) {

        // $this->output->enable\_profiler(TRUE);                // Habilita profiler (comentado)

        $Renglon = $this->Modelo->verTicket($id);

        $datos['ticket'] = $Renglon->row();                    // Obtiene un solo renglón

        $datos['comentarios'] = $this->Modelo->verTicket\_comentarios($id);         // Comentarios

        $datos['comentarios\_fotos'] = $this->Modelo->verTicket\_comentarios\_fotos($id); // Fotos

        $datos['archivos'] = $this->Modelo->verTicketArchivos($id);               // Archivos adjuntos

        // $datos['controlador'] = $this->router->fetch\_class();                   // Controlador actual (comentado)

        $this->load->view('header');

        $this->load->view('cafeteria/ver', $datos);

    }

    // Muestra una imagen adjunta a un comentario

    function ver\_foto($id) {

        $photo = $this->Modelo->getFoto($id);

        if ($photo) {

            header("Content-type: image/png");

            echo $photo->archivo;

        } else {

            echo "ERROR";

        }

    }

    // Agrega un comentario a un ticket existente

    public function agregarComentario() {

        $idTicket = $this->input->post('idticket');

        $data = array(

            'id\_comentario' => $idTicket,

            'usuario' => $this->session->id,

            'comentario' => $this->input->post('comentario'),

        );

        $this->Modelo->agregar\_comentario($data);

        redirect(base\_url('cafeteria/ver/' . $idTicket));

    }

    // Cambia el estatus del ticket (ABIERTO, EN CURSO, etc.)

    public function estatus($idTicket, $estatus) {

        $Res = $this->Modelo->getUsuarioTicket($idTicket);

        switch ($estatus) {

            case '1': $Stat = 'ABIERTO'; break;

            case '2': $Stat = 'EN CURSO'; break;

            case '3': $Stat = 'DETENIDO'; break;

            case '4': $Stat = 'CANCELADO'; break;

            case '5': $Stat = 'SOLUCIONADO'; break;

            case '6': $Stat = 'CERRADO'; break;

            default:

                redirect(base\_url('inicio')); exit(); break;

        }

        $correo = $Res->correo;

        $usuario = $Res->User;

        $data = array('estatus' => $Stat);

        $this->Modelo->update($idTicket, $data);

        redirect(base\_url('cafeteria/ver/' . $idTicket));

    }

    // Muestra los tickets creados por el usuario actual

    public function mis\_tickets() {

        $datos['tickets'] = $this->Modelo->getMis\_tickets($this->session->id);

        $this->load->view('header');

        $this->load->view('cafeteria/mis\_tickets', $datos);

    }

    // Muestra todos los tickets disponibles para el usuario administrador

    public function administrar() {

        $datos['tickets'] = $this->Modelo->get\_tickets($this->session->id);

        $this->load->view('header');

        $this->load->view('cafeteria/tickets', $datos);

    }

    // Cierra un ticket desde una petición AJAX

    function ajax\_cerrarTicket() {

        $id = $this->input->post('id');

        $comentario = $this->input->post('comentario');

        $t->calificacion = $this->input->post('calificacion'); // Calificación del servicio

        $t->estatus = "CERRADO";                               // Marca como cerrado

        $this->Conexion->modificar('comentarios\_cafeteria', $t, null, array('id' => $id)); // Actualiza en BD

        if (!empty($comentario)) {

            $data = array(

                'ticket' => $id,

                'usuario' => $this->session->id,

                'comentario' => $comentario

            );

            $this->Modelo->agregar\_comentario($data); // Agrega comentario final

        }

    }

}

## Clientes.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

// Controlador Clientes para gestionar datos de empresas y contactos

class Clientes extends CI\_Controller {

    // Constructor: carga modelo de conexión a base de datos

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('conexion\_model', 'Conexion'); // Modelo de conexión a la base de datos

    }

    // Obtiene clientes desde la tabla 'empresas' (filtrados por ID o nombre) vía Ajax

    function ajax\_getClientes() {

        $id = $this->input->post('id');          // ID del cliente (opcional)

        $nombre = $this->input->post('nombre');  // Nombre del cliente (opcional)

        // Consulta base para obtener clientes

        $query = "SELECT id, nombre, razon\_social, rfc, calle, numero, numero\_interior, colonia,

                  credito\_cliente, credito\_cliente\_plazo, horario\_facturas,

                  ultimo\_dia\_facturas, documentos\_facturacion, codigo\_impresion

                  FROM empresas

                  WHERE cliente = 1";

        // Filtro por ID si se proporciona

        if ($id) {

            $query .= " AND id = '$id'";

        } else {

            // Filtro por nombre si se proporciona

            if ($nombre) {

                $query .= " AND nombre LIKE '%$nombre%'";

            }

        }

        $res = $this->Conexion->consultar($query, $id); // Ejecuta consulta

        if ($res) {

            echo json\_encode($res);                    // Devuelve resultados en JSON

        } else {

            echo "";                                   // Si no hay resultados, devuelve vacío

        }

    }

    // Obtiene contactos de empresas (por ID o por empresa) vía Ajax

    function ajax\_getContactos() {

        $id = $this->input->post('id');                // ID del contacto (opcional)

        $empresa = $this->input->post('id\_cliente');   // ID de la empresa (opcional)

        // Consulta base para obtener contactos activos

        $query = "SELECT \* FROM empresas\_contactos WHERE activo = 1";

        // Filtro por ID si se proporciona

        if ($id) {

            $query .= " AND id = '$id'";

        } else {

            // Filtro por empresa si se proporciona

            if ($empresa) {

                $query .= " AND empresa = '$empresa'";

            }

        }

        $res = $this->Conexion->consultar($query, $id); // Ejecuta consulta

        if ($res) {

            echo json\_encode($res);                     // Devuelve resultados en JSON

        }

    }

}

## Compras.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

// Controlador Compras para gestionar requisiciones, QR, y flujos relacionados

class Compras extends CI\_Controller {

// Constructor: carga modelos, librerías y helpers necesarios

function \_\_construct() {

parent::\_\_construct();

$this->load->model('compras\_model','Modelo'); // Modelo principal de compras

$this->load->model('MLConexion\_model', 'MLConexion'); // Modelo para conexión con ML

$this->load->model('descargas\_model'); // Modelo para descargas

$this->load->model('conexion\_model', 'Conexion'); // Modelo de conexión general

$this->load->helper('download'); // Helper para descargas

$this->load->library('correos'); // Librería de correos (general)

$this->load->library('correos\_pr'); // Librería de correos para PR

$this->load->library('AOS\_funciones'); // Funciones auxiliares AOS

}

// Carga vista para generar una nueva QR

function generar\_qr() {

$data['intervalo'] = $this->Modelo->intervalos(); // Obtiene intervalos disponibles

$this->load->view('header');

$this->load->view('compras/generar\_qr', $data); // Vista para generación de QR

}

// Muestra los detalles de un QR específico

function ver\_qr($id) {

$data['comentarios'] = $this->Modelo->verQr\_comentarios($id); // Comentarios del QR

$data['comentarios\_fotos'] = $this->Modelo->verQr\_comentarios\_fotos($id); // Fotos relacionadas

$data['conceptos'] = $this->Modelo->conceptos(); // Catálogo de conceptos

$data['qr'] = $this->Modelo->getDetalleQR($id); // Detalle del QR

// Consulta la fecha del rechazo si aplica

$fecha\_rechazo = $this->Conexion->consultar(

"SELECT b.fecha FROM requisiciones\_cotizacion qr

JOIN bitacora\_qrs b ON qr.id = b.qr

WHERE b.estatus = 'RECHAZADO' AND b.qr = " . $id, true

);

$data['fecha\_rechazo'] = $fecha\_rechazo;

// Define estilo del botón según estatus del QR

switch ($data['qr']->estatus) {

case 'ABIERTO':

$data['btn\_estatus'] = "btn-primary";

break;

case 'LIBERADO':

case 'COMPRA APROBADA':

$data['btn\_estatus'] = "btn-success";

break;

case 'COTIZANDO':

$data['btn\_estatus'] = "btn-warning";

break;

case 'CANCELADO':

$data['btn\_estatus'] = "btn-default";

break;

case 'RECHAZADO':

case 'COMPRA RECHAZADA':

$data['btn\_estatus'] = "btn-danger";

break;

default:

break;

}

$this->load->view('header');

$this->load->view('compras/ver\_qr', $data); // Vista para ver QR

}

// Carga la vista para editar un QR específico

function editar\_qr($id) {

$data['qr'] = $this->Modelo->getDetalleQR($id); // Datos del QR a editar

$data['intervalo'] = $this->Modelo->intervalos(); // Intervalos disponibles

// $this->load->view('debug', $data); // Para depuración (comentado)

$this->load->view('header');

$this->load->view('compras/editar\_qr', $data);

}

// Carga la vista para clonar un QR específico

function clonar\_qr($id) {

$data['qr'] = $this->Modelo->getDetalleQR($id); // Datos del QR original

$data['intervalo'] = $this->Modelo->intervalos(); // Intervalos disponibles

$this->load->view('header');

$this->load->view('compras/clonar\_qr', $data);

}

// Muestra listado de requisiciones filtradas por estatus o ID

function requisiciones($estatus = 'TODO', $id = "") {

$data['estatus'] = strtoupper($estatus); // Estatus seleccionado

$data['qr'] = $id; // ID del QR (si aplica)

$data['asignado'] = $this->Modelo->usuariosCompras(); // Usuarios del área de compras

$data['otros\_aprobadores'] = $data['estatus'] == 'TODO' ? '' : 'unchecked'; // Filtro de vista

$this->load->view('header');

$this->load->view('compras/catalogo\_qr', $data); // Vista del catálogo

}

// Muestra la vista con las requisiciones del usuario actual

function mis\_qrs() {

$this->load->view('header');

$this->load->view('compras/mis\_qrs');

}

// Genera una nueva QR con datos recibidos vía POST (JSON + atributos)

function ajax\_generarQR() {

$\_info = json\_decode($this->input->post('info')); // Información general en formato JSON

$\_atributos = $this->input->post('atributos'); // Atributos específicos

$unidad = "";

$clave\_unidad = "";

$intervalo = null;

// Si hay intervalo definido, se asigna

if (!empty($\_info->intervalo)) {

$intervalo = $\_info->intervalo;

}

// Si el tipo es 'PRODUCTO', se extraen unidad y clave, y se ignora el intervalo

if ($\_info->tipo == 'PRODUCTO') {

$unidad = $\_info->unidad;

$clave\_unidad = $\_info->clave\_unidad;

$intervalo = null;

}

// Preparación de datos para guardar la QR

$info = array(

'usuario' => $this->session->id,

'archivo' => "0",

'nombre\_archivo' => "",

'tipo' => $\_info->tipo,

'subtipo' => $\_info->subtipo,

'cantidad' => $\_info->cantidad,

'cantidad\_aprobada' => 0,

'unidad' => $unidad,

'clave\_unidad' => $clave\_unidad,

'descripcion' => $\_info->descripcion,

'prioridad' => $\_info->prioridad,

'lugar\_entrega' => $\_info->lugar\_entrega,

'comentarios' => $\_info->comentarios,

'critico' => $\_info->critico,

'destino' => $\_info->destino,

'atributos' => $\_atributos,

'notificaciones' => $\_info->notificaciones,

'estatus' => 'ABIERTO',

'especificos' => $\_info->especificos,

'intervalo' => $intervalo

);

// Si es tipo PRODUCTO y tiene tipo de calibración, se añade

if ($\_info->tipo == 'PRODUCTO' && isset($\_info->tipocalibracion)) {

$info['tipocalibracion'] = $\_info->tipocalibracion;

}

$res = $this->Modelo->generarQR($info); // Se crea la QR en la base de datos

// Registra movimiento en bitácora de QR

$bitacoraQR = array(

'qr' => intval($res),

'user' => $this->session->id,

'estatus' => 'ABIERTO'

);

$this->Modelo->estatusQR($bitacoraQR);

// Si la creación fue exitosa

if ($res) {

$datos = array(

'id' => $res,

'fecha' => date('d/m/Y h:i A'),

'usuario' => $this->session->nombre,

'cantidad' => $\_info->cantidad,

'unidad' => $unidad,

'descripcion' => $\_info->descripcion,

'atributos' => $\_atributos,

'prioridad' => $\_info->prioridad,

'comentarios' => $\_info->comentarios,

'correos' => array\_merge(

array($this->session->correo),

$this->Modelo->getCorreosQR()

)

);

// Solo se envía correo si la prioridad no es NORMAL

if ($\_info->prioridad != 'NORMAL') {

$this->correos->creacionQR($datos);

}

echo $res; // Devuelve el ID de la nueva QR

}

}

// Prueba: imprime lista de correos configurados para QR

function test() {

print\_r($this->Modelo->getCorreosQR());

}

// Edita una QR existente con nueva información recibida vía POST

function ajax\_editarQR() {

$\_info = json\_decode($this->input->post('info')); // Datos principales

$\_atributos = $this->input->post('atributos'); // Atributos editados

$unidad = "";

$clave\_unidad = "";

$intervalo = null;

$especificos = null;

// Registra edición en bitácora

$bitacoraQR = array(

'qr' => intval($\_info->id),

'user' => $this->session->id,

'estatus' => 'EDICION'

);

$this->Modelo->estatusQR($bitacoraQR);

// Asigna intervalo si existe

if (!empty($\_info->intervalo)) {

$intervalo = $\_info->intervalo;

}

// Asigna específicos si existen

if (!empty($\_info->especificos)) {

$especificos = $\_info->especificos;

}

// Si el tipo es PRODUCTO, se ajustan los valores y se ignora el intervalo

if ($\_info->tipo == 'PRODUCTO') {

$unidad = $\_info->unidad;

$clave\_unidad = $\_info->clave\_unidad;

$intervalo = null;

}

// Datos a actualizar en la QR

$info = array(

'id' => $\_info->id,

'usuario' => $this->session->id,

'tipo' => $\_info->tipo,

'subtipo' => $\_info->subtipo,

'cantidad' => $\_info->cantidad,

'unidad' => $unidad,

'clave\_unidad' => $clave\_unidad,

'descripcion' => $\_info->descripcion,

'prioridad' => $\_info->prioridad,

'lugar\_entrega' => $\_info->lugar\_entrega,

'comentarios' => $\_info->comentarios,

'critico' => $\_info->critico,

'destino' => $\_info->destino,

'atributos' => $\_atributos,

'notificaciones' => $\_info->notificaciones,

'estatus' => 'ABIERTO',

'especificos' => $especificos,

'intervalo' => $intervalo

);

// Si tiene tipo de calibración y es producto, lo añade

if ($\_info->tipo == 'PRODUCTO' && isset($\_info->tipocalibracion)) {

$info['tipocalibracion'] = $\_info->tipocalibracion;

}

// Si se escribió un comentario durante la edición, lo guarda

if ($\_info->coments) {

$coments = array(

'comentario' => $\_info->coments,

'usuario' => $this->session->id,

'qr' => $\_info->id

);

$this->Modelo->agregar\_comentario($coments);

}

$res = $this->Modelo->editarQR($info); // Ejecuta la actualización

// Si fue exitosa la edición

if ($res) {

$datos = array(

'id' => $\_info->id,

'fecha' => date('d/m/Y h:i A'),

'usuario' => $this->session->nombre,

'cantidad' => $\_info->cantidad,

'unidad' => $unidad,

'descripcion' => $\_info->descripcion,

'atributos' => $\_atributos,

'prioridad' => $\_info->prioridad,

'comentarios' => $\_info->comentarios,

'correos' => array\_merge(

array($this->session->correo),

$this->Modelo->getCorreosQR()

)

);

// Solo si la prioridad no es NORMAL se notifica por correo

if ($\_info->prioridad != 'NORMAL') {

$this->correos->edicionQR($datos);

}

echo "1"; // Confirma éxito

}

}

// Filtra y devuelve QRs según múltiples criterios recibidos vía POST (prioridad, tipo, estatus, fechas, texto, etc.)

function ajax\_getQRs() {

$prioridad = json\_decode($this->input->post('prioridad'));

$tipo = json\_decode($this->input->post('tipo'));

$estatus = $this->input->post('estatus');

$texto = $this->input->post('texto');

$parametro = $this->input->post('parametro');

$usuario = $this->input->post('usuario');

$asignado = $this->input->post('asignado');

$archivo = $this->input->post('archivo');

$fecha1 = $this->input->post('fecha1');

$fecha2 = $this->input->post('fecha2');

$f1 = strval($fecha1) . ' 00:00:00';

$f2 = strval($fecha2) . ' 23:59:59';

// Consulta base con JOIN para obtener nombre del usuario

$query = "SELECT R.id, R.fecha, R.usuario, R.prioridad, R.tipo, R.subtipo, R.cantidad,

R.cantidad\_aprobada, R.unidad, R.clave\_unidad, R.descripcion, R.atributos,

R.critico, R.destino, R.lugar\_entrega, R.comentarios, R.estatus, R.asignado,

CONCAT(U.nombre, ' ', U.paterno) AS User

FROM requisiciones\_cotizacion R

LEFT JOIN usuarios U ON R.usuario = U.id

WHERE 1 = 1";

// Filtros opcionales según campos recibidos

if ($estatus != 'TODO') {

$query .= " AND R.estatus = '$estatus'";

}

if (!empty($asignado)) {

$query .= " AND R.asignado = '$asignado'";

}

if ($usuario == '1') {

$idUser = $this->session->id;

$query .= " AND R.usuario = '$idUser'";

}

if (count($prioridad) > 0) {

$query .= " AND (1 = 0";

foreach ($prioridad as $value) {

$query .= " OR R.prioridad = '$value'";

}

$query .= ")";

}

if (isset($tipo) && count($tipo) > 0) {

$query .= " AND (1 = 0";

foreach ($tipo as $value) {

$query .= " OR R.tipo = '$value'";

}

$query .= ")";

}

// Control de acceso según privilegios para consumo interno o venta

if ($this->session->privilegios['crear\_qr\_interno'] != $this->session->privilegios['crear\_qr\_venta']) {

if ($this->session->privilegios['editar\_qr'] == "0" && $this->session->privilegios['liberar\_qr'] == "0") {

if ($this->session->privilegios['crear\_qr\_interno'] == "1") {

$query .= " AND R.destino = 'CONSUMO INTERNO'";

} else {

$query .= " AND R.destino = 'VENTA'";

}

}

}

// Filtro por texto libre según el parámetro seleccionado

if (!empty($texto)) {

switch ($parametro) {

case 'folio':

$query .= " AND R.id = '$texto'";

break;

case 'usuario':

$query .= " AND CONCAT(U.nombre, ' ', U.paterno) LIKE '%$texto%'";

break;

case 'contenido':

$query .= " AND (R.descripcion LIKE '%$texto%'

OR UPPER(R.atributos->'$.marca') LIKE UPPER('%$texto%')

OR UPPER(R.atributos->'$.modelo') LIKE UPPER('%$texto%'))";

break;

}

}

// Filtro por rango de fechas

if (!empty($fecha1) && !empty($fecha2)) {

$query .= " AND R.fecha BETWEEN '$f1' AND '$f2'";

}

// Si no es consulta de archivo histórico

if ($archivo != 1) {

$query .= " AND R.fecha > '2021-01-01 00:00:00'";

}

$query .= " ORDER BY R.fecha DESC";

// Ejecuta la consulta

$res = $this->Modelo->consulta($query);

if ($res) {

echo json\_encode($res);

} else {

echo "";

}

}

// Devuelve todas las QRs creadas por un usuario específico

function ajax\_getMisQRs() {

$user = $this->input->post('usuario');

$res = $this->Modelo->getMisQrs($user);

if ($res) {

echo json\_encode($res);

} else {

echo "";

}

}

// Devuelve los comentarios de una QR específica

function ajax\_getQRComentarios() {

$qr = $this->input->post('qr');

$res = $this->Conexion->consultar(

"SELECT C.\*, CONCAT(U.nombre, ' ', U.paterno, ' ', U.materno) AS User

FROM qr\_comentarios C

INNER JOIN usuarios U ON U.id = C.usuario

WHERE C.qr = $qr"

);

if ($res) {

echo json\_encode($res);

}

}

// Devuelve el detalle completo de una QR específica

function ajax\_getDetalleQR() {

$id = $this->input->post('idQR');

$res = $this->Modelo->getDetalleQR($id);

if ($res) {

echo json\_encode($res);

} else {

echo "";

}

}

// Devuelve usuarios con privilegios específicos para recibir notificaciones de QR

function ajax\_getUsuariosQRNotificaciones() {

$privilegio = $this->input->post('privilegio');

$id = $this->input->post('id');

// Consulta base

$query = "SELECT U.id, CONCAT(U.nombre, ' ', U.paterno) AS Nombre, P.puesto AS Puesto, U.correo

FROM usuarios U

INNER JOIN puestos P ON U.puesto = P.id

INNER JOIN privilegios PR ON PR.usuario = U.id

WHERE U.activo = 1";

// Filtro por ID o privilegio específico

if ($id) {

$query .= " AND U.id = '$id'";

} else {

$query .= " AND PR.$privilegio = 1";

}

$query .= " ORDER BY Nombre ASC";

$res = $this->Conexion->consultar($query, $id);

if ($res) {

echo json\_encode($res);

}

}

// Devuelve los proveedores asignados a una QR específica

function ajax\_getProveedoresAsignados() {

$idQR = $this->input->post('idQR');

// Consulta con información de empresa, propuesta y asignador

$query = "SELECT E.id, QP.id as idQP, P.entrega, E.nombre, QP.monto, QP.total, QP.moneda,

QP.tiempo\_entrega, QP.dias\_habiles, QP.comentarios, QP.nominado,

QP.seleccionado, QP.nombre\_archivo, QP.vencimiento, QP.fechaAsignacion,

CONCAT(U.nombre, ' ', U.paterno) AS Asignador

FROM qr\_proveedores QP

INNER JOIN empresas E ON E.id = QP.empresa

INNER JOIN proveedores P ON P.empresa = E.id

LEFT JOIN usuarios U ON U.id = QP.asignador

WHERE QP.qr = '$idQR'";

$res = $this->Modelo->consulta($query);

if ($res) {

echo json\_encode($res);

} else {

echo "";

}

}

// Devuelve las propuestas enviadas por proveedores para una QR específica

function ajax\_getPropuestas() {

$idQR = $this->input->post('idQR');

$query = "SELECT E.id, QP.id as idQP, P.entrega, P.rma\_requerido, E.nombre, QP.monto, QP.total,

QP.moneda, QP.tiempo\_entrega, QP.dias\_habiles, QP.comentarios, QP.nominado,

QP.seleccionado, QP.nombre\_archivo, QP.vencimiento, QR.cantidad, QR.descripcion,

QR.tipo, QR.subtipo, IFNULL(JSON\_UNQUOTE(atributos->'$.serie'), '') AS Serie,

QR.nombre\_archivoEjemplo, QR.id as QR

FROM qr\_proveedores QP

INNER JOIN requisiciones\_cotizacion QR ON QR.id = QP.qr

INNER JOIN empresas E ON E.id = QP.empresa

INNER JOIN proveedores P ON P.empresa = E.id

WHERE QP.qr = '$idQR'";

$res = $this->Modelo->consulta($query);

if ($res) {

echo json\_encode($res);

} else {

echo "";

}

}

// Asigna un proveedor a una QR (registro inicial con valores por defecto)

function ajax\_setProveedor() {

$data['qr'] = $this->input->post('idQR');

date\_default\_timezone\_set('America/Chihuahua');

$data['fechaAsignacion'] = date('Y-m-d h:i:s');

$data['empresa'] = $this->input->post('idProv');

$data['total'] = "0";

$data['dias\_habiles'] = "1";

$data['nominado'] = "0";

$data['seleccionado'] = "0";

$data['comentarios'] = "";

$data['factor'] = "0";

$data['asignador'] = $this->session->id;

$res = $this->Modelo->setProveedor($data);

if ($res) {

echo "1";

}

}

// Marca un proveedor como el seleccionado para una QR

function ajax\_setProveedorSugerido() {

$qr = $this->input->post('qr');

$qr\_prov = $this->input->post('qr\_prov');

// Primero desmarca todos los proveedores de esa QR

$query = "UPDATE qr\_proveedores SET seleccionado = '0' WHERE qr = '$qr'";

$this->Modelo->update($query);

// Luego marca como seleccionado al proveedor indicado

$query2 = "UPDATE qr\_proveedores SET seleccionado = '1' WHERE id = '$qr\_prov'";

$this->Modelo->update($query2);

echo "1";

}

// Elimina un proveedor asignado a una QR

function ajax\_eliminarProveedor() {

$data['qr'] = $this->input->post('qr');

$data['empresa'] = $this->input->post('empresa');

$res = $this->Modelo->deleteProveedor($data);

if ($res) {

echo "1";

}

}

// Guarda información actualizada de todos los proveedores de una QR

function guardarProveedores() {

$datos = json\_decode($this->input->post('datos'), TRUE);

// Se actualiza cada proveedor uno por uno

foreach ($datos as $value) {

$this->Modelo->updateProveedor($value);

}

$id = $datos[0]["id"];

// Se actualiza el campo de máximo vencimiento en la tabla de requisiciones

$this->Conexion->comando(

"UPDATE requisiciones\_cotizacion

SET maximo\_vencimiento = (

SELECT MAX(vencimiento)

FROM qr\_proveedores

WHERE qr = (

SELECT qr

FROM qr\_proveedores

WHERE id = $id

)

)"

);

echo "1";

}

// Sugiere proveedores según coincidencia en palabras clave (tags)

function proveedoresSugeridos() {

$tags = $this->input->post('tags');

$arreglo = explode(" ", trim($tags));

$arreglo = array\_diff($arreglo, array("")); // Elimina vacíos

// Consulta proveedores con coincidencias en tags

$query = "SELECT E.id, E.nombre

FROM empresas E

INNER JOIN proveedores P ON P.empresa = E.id

WHERE E.proveedor = 1 AND (1 != 1";

foreach ($arreglo as $value) {

$query .= " OR tags LIKE '%," . $value . ",%'";

}

$query .= ")";

$res = $this->Modelo->consulta($query);

if ($res) {

echo json\_encode($res);

} else {

echo "";

}

}

// Sugiere proveedores según coincidencia con la marca en atributos

function proveedoresSugeridosMarca() {

$tags = $this->input->post('tags');

$query = "SELECT E.id, E.nombre, IFNULL((

SELECT COUNT(QP.id)

FROM qr\_proveedores QP

INNER JOIN requisiciones\_cotizacion QR ON QR.id = QP.qr

WHERE QP.monto > 0

AND QP.empresa = E.id

AND UPPER(QR.atributos->'$.marca') = UPPER('\"$tags\"')

), 0) AS QtyQr

FROM empresas E

INNER JOIN proveedores P ON P.empresa = E.id

WHERE E.proveedor = 1

AND (1 != 1 OR tags LIKE '%,$tags,%')

AND !ISNULL(P.entrega)

AND JSON\_LENGTH(P.tipo) > 0

AND JSON\_LENGTH(P.formas\_pago) > 0";

$query .= " UNION ";

$query .= "SELECT E.id, E.nombre, IFNULL((

SELECT COUNT(QP.id)

FROM qr\_proveedores QP

INNER JOIN requisiciones\_cotizacion QR ON QR.id = QP.qr

WHERE QP.monto > 0

AND QP.empresa = E.id

AND UPPER(QR.atributos->'$.marca') = UPPER('\"$tags\"')

), 0) AS QtyQr

FROM qr\_proveedores QP

INNER JOIN requisiciones\_cotizacion QR ON QP.qr = QR.id

INNER JOIN empresas E ON E.id = QP.empresa

WHERE E.proveedor = 1

AND QP.monto > 0

AND UPPER(QR.atributos->'$.marca') = UPPER('\"$tags\"')";

$res = $this->Modelo->consulta($query);

echo $res ? json\_encode($res) : "";

}

// Sugiere proveedores según coincidencia con modelo en atributos

function proveedoresSugeridosModelo() {

$tags = $this->input->post('tags');

$query = "SELECT E.id, E.nombre, IFNULL((

SELECT COUNT(QP.id)

FROM qr\_proveedores QP

INNER JOIN requisiciones\_cotizacion QR ON QR.id = QP.qr

WHERE QP.monto > 0

AND QP.empresa = E.id

AND UPPER(QR.atributos->'$.modelo') = UPPER('\"$tags\"')

), 0) AS QtyQr

FROM qr\_proveedores QP

INNER JOIN requisiciones\_cotizacion QR ON QP.qr = QR.id

INNER JOIN empresas E ON E.id = QP.empresa

WHERE E.proveedor = 1

AND QP.monto > 0

AND UPPER(QR.atributos->'$.modelo') = UPPER('\"$tags\"')";

$res = $this->Modelo->consulta($query);

echo $res ? json\_encode($res) : "";

}

// Obtiene información detallada de un proveedor asignado a una QR

function ajax\_getProveedor() {

$id = $this->input->post('id');

$query = "SELECT QP.id, QP.qr, QP.empresa, QP.costos, QP.monto, QP.moneda, QP.total,

QP.tiempo\_entrega, QP.dias\_habiles, QP.comentarios, QP.nominado,

QP.seleccionado, QP.vencimiento, QP.nombre\_archivo, E.nombre,

P.entrega, QR.nombre\_archivoEjemplo

FROM qr\_proveedores QP

INNER JOIN empresas E ON E.id = QP.empresa

INNER JOIN proveedores P ON P.empresa = E.id

JOIN requisiciones\_cotizacion QR ON QR.id = QP.qr

WHERE QP.id = '$id'";

$res = $this->Modelo->consulta($query, TRUE);

echo $res ? json\_encode($res) : "";

}

// Obtiene lista de proveedores filtrando por texto y condiciones necesarias

function ajax\_getProveedores() {

$texto = $this->input->post('texto');

$query = "SELECT E.\*

FROM empresas E

INNER JOIN proveedores P ON E.id = P.empresa

WHERE E.proveedor = 1

AND (P.tags LIKE '%,$texto,%' OR E.nombre LIKE '%$texto%')

AND !ISNULL(P.entrega)

AND JSON\_LENGTH(P.tipo) > 0

AND JSON\_LENGTH(P.formas\_pago) > 0";

$res = $this->Modelo->consulta($query);

echo $res ? json\_encode($res) : "";

}

// Cambia el estatus de una QR, registra en bitácora y envía correo si aplica

function ajax\_setEstatusQR() {

$estatus = $this->input->post('estatus');

$idqr = $this->input->post('idqr');

$liberador = $this->session->id;

if ($estatus == "LIBERADO") {

$query = "UPDATE requisiciones\_cotizacion

SET estatus = '$estatus',

fecha\_liberacion = CURRENT\_TIMESTAMP(),

liberador = $liberador

WHERE id = '$idqr'";

} else {

$query = "UPDATE requisiciones\_cotizacion

SET estatus = '$estatus'

WHERE id = '$idqr'";

}

$this->Modelo->update($query);

// Registra cambio en bitácora

$bitacoraQR = array(

'qr' => intval($idqr),

'user' => $this->session->id,

'estatus' => $estatus

);

$this->Modelo->estatusQR($bitacoraQR);

// Si fue exitoso, enviar correo si es necesario

if (in\_array($estatus, ["LIBERADO", "COMPRA APROBADA"])) {

$qr = $this->Modelo->getDetalleQR($idqr);

$datos = array(

'id' => $idqr,

'fecha' => date('d/m/Y h:i A'),

'usuario' => $qr->User,

'cliente' => $qr->Client,

'prioridad' => $qr->prioridad,

'unidad' => $qr->unidad,

'cantidad' => $qr->cantidad,

'descripcion' => $qr->descripcion,

'atributos' => $qr->atributos,

'comentarios' => $qr->comentarios,

'correos' => array($qr->correo),

'estatus' => $estatus

);

// Añade notificados adicionales

$Notificar = json\_decode($qr->notificaciones);

$query = "SELECT U.correo FROM usuarios U WHERE 1 != 1";

foreach ($Notificar as $value) {

$query .= " OR U.id = $value";

}

$res = $this->Conexion->consultar($query);

foreach ($res as $value) {

array\_push($datos['correos'], $value->correo);

}

$this->correos->liberarQR($datos);

}

echo "1";

}

// Cambia el estatus de una QR e incluye un comentario con notificación por correo

function ajax\_setEstatusMsjQR() {

$idqr = $this->input->post('idqr');

$estatus = $this->input->post('estatus');

$comentario\_original = $this->input->post('comentario');

$comentario = "<b><font color='red'>$estatus:</font></b> " . $comentario\_original;

$tags = $this->input->post('txtTags');

$correos = explode(",", $tags);

$query = "UPDATE requisiciones\_cotizacion SET estatus = '$estatus' WHERE id = '$idqr'";

$this->Modelo->update($query);

// Bitácora

$bitacoraQR = array(

'qr' => intval($idqr),

'user' => $this->session->id,

'estatus' => $estatus

);

$this->Modelo->estatusQR($bitacoraQR);

// Guarda el comentario del cambio de estatus

$data = array(

'qr' => $idqr,

'usuario' => $this->session->id,

'comentario' => $comentario

);

$this->Modelo->agregar\_comentario($data);

$qr = $this->Modelo->getDetalleQR($idqr);

$datos = array(

'id' => $idqr,

'fecha' => date('d/m/Y h:i A'),

'usuario' => $qr->User,

'cliente' => $qr->Client,

'prioridad' => $qr->prioridad,

'unidad' => $qr->unidad,

'cantidad' => $qr->cantidad,

'descripcion' => $qr->descripcion,

'atributos' => $qr->atributos,

'comentarios' => $comentario\_original,

'correos' => array($qr->correo)

);

// Envía correo según tipo de estatus

if ($estatus == "RECHAZADO" || $estatus == "COMPRA RECHAZADA") {

$this->correos->rechazoQR($datos);

} else if ($estatus == "LIBERADO") {

$this->correos->liberarQR($datos);

}

// Si se agregaron correos adicionales manualmente

if (count($correos) > 0) {

$datos2 = array(

'id' => $idqr,

'comentario' => $comentario,

'correos' => $correos

);

$this->correos->comentarioQR($datos2);

}

echo $comentario;

}

// Marca uno o varios proveedores como nominados para una QR

function ajax\_setProveedoresNominados() {

$res = TRUE;

$qr\_proveedor = json\_decode($this->input->post('qr\_proveedores')); // IDs de proveedores a nominar

$qr = $this->input->post('qr'); // ID de la QR

// Primero se desmarcan todos

$this->Modelo->update("UPDATE qr\_proveedores SET nominado = 0 WHERE qr = $qr");

// Luego se marcan los seleccionados

foreach ($qr\_proveedor as $value) {

if (!$this->Modelo->update("UPDATE qr\_proveedores SET nominado = 1 WHERE id = $value")) {

$res = FALSE;

}

}

echo $res ? "1" : "";

}

// Sube archivo PDF asociado a una QR (como archivo principal)

function ajax\_subirArchivoQR() {

$datos['id'] = $this->input->post('qr');

$datos['archivo'] = file\_get\_contents($\_FILES['file']['tmp\_name']);

$datos['nombre\_archivo'] = str\_pad($datos['id'], 6, "0", STR\_PAD\_LEFT) . ".pdf";

if (!$this->Modelo->setQRFile($datos)) {

trigger\_error("Error al subir archivo", E\_USER\_ERROR);

} else {

echo "1";

}

}

// Elimina el archivo PDF asociado a una QR

function ajax\_borrarArchivoQR() {

$datos['id'] = $this->input->post('qr');

$datos['archivo'] = "";

$datos['nombre\_archivo'] = "";

if (!$this->Modelo->setQRFile($datos)) {

trigger\_error("Error al borrar archivo", E\_USER\_ERROR);

} else {

echo "1";

}

}

// Sube archivo de evidencia (como propuesta de proveedor)

function ajax\_subirEvidencia() {

$datos['id'] = $this->input->post('qr\_prov');

$datos['archivo'] = file\_get\_contents($\_FILES['file']['tmp\_name']);

$datos['nombre\_archivo'] = $\_FILES['file']['name'];

if (!$this->Modelo->updateProveedor($datos)) {

trigger\_error("Error al subir evidencia", E\_USER\_ERROR);

} else {

echo $datos['nombre\_archivo'];

}

}

// Elimina archivo de evidencia cargado por proveedor

function ajax\_eliminarEvidencia() {

$datos['id'] = $this->input->post('qr\_prov');

$datos['archivo'] = null;

$datos['nombre\_archivo'] = null;

if (!$this->Modelo->updateProveedor($datos)) {

trigger\_error("Error al eliminar evidencia", E\_USER\_ERROR);

} else {

echo "1";

}

}

// Muestra un archivo de evidencia en el navegador (inline PDF)

function getEvidencia($qr\_prov) {

$row = $this->descargas\_model->getFile($qr\_prov, 'qr\_proveedores');

$file = $row->archivo;

$nombre = $row->nombre\_archivo;

header('Content-type: application/pdf');

header('Content-Disposition: inline; filename="' . $nombre . '"');

header('Content-Transfer-Encoding: binary');

header('Accept-Ranges: bytes');

echo $file;

}

// Muestra el archivo principal PDF de una QR

function getQrFile($qr) {

$row = $this->descargas\_model->getFile($qr, 'requisiciones\_cotizacion');

$file = $row->archivo;

$nombre = $row->nombre\_archivo;

header('Content-type: application/pdf');

header('Content-Disposition: inline; filename="' . $nombre . '"');

header('Content-Transfer-Encoding: binary');

header('Accept-Ranges: bytes');

echo $file;

}

// Agrega un comentario a una QR y notifica a usuarios relacionados

function agregarComentario() {

$idQr = $this->input->post('idQr');

$comentario = $this->input->post('comentario');

$tags = $this->input->post('txtTags');

$correos = explode(",", $tags);

// Obtiene correos del solicitante y del asignado

$query = "SELECT ur.correo AS correoReq, ua.correo AS correoA

FROM requisiciones\_cotizacion qr

JOIN usuarios ur ON ur.id = qr.usuario

LEFT JOIN usuarios ua ON ua.id = qr.asignado

WHERE qr.id = $idQr";

$res = $this->Modelo->consulta($query, TRUE);

$mails = explode(',', $res->correoReq . ',' . $res->correoA);

// Enviar correo principal de comentario

$correoData = array(

'id' => $idQr,

'comentario' => $comentario,

'correos' => $mails,

'usuario' => $this->session->nombre

);

$this->correos->comentarioQR($correoData);

// Guarda comentario en base de datos

$data = array(

'qr' => $idQr,

'usuario' => $this->session->id,

'comentario' => $comentario

);

$this->Modelo->agregar\_comentario($data);

// Enviar notificaciones a correos adicionales si existen

if (count($correos) > 0) {

$datos = array(

'id' => $idQr,

'comentario' => $comentario,

'correos' => $correos,

'usuario' => $this->session->nombre

);

$this->correos->comentarioQR($datos);

}

redirect(base\_url('compras/ver\_qr/' . $idQr));

}

// Obtiene resumen de propuestas por marca para un proveedor específico

function ajax\_getResumenMarca() {

$prov = $this->input->post('prov');

$marca = $this->input->post('marca');

$query = "SELECT QP.id AS idQP, QP.qr, QR.descripcion, QP.total, QP.moneda,

QP.tiempo\_entrega, QP.dias\_habiles, QP.nombre\_archivo, QP.vencimiento

FROM qr\_proveedores QP

INNER JOIN requisiciones\_cotizacion QR ON QR.id = QP.qr

WHERE QP.monto > 0

AND QP.empresa = '$prov'

AND UPPER(QR.atributos->'$.marca') = UPPER('\"$marca\"')";

$res = $this->Modelo->consulta($query);

echo $res ? json\_encode($res) : "";

}

// Obtiene resumen de propuestas por modelo para un proveedor específico

function ajax\_getResumenModelo() {

$prov = $this->input->post('prov');

$modelo = $this->input->post('modelo');

$query = "SELECT QP.id AS idQP, QP.qr, QR.descripcion, QP.total, QP.moneda,

QP.tiempo\_entrega, QP.dias\_habiles, QP.nombre\_archivo, QP.vencimiento

FROM qr\_proveedores QP

INNER JOIN requisiciones\_cotizacion QR ON QR.id = QP.qr

WHERE QP.monto > 0

AND QP.empresa = '$prov'

AND UPPER(QR.atributos->'$.modelo') = UPPER('\"$modelo\"')";

$res = $this->Modelo->consulta($query);

echo $res ? json\_encode($res) : "";

}

// Consulta PRs activas relacionadas a una QR específica

function ajax\_getPRS\_QR() {

$idQR = $this->input->post('id\_qr');

$query = "SELECT PR.id, PR.fecha, PR.estatus, CONCAT(U.nombre, ' ', U.paterno) AS User

FROM prs PR

INNER JOIN usuarios U ON PR.usuario = U.id

WHERE (PR.estatus != 'CANCELADO' AND PR.estatus != 'CERRADO')

AND PR.qr = $idQR";

$res = $this->Conexion->consultar($query);

echo $res ? json\_encode($res) : "";

}

// Carga el dashboard general de compras

function dashboard() {

$data['asignado'] = $this->Modelo->usuariosCompras(); // Usuarios del área de compras

$this->load->view('header');

$this->load->view('compras/dashboard', $data);

}

// Genera reporte resumen de QRs activas para el dashboard

function ajax\_getReporteQR() {

$asignado = $this->input->post('asignado');

$us = ($asignado != 'TODO') ? " AND asignado = $asignado" : "";

$query = "SELECT

COUNT(\*) AS Total,

(SELECT COUNT(\*) FROM requisiciones\_cotizacion WHERE estatus = 'ABIERTO' AND fecha > '2021-01-01 00:00:00' $us) AS Abiertos,

(SELECT MIN(fecha) FROM requisiciones\_cotizacion WHERE estatus = 'ABIERTO' AND fecha > '2021-01-01 00:00:00' $us) AS ultAbiertos,

(SELECT COUNT(\*) FROM requisiciones\_cotizacion WHERE estatus = 'RECHAZADO' AND fecha > '2021-01-01 00:00:00' $us) AS Rechazados,

(SELECT MIN(fecha) FROM requisiciones\_cotizacion WHERE estatus = 'RECHAZADO' AND fecha > '2021-01-01 00:00:00' $us) AS ultRechazados,

(SELECT COUNT(\*) FROM requisiciones\_cotizacion WHERE estatus = 'COTIZANDO' AND fecha > '2021-01-01 00:00:00' $us) AS Cotizando,

(SELECT MIN(fecha) FROM requisiciones\_cotizacion WHERE estatus = 'COTIZANDO' AND fecha > '2021-01-01 00:00:00' $us) AS ultCotizando

FROM requisiciones\_cotizacion

WHERE fecha > '2021-01-01 00:00:00' $us";

$res = $this->Conexion->consultar($query, TRUE);

echo json\_encode($res);

}

// Genera reporte resumen de PRs activas para el dashboard

function ajax\_getReportePR() {

$query = "SELECT

COUNT(\*) AS Total,

(SELECT COUNT(\*) FROM prs WHERE estatus = 'PENDIENTE') AS Pendientes,

(SELECT MIN(fecha) FROM prs WHERE estatus = 'PENDIENTE') AS ultPendientes,

(SELECT COUNT(\*) FROM prs WHERE estatus = 'APROBADO') AS Aprobados,

(SELECT MIN(fecha) FROM prs WHERE estatus = 'APROBADO') AS ultAprobados,

(SELECT COUNT(\*) FROM prs WHERE estatus = 'RECHAZADO') AS Rechazados,

(SELECT MIN(fecha) FROM prs WHERE estatus = 'RECHAZADO') AS ultRechazados,

(SELECT COUNT(\*) FROM prs WHERE estatus = 'EN SELECCION') AS Seleccion,

(SELECT MIN(fecha) FROM prs WHERE estatus = 'EN SELECCION') AS ultSeleccion,

(SELECT COUNT(\*) FROM prs WHERE estatus = 'PO AUTORIZADA') AS PO\_Autorizada,

(SELECT MIN(fecha) FROM prs WHERE estatus = 'PO AUTORIZADA') AS ultPO\_Autorizada,

(SELECT COUNT(\*) FROM prs WHERE estatus = 'EN PO') AS En\_PO,

(SELECT MIN(fecha) FROM prs WHERE estatus = 'EN PO') AS ultEn\_PO,

(SELECT COUNT(\*) FROM prs WHERE estatus = 'POR RECIBIR') AS Por\_Recibir,

(SELECT MIN(fecha) FROM prs WHERE estatus = 'POR RECIBIR') AS ultPor\_Recibir

FROM prs";

$res = $this->Conexion->consultar($query, TRUE);

echo json\_encode($res);

}

// Reporte de órdenes de compra por estatus

function ajax\_getReportePO() {

$requisitor = $this->input->post('requisitor');

$us = ($requisitor != 'TODO') ? " and usuario = $requisitor" : "";

$query = 'SELECT count(\*) as Total, ';

// Subconsultas para contar y obtener la fecha más antigua de cada estatus

$estados = [

'EN PROCESO' => 'EnProceso',

'PENDIENTE AUTORIZACION' => 'PendienteAutorizacion',

'AUTORIZADA' => 'Autorizada',

'RECHAZADA' => 'Rechazada',

'ORDENADA' => 'Ordenada',

'RECIBIDA' => 'Recibida',

'RECIBIDA PARCIAL' => 'Parcial',

'RECIBIDA TOTAL' => 'RecibidaTotal',

'LISTA PARA CERRAR' => 'Lista'

];

foreach ($estados as $estatus => $alias) {

$query .= "(SELECT count(\*) FROM ordenes\_compra WHERE estatus = \"$estatus\" AND publish = 1 AND fecha > \"2022-01-01 00:00:00\"$us) AS $alias, ";

$query .= "(SELECT MIN(fecha) FROM ordenes\_compra WHERE estatus = \"$estatus\" AND publish = 1 AND fecha > \"2022-01-01 00:00:00\"$us) AS ult$alias, ";

}

// Remueve la última coma

$query = rtrim($query, ', ');

// Consulta principal

$query .= " FROM ordenes\_compra WHERE publish = 1 AND fecha > \"2022-01-01 00:00:00\"$us";

$res = $this->Conexion->consultar($query, TRUE);

echo json\_encode($res);

}

// Carga vista para solicitudes de compra (PR)

function solicitudes\_compra($estatus = 'TODO') {

$estatus = strtoupper($estatus);

$data['estatus'] = str\_replace('\_', ' ', $estatus);

$data['otros\_aprobadores'] = ($data['estatus'] == 'TODO') ? '' : 'checked';

$this->load->view('header');

$this->load->view('compras/catalogo\_pr', $data);

}

// Vista detallada de una PR

function ver\_pr($id) {

$data['comentarios'] = $this->Modelo->verPr\_comentarios($id);

$data['comentarios\_fotos'] = $this->Modelo->verPr\_comentarios\_fotos($id);

$data['surtidorPR'] = $this->Modelo->surtidorPR($id);

$data['atributos'] = $this->Modelo->atributosPr($id);

$data['pr'] = $this->Modelo->getPR($id);

// Determina color de botón según estatus

switch ($data['pr']->estatus) {

case 'APROBADO':

case 'PO AUTORIZADA':

$data['btn\_estatus'] = "btn-success"; break;

case 'PENDIENTE':

case 'EN SELECCION':

case 'POR RECIBIR':

$data['btn\_estatus'] = "btn-warning"; break;

case 'EN PO':

case 'CERRADO':

$data['btn\_estatus'] = "btn-primary"; break;

case 'RECHAZADO':

$data['btn\_estatus'] = "btn-danger"; break;

case 'PROCESADO':

case 'CANCELADO':

$data['btn\_estatus'] = "btn-default"; break;

default:

$data['btn\_estatus'] = "btn-warning"; break;

}

$this->load->view('header');

$this->load->view('compras/ver\_pr', $data);

}

// Vista de PRs del usuario actual

function mis\_prs() {

$this->load->view('header');

$this->load->view('compras/mis\_prs');

}

// Genera una nueva PR desde una QR

function ajax\_generarPR() {

$this->load->model('compras\_model');

$qrt = $this->input->post('qr');

$qty = $this->input->post('qty');

$precio = $this->input->post('precio');

$descripcion = $this->input->post('descripcion');

$serie = $this->input->post('serie');

$qr\_prov = $this->input->post('qr\_prov');

$item = $this->input->post('item');

$id = $this->input->post('id');

$qr = $this->Modelo->getDetalleQR($qrt);

$qr\_prov = $this->Modelo->getQRProv($qr\_prov);

// Modifica atributos si existe clave 'serie'

$att = json\_decode($qr->atributos, TRUE);

if (array\_key\_exists('serie', $att)) {

$att['serie'] = $serie ?: 'N/A';

if ($id) $att['id'] = $id;

if ($item) $att['item'] = $item;

}

// Datos para la nueva PR

$datos = [

'qr' => $qr->id,

'qr\_proveedor' => $qr\_prov->id,

'usuario' => $this->session->id,

'prioridad' => $qr->prioridad,

'tipo' => $qr->tipo,

'subtipo' => $qr->subtipo,

'cantidad' => $qty,

'precio\_unitario' => $qr\_prov->monto,

'importe' => $qr\_prov->monto \* $qty,

'moneda' => $qr\_prov->moneda,

'unidad' => $qr->unidad,

'clave\_unidad' => $qr->clave\_unidad,

'descripcion' => $descripcion,

'atributos' => json\_encode($att),

'critico' => $qr->critico,

'destino' => $qr->destino,

'lugar\_entrega' => $qr->lugar\_entrega,

'comentarios' => "",

'estatus' => "PENDIENTE"

];

$funciones['fecha'] = 'CURRENT\_TIMESTAMP()';

$res = $this->Conexion->insertar('prs', $datos, $funciones);

// Guarda bitácora de estatus inicial

$this->compras\_model->estatusPR([

'pr' => intval($res),

'user' => $this->session->id,

'estatus' => 'PENDIENTE'

]);

// Si es SERVICIO, transfiere atributos temporales

if ($qr->tipo == "SERVICIO" || ($qr->tipo == "PRODUCTO" && $qr->destino == "VENTA")) {

$result = $this->Conexion->consultar("SELECT \* FROM qr\_atributos\_temp WHERE idQr = $qrt");

if ($result) {

foreach ($result as $elem) {

$atributosPR = [

'idPr' => intval($res),

'item' => $elem->item,

'asignado' => $elem->asignado

];

if ($qr->tipo == "SERVICIO") {

$atributosPR['equipo'] = $elem->equipo;

$atributosPR['serie'] = $elem->serie;

$atributosPR['modelo'] = $elem->modelo;

$atributosPR['fabricante'] = $elem->fabricante;

}

$this->Conexion->insertar('pr\_atributos', $atributosPR);

}

$this->Conexion->eliminar('qr\_atributos\_temp', ['idQr' => $qrt]);

}

}

// Enviar correo si se generó correctamente

if ($res > 0) {

$correoData = [

'id' => $res,

'fecha' => date('d/m/Y h:i A'),

'usuario' => $this->session->nombre,

'cantidad' => $qty,

'unidad' => $qr->unidad,

'descripcion' => $qr->descripcion,

'atributos' => $qr->atributos,

'prioridad' => $qr->prioridad,

'comentarios' => "",

'correos' => array\_merge([$this->session->correo], $this->Modelo->getAprobadorPR($this->session->id, $qr->destino))

];

$this->correos\_pr->creacionPR($correoData);

echo $res;

}

}

// Edita cantidad e importe de una PR, reinicia estatus a 'PENDIENTE'

function ajax\_editarPR() {

$id = $this->input->post('id');

$qty = $this->input->post('qty');

$query = "UPDATE prs SET cantidad = $qty, importe = precio\_unitario \* cantidad, estatus = 'PENDIENTE' WHERE id = $id";

$this->Conexion->comando($query);

echo "1";

}

// Recupera lista de PRs filtradas por múltiples criterios

function ajax\_getPRs() {

$curUser = $this->session->id;

$misprs = $this->input->post('misprs');

$prioridad = json\_decode($this->input->post('prioridad'));

$estatus = $this->input->post('estatus');

$texto = $this->input->post('texto');

$parametro = $this->input->post('parametro');

$fecha1 = $this->input->post('fecha1');

$fecha2 = $this->input->post('fecha2');

$f1 = $fecha1 . ' 00:00:00';

$f2 = $fecha2 . ' 23:59:59';

$tipo = json\_decode($this->input->post('tipo'));

$stock = $this->input->post('stock');

$archivo = $this->input->post('archivo');

$query = "SELECT

PR.id, PR.fecha, PR.usuario, PR.prioridad, PR.tipo, PR.subtipo,

PR.cantidad, PR.unidad, PR.clave\_unidad, PR.descripcion, PR.atributos,

PR.critico, PR.destino, PR.lugar\_entrega, PR.comentarios, PR.estatus,

CONCAT(U.nombre, ' ', U.paterno) AS User,

U.autorizador\_compras, U.autorizador\_compras\_venta, PR.stock,

IFNULL((

SELECT OCC.po

FROM ordenes\_compra\_conceptos OCC

INNER JOIN ordenes\_compra OC ON OCC.po = OC.id

WHERE OCC.pr = PR.id AND OC.estatus != 'CANCELADA' LIMIT 1

), 0) AS POActual

FROM prs PR

LEFT JOIN usuarios U ON PR.usuario = U.id

WHERE 1 = 1";

if ($estatus != 'TODO') {

$query .= " AND PR.estatus = '$estatus'";

}

if (count($prioridad) > 0) {

$query .= " AND (0";

foreach ($prioridad as $value) {

$query .= " OR PR.prioridad = '$value'";

}

$query .= ")";

}

if ($misprs == "1") {

$query .= " AND IF(PR.destino = 'VENTA', U.autorizador\_compras\_venta = $curUser, U.autorizador\_compras = $curUser)";

}

if ($stock == "1") {

$query .= " AND PR.stock = 1";

}

if (!empty($texto)) {

switch ($parametro) {

case "folio":

$query .= " AND PR.id = '$texto'";

break;

case "usuario":

$query .= " HAVING User LIKE '%$texto%'";

break;

case "contenido":

$query .= " AND (

PR.descripcion LIKE '%$texto%' OR

UPPER(PR.atributos->'$.marca') LIKE UPPER('%$texto%') OR

UPPER(PR.atributos->'$.modelo') LIKE UPPER('%$texto%')

)";

break;

case "productos":

$query .= " AND PR.tipo LIKE '%$texto%'";

break;

case "servicios":

$query .= " AND PR.subtipo LIKE '%$texto%'";

break;

}

}

if (isset($tipo) && count($tipo) > 0) {

$query .= " AND (0";

foreach ($tipo as $value) {

$query .= " OR PR.tipo = '$value'";

}

$query .= ")";

}

if (!empty($fecha1) && !empty($fecha2)) {

$query .= " AND PR.fecha BETWEEN '$f1' AND '$f2'";

}

if (true) {

if ($archivo != 1) {

$query .= " AND PR.fecha > '2022-01-01 00:00:00'";

}

$query .= " ORDER BY PR.fecha DESC";

}

$res = $this->Modelo->consulta($query);

if ($res) {

echo json\_encode($res);

}

}

// Obtiene PRs del usuario actual con filtros avanzados

function ajax\_getMisPRs() {

$user = $this->session->id;

$prioridad = json\_decode($this->input->post('prioridad'));

$estatus = $this->input->post('estatus');

$texto = $this->input->post('texto');

$parametro = $this->input->post('parametro');

$misprs = $this->input->post('misprs');

$query = "SELECT

PR.id, PR.fecha, PR.usuario,

CONCAT(U.nombre, ' ', U.paterno, ' ', U.materno) AS User,

PR.prioridad, PR.tipo, PR.subtipo, PR.cantidad, PR.unidad, PR.clave\_unidad,

PR.descripcion, PR.atributos, PR.critico, PR.destino, PR.lugar\_entrega,

PR.comentarios, PR.estatus,

IFNULL((

SELECT OCC.po

FROM ordenes\_compra\_conceptos OCC

INNER JOIN ordenes\_compra OC ON OCC.po = OC.id

WHERE OCC.pr = PR.id AND OC.estatus != 'CANCELADA' LIMIT 1

), 0) AS POActual

FROM prs PR

INNER JOIN usuarios U ON U.id = PR.usuario

WHERE 1 = 1";

if ($misprs == "1") {

$query .= " AND PR.usuario = $user";

}

if ($estatus != 'TODO') {

$query .= " AND PR.estatus = '$estatus'";

}

if (count($prioridad) > 0) {

$query .= " AND (0";

foreach ($prioridad as $value) {

$query .= " OR PR.prioridad = '$value'";

}

$query .= ")";

}

if (!empty($texto)) {

if ($parametro == "folio") {

$query .= " AND PR.id = '$texto'";

}

if ($parametro == "contenido") {

$query .= " AND (

PR.descripcion LIKE '%$texto%' OR

UPPER(PR.atributos->'$.marca') LIKE UPPER('%$texto%') OR

UPPER(PR.atributos->'$.modelo') LIKE UPPER('%$texto%')

)";

}

}

$query .= " ORDER BY PR.fecha DESC";

$res = $this->Modelo->consulta($query);

if ($res) {

echo json\_encode($res);

} else {

echo "";

}

}

// Obtiene información detallada de una PR específica

function ajax\_getPR() {

$id = $this->input->post('id');

$query = "SELECT PR.\*, CONCAT(U.nombre, ' ', U.paterno) AS User, U.correo

FROM prs PR

INNER JOIN usuarios U ON U.id = PR.usuario

WHERE PR.id = $id";

$res = $this->Conexion->consultar($query, TRUE);

if ($res) {

echo json\_encode($res);

} else {

echo "";

}

}

// Obtiene detalles del proveedor relacionado a una PR

function ajax\_getProveedorPR() {

$id = $this->input->post('id');

$query = "SELECT

E.id, QP.id AS idQP, P.entrega, E.nombre,

QP.monto, QP.total, QP.moneda, QP.tiempo\_entrega,

QP.dias\_habiles, QP.comentarios, QP.nominado, QP.seleccionado,

QP.nombre\_archivo, QP.vencimiento

FROM qr\_proveedores QP

INNER JOIN empresas E ON E.id = QP.empresa

INNER JOIN proveedores P ON P.empresa = E.id

WHERE QP.id = '$id'";

$res = $this->Modelo->consulta($query);

if ($res) {

echo json\_encode($res);

} else {

echo "";

}

}

// Cambia el estatus de una PR y envía correo si aplica

function ajax\_setEstatusPR() {

$this->load->model('compras\_model');

$estatus = $this->input->post('estatus');

$id = $this->input->post('id');

$aprobador = $this->session->id;

// Bitácora del cambio de estatus (no se almacena en esta función, pero se construye)

$bitacora = [$estatus, date("Y-m-d h:i:sa"), $aprobador];

$estatusBit = json\_encode([$bitacora]);

$query = "UPDATE prs SET estatus = '$estatus' WHERE id = '$id'";

$data['pr'] = $id;

$data['user'] = $aprobador;

$data['estatus'] = $estatus;

// Si fue aprobado, agrega fecha y aprobador

if ($estatus == "APROBADO") {

$query = "UPDATE prs SET estatus = '$estatus', fecha\_aprobacion = CURRENT\_TIMESTAMP(), aprobador = $aprobador WHERE id = '$id'";

$this->compras\_model->estatusPR($data);

}

// Si fue cerrado, agrega fecha de entrega

if ($estatus == "CERRADO") {

$query = "UPDATE prs SET estatus = '$estatus', entregado = CURRENT\_TIMESTAMP() WHERE id = '$id'";

if ($this->Modelo->update($query)) {

echo "1";

}

exit();

}

// Ejecuta la actualización de estatus

$res = $this->Modelo->update($query);

if ($res) {

$pr = $this->Modelo->getPR($id);

$datos['id'] = $id;

$datos['fecha'] = date('d/m/Y h:i A');

$datos['usuario'] = $pr->User;

$datos['prioridad'] = $pr->prioridad;

$datos['unidad'] = $pr->unidad;

$datos['cantidad'] = $pr->cantidad;

$datos['descripcion'] = $pr->descripcion;

$datos['atributos'] = $pr->atributos;

$datos['comentarios'] = $pr->comentarios;

$datos['estatus'] = $estatus;

// Si fue aprobado, notifica por correo

if ($estatus == "APROBADO") {

$datos['correos'] = array\_merge([$qr->correo], $this->Modelo->getCorreosQR());

$this->correos\_pr->liberarPR($datos);

}

echo "1";

} else {

echo "";

}

}

// Agrega comentario a una PR y notifica a los correos relacionados

function agregarComentarioPR() {

$id = $this->input->post('id');

$comentario = $this->input->post('comentario');

$tags = $this->input->post('txtTags');

$correos = explode(",", $tags);

// Correos del solicitante y del asignado a la QR relacionada

$query = "SELECT u.correo AS correoReq, ua.correo AS correoA

FROM prs pr

JOIN usuarios u ON u.id = pr.usuario

JOIN requisiciones\_cotizacion qr ON qr.id = pr.qr

JOIN usuarios ua ON qr.asignado = ua.id

WHERE pr.id = $id";

$res = $this->Modelo->consulta($query, TRUE);

$mails = explode(',', $res->correoReq . ',' . $res->correoA);

// Datos para enviar notificación de comentario

$data['id'] = $id;

$data['comentario'] = $comentario;

$data['correos'] = $mails;

$data['nombre'] = $this->session->nombre;

$this->correos\_pr->comentarioPR($data);

// Guarda el comentario en la base de datos

$insert = [

'pr' => $id,

'usuario' => $this->session->id,

'comentario' => $comentario

];

$this->Modelo->agregar\_comentarioPR($insert);

// Envía comentario a correos marcados con etiquetas (si hay)

if (count($correos) > 0) {

$datos['id'] = $id;

$datos['comentario'] = $comentario;

$datos['correos'] = $correos;

$this->correos\_pr->comentarioPR($datos);

}

redirect(base\_url('compras/ver\_pr/' . $id));

}

// Cambia estatus de PR con comentario incluido y notificación por correo

function ajax\_setEstatusMsjPR() {

$this->load->model('compras\_model');

$id = $this->input->post('id');

$estatus = $this->input->post('estatus');

$comentario\_original = $this->input->post('comentario');

$comentario = "<b><font color='red'>$estatus:</font></b> " . $comentario\_original;

$tags = $this->input->post('txtTags');

$correos = explode(",", $tags);

// Actualiza estatus en base de datos

$query = "UPDATE prs SET estatus = '$estatus' WHERE id = '$id'";

$data['pr'] = $id;

$data['user'] = $this->session->id;

$data['estatus'] = $estatus;

$this->compras\_model->estatusPR($data);

$res = $this->Modelo->update($query);

if ($res) {

// Registra comentario con el estatus como prefijo

$data = [

'pr' => $id,

'usuario' => $this->session->id,

'comentario' => $comentario,

];

$this->Modelo->agregar\_comentarioPR($data);

$pr = $this->Modelo->getPR($id);

$datos['id'] = $id;

$datos['fecha'] = date('d/m/Y h:i A');

$datos['usuario'] = $pr->User;

$datos['prioridad'] = $pr->prioridad;

$datos['unidad'] = $pr->unidad;

$datos['cantidad'] = $pr->cantidad;

$datos['descripcion'] = $pr->descripcion;

$datos['atributos'] = $pr->atributos;

$datos['comentarios'] = $pr->comentarios;

$datos['estatus'] = $estatus;

// Enviar correo solo si fue rechazado

if ($estatus == "RECHAZADO") {

$datos['correos'] = [$pr->correo];

$this->correos\_pr->rechazoPR($datos);

}

// Correos personalizados vía etiquetas

if (count($correos) > 0) {

$datos2['id'] = $id;

$datos2['comentario'] = $comentario;

$datos2['correos'] = $correos;

$this->correos\_pr->comentarioPR($datos2);

}

echo $comentario;

} else {

echo "";

}

}

// Retorna lista de usuarios con permiso para aprobar PRs

function ajax\_getLiberadoresCompra() {

$query = "SELECT U.id, CONCAT(U.nombre, ' ', U.paterno, ' ', U.materno) AS Name

FROM usuarios U

INNER JOIN privilegios P ON P.usuario = U.id

WHERE U.activo = 1 AND P.aprobar\_pr = 1";

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

// Retorna lista de usuarios con permiso para aprobar cotizaciones

function ajax\_getLiberadoresCotizacion() {

$query = "SELECT U.id, CONCAT(U.nombre, ' ', U.paterno, ' ', U.materno) AS Name

FROM usuarios U

INNER JOIN privilegios P ON P.usuario = U.id

WHERE U.activo = 1 AND P.aprobar\_cotizacion = 1";

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function exportarQR() {

// Recolección de filtros desde POST

$parametro = $this->input->post('rbBusqueda');

$texto = $this->input->post('txtBusqueda');

$estatus = $this->input->post('opEstatus');

$usuario = $this->input->post('cbMisQrs');

$cbNormal = $this->input->post('cbNormal');

$cbUrgente = $this->input->post('cbUrgente');

$cbInfoUrgente = $this->input->post('cbInfoUrgente');

$cbProducto = $this->input->post('cbProducto');

$cbServicio = $this->input->post('cbServicio');

$asignado = $this->input->post('opAsignado');

$archivo = $this->input->post('cbArchivo');

$fecha1 = $this->input->post('fecha1');

$fecha2 = $this->input->post('fecha2');

$f1 = $fecha1 . ' 00:00:00';

$f2 = $fecha2 . ' 23:59:59';

// Consulta base

$query = "SELECT R.id, R.fecha, R.usuario, R.prioridad, R.tipo, R.subtipo, R.cantidad, R.cantidad\_aprobada, R.unidad, R.clave\_unidad, R.descripcion, R.atributos, R.critico, R.destino, R.lugar\_entrega, R.comentarios, R.estatus, R.asignado, CONCAT(U.nombre, ' ', U.paterno) as User, R.fecha\_liberacion

FROM requisiciones\_cotizacion R

LEFT JOIN usuarios U ON R.usuario = U.id

WHERE 1 = 1";

// Filtros aplicados

if ($estatus != 'TODO') {

$query .= " AND R.estatus = '$estatus'";

}

if (!empty($asignado)) {

$query .= " AND R.asignado = '$asignado'";

}

if ($usuario == 'NORMAL') {

$idUser = $this->session->id;

$query .= " AND R.usuario = '$idUser'";

}

// Prioridades

$query .= " AND (1 = 0";

if (!empty($cbNormal)) $query .= " OR R.prioridad = '$cbNormal'";

if (!empty($cbUrgente)) $query .= " OR R.prioridad = '$cbUrgente'";

if (!empty($cbInfoUrgente)) $query .= " OR R.prioridad = '$cbInfoUrgente'";

$query .= ")";

// Tipo de producto o servicio

$query .= " AND (1 = 0";

if (!empty($cbProducto)) $query .= " OR R.tipo = '$cbProducto'";

if (!empty($cbServicio)) $query .= " OR R.tipo = '$cbServicio'";

$query .= ")";

// Filtro por privilegios internos o de venta

if ($this->session->privilegios['crear\_qr\_interno'] != $this->session->privilegios['crear\_qr\_venta']) {

if ($this->session->privilegios['editar\_qr'] == "0" && $this->session->privilegios['liberar\_qr'] == "0") {

if ($this->session->privilegios['crear\_qr\_interno'] == "1") {

$query .= " AND R.destino = 'CONSUMO INTERNO'";

} else {

$query .= " AND R.destino = 'VENTA'";

}

}

}

// Búsqueda por texto

if (!empty($texto)) {

if ($parametro == "folio") {

$query .= " AND R.id = '$texto'";

}

if ($parametro == "usuario") {

$query .= " AND CONCAT(U.nombre, ' ', U.paterno) LIKE '%$texto%'";

}

if ($parametro == "contenido") {

$query .= " AND (R.descripcion LIKE '%$texto%'

OR UPPER(R.atributos->'$.marca') LIKE UPPER('%$texto%')

OR UPPER(R.atributos->'$.modelo') LIKE UPPER('%$texto%'))";

}

}

// Rango de fechas

if (!empty($fecha1) && !empty($fecha2)) {

$query .= " AND R.fecha BETWEEN '$f1' AND '$f2'";

}

// Solo QRs del último año

$query .= " AND R.maximo\_vencimiento > (CURRENT\_DATE() - INTERVAL 1 YEAR)";

// Filtrar si no es exportación completa

if ($archivo != "1") {

$query .= " AND R.fecha > '2021-01-01 00:00:00'";

}

$query .= " ORDER BY R.fecha DESC";

$result = $this->Conexion->consultar($query);

// Construcción de tabla para exportar

$salida = '<table style="border: 1px solid black; border-collapse: collapse;">

<thead>

<tr>

<th style="background-color: #F3F1F1; border: 1px solid black;">QR</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Fecha</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Requisitor</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Prioridad</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Tipo</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Subtipo</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Cantidad</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Estatus</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Fecha Liberado</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Días Transcurridos</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Descripción</th>

</tr>

</thead>

<tbody>';

$d = 2;

foreach ($result as $row) {

$salida .= '<tr>

<td style="border: 1px solid black;">' . $row->id . '</td>

<td style="border: 1px solid black;">' . $row->fecha . '</td>

<td style="border: 1px solid black;">' . $row->User . '</td>

<td style="border: 1px solid black;">' . $row->prioridad . '</td>

<td style="border: 1px solid black;">' . $row->tipo . '</td>

<td style="border: 1px solid black;">' . $row->subtipo . '</td>

<td style="border: 1px solid black;">' . $row->cantidad . '</td>

<td style="border: 1px solid black;">' . $row->estatus . '</td>

<td style="border: 1px solid black;">' . $row->fecha\_liberacion . '</td>

<td style="border: 1px solid black;">=SI(ESBLANCO($I' . $d . '),"",DIAS($I' . $d . ',$B' . $d . '))</td>

<td style="border: 1px solid black;">' . $row->descripcion . '</td>

</tr>';

$d++;

}

$salida .= '</tbody></table>';

// Preparar headers para descarga como archivo Excel

$timestamp = date('m-d-Y');

$filename = 'QR\_' . $timestamp . '.xls';

header("Content-Type: application/vnd.ms-excel");

header("Content-Disposition: attachment; filename=\"$filename\"");

header("Pragma: no-cache");

header("Expires: 0");

header("Content-Transfer-Encoding: binary");

echo $salida;

}

function exportarPR()

{

// Entrada de parámetros desde POST

$opc = $this->input->post('rbBusqueda');

$texto = $this->input->post('txtBusqueda');

$estatus = $this->input->post('opEstatus');

$fecha1 = $this->input->post('fecha1');

$fecha2 = $this->input->post('fecha2');

$f1 = strval($fecha1) . ' 00:00:00';

$f2 = strval($fecha2) . ' 23:59:59';

$stock = $this->input->post('stock');

$archivo = $this->input->post('cbArchivo');

// Consulta base

$query = "SELECT

PR.id as PR,

PR.fecha as Fecha,

concat(U.nombre, ' ', U.paterno) as Requisitor,

PR.prioridad as Prioridad,

PR.tipo as Tipo,

PR.subtipo as Subtipo,

PR.cantidad as Cantidad,

PR.estatus as Estatus,

ifnull((

SELECT OCC.po

FROM ordenes\_compra\_conceptos OCC

INNER JOIN ordenes\_compra OC ON OCC.po = OC.id

WHERE OCC.pr = PR.id AND OC.estatus != 'CANCELADA'

LIMIT 1

), 0) as PO,

concat('$ ', format(PR.importe, 2)) as MONTO,

PR.moneda as Moneda,

(

SELECT P.entrega

FROM qr\_proveedores QP

INNER JOIN empresas E ON E.id = QP.empresa

INNER JOIN proveedores P ON P.empresa = E.id

WHERE QP.id = PR.qr\_proveedor

) as Lugar,

(

SELECT E.nombre

FROM qr\_proveedores QP

INNER JOIN empresas E ON E.id = QP.empresa

INNER JOIN proveedores P ON P.empresa = E.id

WHERE QP.id = PR.qr\_proveedor

) as Proveedor,

PR.descripcion as Descripcion

FROM prs PR

LEFT JOIN usuarios U ON PR.usuario = U.id

WHERE 1 = 1";

// Filtros dinámicos

if ($estatus != 'TODO') {

$query .= " AND PR.estatus = '$estatus'";

}

if (!empty($texto)) {

if ($opc == "folio") {

$query .= " AND PR.id = '$texto'";

}

if ($opc == "usuario") {

$query .= " HAVING Requisitor LIKE '%$texto%'";

}

if ($opc == "contenido") {

$query .= " AND (PR.descripcion LIKE '%$texto%'

OR UPPER(PR.atributos->'$.marca') LIKE UPPER('%$texto%')

OR UPPER(PR.atributos->'$.modelo') LIKE UPPER('%$texto%'))";

}

}

if (!empty($fecha1) && !empty($fecha2)) {

$query .= " AND PR.fecha BETWEEN '$fecha1' AND '$fecha2'";

}

if (!empty($stock)) {

$query .= " AND PR.stock = 1";

}

if ($archivo != "1") {

$query .= " AND PR.fecha > '2021-01-01 00:00:00'";

}

$query .= " ORDER BY PR.fecha DESC";

// Ejecutar consulta

$result = $this->Conexion->consultar($query);

// Construcción de tabla HTML para exportar a Excel

$salida = '';

$salida .= '<table style="border: 1px solid black; border-collapse: collapse;">

<thead>

<tr>

<th style="background-color: #F3F1F1; border: 1px solid black;">PR</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Fecha</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Requisitor</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Prioridad</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Tipo</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Subtipo</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Cantidad</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Estatus</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">PO</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Monto</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Moneda</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Lugar de entrega</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Proveedor</th>

<th style="background-color: #F3F1F1; border: 1px solid black;">Descripcion</th>

</tr>

</thead>

<tbody>';

foreach ($result as $row) {

$salida .= '

<tr>

<td style="border: 1px solid black;">' . $row->PR . '</td>

<td style="border: 1px solid black;">' . $row->Fecha . '</td>

<td style="border: 1px solid black;">' . $row->Requisitor . '</td>

<td style="border: 1px solid black;">' . $row->Prioridad . '</td>

<td style="border: 1px solid black;">' . $row->Tipo . '</td>

<td style="border: 1px solid black;">' . $row->Subtipo . '</td>

<td style="border: 1px solid black;">' . $row->Cantidad . '</td>

<td style="border: 1px solid black;">' . $row->Estatus . '</td>

<td style="border: 1px solid black;">' . $row->PO . '</td>

<td style="border: 1px solid black;">' . $row->MONTO . '</td>

<td style="border: 1px solid black;">' . $row->Moneda . '</td>

<td style="border: 1px solid black;">' . $row->Lugar . '</td>

<td style="border: 1px solid black;">' . $row->Proveedor . '</td>

<td style="border: 1px solid black;">' . $row->Descripcion . '</td>

</tr>';

}

$salida .= '</tbody></table>';

// Preparar headers para descargar el archivo como Excel

$timestamp = date('m/d/Y', time());

$filename = 'PR\_' . $timestamp . '.xls';

header("Content-Type: application/vnd.ms-excel");

header("Content-Disposition: attachment; filename=\"$filename\"");

header("Pragma: no-cache");

header("Expires: 0");

header('Content-Transfer-Encoding: binary');

echo $salida;

}

function AllPR(){

// Obtiene el ID de la PR desde POST

$id = $this->input->post('CURRENT\_PR');

// Consulta toda la información de la PR específica

$query = "SELECT \* from prs where id = ".$id;

// Ejecuta la consulta

$res = $this->Conexion->consultar($query, $id);

// Si hay resultado, lo devuelve como JSON

if($res) {

echo json\_encode($res);

}

}

function ajax\_getNombresUsuarios(){

// Obtiene el ID de la PR desde POST

$ids = $this->input->post('CURRENT\_PR');

// Consulta la bitácora de estatus para esa PR y quién los realizó

$query = "SELECT concat(u.nombre, ' ', u.paterno) as user, b.fecha, b.estatus

from bitacora\_prs b

JOIN usuarios u on b.user = u.id

where pr = ".$ids;

// Ejecuta la consulta y devuelve JSON si hay resultados

$res = $this->Conexion->consultar($query);

if($res) {

echo json\_encode($res);

}

}

function bitacoraQR(){

// Obtiene el ID del QR desde POST

$ids = $this->input->post('id');

// Consulta la bitácora de estatus para ese QR y quién los realizó

$query = "SELECT concat(u.nombre, ' ', u.paterno) as user, b.fecha, b.estatus

from bitacora\_qrs b

JOIN usuarios u on b.user = u.id

where qr = ".$ids;

// Ejecuta la consulta y devuelve JSON si hay resultados

$res = $this->Conexion->consultar($query);

if($res) {

echo json\_encode($res);

}

}

public function ajax\_surtirPR()

{

// Obtiene el ID de la PR desde POST

$pr = $this->input->post('id');

// Establece zona horaria y obtiene fecha/hora actual

date\_default\_timezone\_set('America/Chihuahua');

$date = date('Y-m-d h:i:s');

// Prepara los datos para actualizar la PR

$data['estatus'] = "POR RECIBIR";

$data['stock'] = 1;

$data['surtidorStock'] = $this->session->id;

$data['fechaSurtido'] = $date;

$where['id'] = intval($pr);

// Ejecuta la actualización en la tabla `prs`

$res = $this->Conexion->modificar('prs', $data, null, $where);

// Registra en bitácora el evento "SURTIDO DE STOCK"

$bitacoraPR['pr'] = intval($pr);

$bitacoraPR['user'] = $this->session->id;

$bitacoraPR['estatus'] = 'SURTIDO DE STOCK';

$this->Modelo->estatusPR($bitacoraPR);

// Registra en bitácora el nuevo estatus "POR RECIBIR"

$bitacoraPR['pr'] = intval($pr);

$bitacoraPR['user'] = $this->session->id;

$bitacoraPR['estatus'] = 'POR RECIBIR';

$this->Modelo->estatusPR($bitacoraPR);

// Devuelve 1 si la actualización fue exitosa

if ($res) {

echo 1;

}

}

function ajax\_getUsuariosQR(){

// Obtener usuarios activos del departamento de COMPRAS con privilegios de editar QR

$query = "SELECT U.id, concat(U.nombre, ' ', U.paterno) as Nombre, P.puesto as Puesto, U.correo, PR.editar\_qr

FROM usuarios U

INNER JOIN puestos P ON U.puesto = P.id

INNER JOIN privilegios PR ON PR.usuario = U.id

WHERE U.activo = 1 AND U.departamento='COMPRAS'";

$res = $this->Conexion->consultar($query);

if($res) {

echo json\_encode($res);

}

}

function asignarUsuariosQR(){

// Obtener datos desde POST

$qr = $this->input->post('qr');

$id = $this->input->post('id');

// Consultar correo del usuario a asignar

$mail = "SELECT correo FROM usuarios WHERE id =".$id;

$correo = $this->Conexion->consultar($mail);

// Enviar correo de asignación

foreach($correo as $elem){

$dato['qr'] = $qr;

$dato['correo'] = $elem->correo;

$this->correos->asignarQR($dato);

}

// Guardar asignación en base de datos

date\_default\_timezone\_set('America/Chihuahua');

$date = date('Y-m-d h:i:s');

$data['asignado'] = intval($id);

$data['asignador'] = $this->session->id;

$data['fechaAsignacion'] = $date;

$where['id'] = intval($qr);

$this->Conexion->modificar('requisiciones\_cotizacion', $data, null, $where);

// Registrar bitácora

$bitacoraQR['qr'] = intval($qr);

$bitacoraQR['user'] = $this->session->id;

$bitacoraQR['estatus'] = 'Asignacion por: ' . $this->session->nombre;

$this->Modelo->estatusQR($bitacoraQR);

// Consultar y devolver información completa de asignación

$query = "SELECT U.id, concat(U.nombre, ' ', U.paterno) as Nombre, P.puesto as Puesto, U.correo,

qr.id as QR, qr.asignado, fechaAsignacion, concat(A.nombre, ' ', A.paterno) as Asignador

FROM usuarios U

INNER JOIN puestos P ON U.puesto = P.id

JOIN requisiciones\_cotizacion qr ON qr.asignado = U.id

JOIN usuarios A ON A.id = qr.asignador

WHERE qr.id = ".$qr;

$res = $this->Conexion->consultar($query);

if($res) {

echo json\_encode($res);

}

}

function buscarML(){

// Recibe parámetros por POST

$item = $this->input->post('item');

$destino = $this->input->post('destino');

$tipo = $this->input->post('tipo');

// Consulta los items disponibles según tipo/destino

if ($destino == 'VENTA' && $tipo == 'PRODUCTO') {

$query = "SELECT rs.Item\_id, rs.Equipo\_ID, rs.serie , rs.tecnico\_id, rs.fechaActEQ,

rs.Modelo, rs.Fabricante, t.Nombre

FROM rsitems rs

JOIN catalogo\_tecnicos t ON rs.tecnico\_id = t.Tecnico\_Id

WHERE item\_id = '".$item."' AND Equipo\_id IS NULL AND fechaACtEQ IS NULL";

} else {

$query = "SELECT rs.Item\_id, rs.Equipo\_ID, rs.serie , rs.tecnico\_id, rs.fechaActEQ,

rs.Modelo, rs.Fabricante, t.Nombre

FROM rsitems rs

JOIN catalogo\_tecnicos t ON rs.tecnico\_id = t.Tecnico\_Id

WHERE item\_id = '".$item."' AND Equipo\_id IS NOT NULL AND fechaACtEQ IS NULL";

}

// Ejecutar consulta

$res = $this->MLConexion->consultar($query);

if($res) {

echo json\_encode($res);

}

}

function agregarAtributos(){

// Recibe atributos e ID de QR por POST

$att = $this->input->post('ATT');

$qr = $this->input->post('qr');

// Inserta los atributos recibidos en la tabla temporal

foreach($att as $elem){

$datos['idQr'] = intval($qr);

$datos['item'] = $elem['Item\_id'];

$datos['equipo'] = $elem['Equipo\_ID'];

$datos['serie'] = $elem['serie'];

$datos['modelo'] = $elem['Modelo'];

$datos['fabricante'] = $elem['Fabricante'];

$datos['asignado'] = $elem['Nombre'];

$this->Conexion->insertar('qr\_atributos\_temp', $datos, $funciones = null);

}

// Consultar todos los atributos agregados para ese QR

$query = "SELECT \* FROM qr\_atributos\_temp WHERE idQr = ".$qr;

$res = $this->Conexion->consultar($query);

if($res) {

echo json\_encode($res);

}

}

function cargarItems(){

// Recibe ID de QR por POST

$qr = $this->input->post('qr');

// Consulta todos los items temporales relacionados a ese QR

$query = "SELECT \* FROM qr\_atributos\_temp WHERE idQr = ".$qr;

$res = $this->Conexion->consultar($query);

if($res) {

echo json\_encode($res);

}

}

function eliminarAtributos(){

// Elimina un atributo temporal del QR y devuelve los atributos restantes

$id = $this->input->post('id');

$qr = $this->input->post('qr');

$where['id'] = $id;

$this->Conexion->eliminar('qr\_atributos\_temp', $where);

$query = "SELECT \* FROM qr\_atributos\_temp WHERE idQr = ".$qr;

$res = $this->Conexion->consultar($query);

if($res) {

echo json\_encode($res);

}

}

function asignarCompras(){

// Asigna un técnico fijo (id 34) a un item en rsitems (ML), y lo etiqueta como 'Compras' en atributos temporales

$item = $this->input->post('item');

$qr = $this->input->post('qr');

$data['tecnico\_id'] = '34';

$where['Item\_id'] = $item;

$this->MLConexion->modificar('rsitems', $data, null, $where);

$dato['asignado'] = 'Compras';

$donde['item'] = $item;

$this->Conexion->modificar('qr\_atributos\_temp', $dato, null, $donde);

$query = "SELECT \* FROM qr\_atributos\_temp WHERE idQr = ".$qr;

$res = $this->Conexion->consultar($query);

if($res) {

echo json\_encode($res);

}

}

function validarAtributos(){

// Verifica si un item ya está registrado en un QR específico

$item = $this->input->post('item');

$qr = $this->input->post('qr');

$query = "SELECT \* FROM qr\_atributos\_temp WHERE item = ".$item." AND idQr = ".$qr;

$res = $this->Conexion->consultar($query);

if($res) {

echo json\_encode($res);

}

}

function getItem(){

// Obtiene detalles de un atributo PR específico

$item = $this->input->post('item');

$query = "SELECT \* FROM pr\_atributos WHERE id = ".$item;

$res = $this->Conexion->consultar($query, TRUE);

if($res) {

echo json\_encode($res);

}

}

function ValidarItem(){

// Verifica si un item ya está ligado a una PR activa (no cancelada)

$item = $this->input->post('item');

$query = "SELECT pa.item, pa.idPr, pr.estatus

FROM pr\_atributos pa

JOIN prs pr ON pa.idPr = pr.id

WHERE pa.item = '".$item."' AND pr.estatus != 'CANCELADO'";

$res = $this->Conexion->consultar($query, TRUE);

if($res) {

echo json\_encode($res);

}

}

function getFileEx(){

// Obtiene el nombre y contenido binario del archivo de ejemplo asociado a un QR

$id = $this->input->post('qr');

$query = "SELECT nombre\_archivoEjemplo, archivoEjemplo

FROM requisiciones\_cotizacion

WHERE id = ".$id;

$res = $this->Modelo->consulta($query, TRUE);

if($res) {

echo json\_encode($res);

} else {

echo "";

}

}

function getFileExample($qr){

// Despliega un archivo PDF ejemplo en el navegador sin forzar descarga

$row = $this->descargas\_model->getFile($qr, 'requisiciones\_cotizacion');

$file = $row->archivoEjemplo;

$nombre = $row->nombre\_archivoEjemplo;

header('Content-type: application/pdf');

header('Content-Disposition: inline; filename="' . $nombre . '"');

header('Content-Transfer-Encoding: binary');

header('Accept-Ranges: bytes');

echo $file;

}

function ajax\_subirArchivoQRClon() {

// Clona el archivo PDF de un QR y lo asocia al QR más reciente

$id = $this->input->post('qr');

// Obtener archivo original del QR base

$query = "SELECT nombre\_archivo, archivo FROM requisiciones\_cotizacion WHERE id = " . $id;

$res = $this->Conexion->consultar($query, TRUE);

$datos['archivo'] = $res->archivo;

// Obtener el último QR generado (para usar su ID como nuevo archivo)

$query = "SELECT \* FROM requisiciones\_cotizacion ORDER BY id DESC LIMIT 1";

$r = $this->Conexion->consultar($query, TRUE);

$datos['id'] = $r->id;

$datos['nombre\_archivo'] = str\_pad($datos['id'], 6, "0", STR\_PAD\_LEFT) . ".pdf";

if (!$this->Modelo->setQRFile($datos)) {

trigger\_error("Error al subir archivo", E\_USER\_ERROR);

} else {

echo "1";

}

}

function checkFile() {

// Verifica si el QR tiene un archivo cargado

$id = $this->input->post('qr');

$query = "SELECT nombre\_archivo FROM requisiciones\_cotizacion WHERE id = " . $id;

$res = $this->Modelo->consulta($query, TRUE);

if ($res->nombre\_archivo) {

echo 1;

} else {

echo 0;

}

}

function uploadFileEx() {

// Sube archivo de ejemplo (binario) a la requisición

$datos['id'] = $this->input->post('qr');

$datos['archivoEjemplo'] = file\_get\_contents($\_FILES['file']['tmp\_name']);

$datos['nombre\_archivoEjemplo'] = $\_FILES['file']['name'];

if (!$this->Modelo->archivo\_ejemplo($datos)) {

trigger\_error("Error al subir archivo", E\_USER\_ERROR);

} else {

echo $datos['nombre\_archivoEjemplo'];

}

}

function eliminarArchivoEx() {

// Elimina el archivo de ejemplo asociado a una requisición

$datos['id'] = $this->input->post('qr');

$datos['archivoejemplo'] = null;

$datos['nombre\_archivoejemplo'] = null;

if (!$this->Modelo->archivo\_ejemplo($datos)) {

trigger\_error("Error al eliminar archivo", E\_USER\_ERROR);

} else {

echo "1";

}

}

function checkDestino() {

// Devuelve el destino asignado a una requisición QR

$id = $this->input->post('qr');

$query = "SELECT destino FROM requisiciones\_cotizacion WHERE id = " . $id;

$res = $this->Conexion->consultar($query, TRUE);

if ($res) {

echo json\_encode($res);

} else {

echo "1";

}

}

}

## Configuracion.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

// Controlador Configuracion: Maneja vistas de configuración general

class Configuracion extends CI\_Controller {

    // Constructor: Carga modelos necesarios

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('conexion\_model','Conexion');          // Modelo para conexión a BD

        $this->load->model('privilegios\_model');                  // Modelo para privilegios y notificaciones

    }

    // Vista principal de configuración de compras

    function compras() {

        $this->load->view('header');                              // Carga el encabezado

        $this->load->view('configuracion/compra\_opciones');       // Carga opciones de compras

    }

    // Vista principal de configuración de servicios

    function servicios() {

        $this->load->view('header');                              // Carga el encabezado

        $this->load->view('configuracion/servicios\_opciones');    // Carga opciones de servicios

    }

    // Vista principal de configuración de requerimientos

    function requerimientos() {

        $this->load->view('header');                              // Carga el encabezado

        $this->load->view('configuracion/requerimientos\_opciones'); // Carga opciones de requerimientos

    }

    // ====== SERVICIOS ======

    // Vista de claves y precios de servicios

    function claves\_precio(){

        $this->load->view('header');                               // Carga el encabezado

        $this->load->view('configuracion/servicios/claves\_precio');// Vista de claves y precios

    }

    // Vista de magnitudes de servicios

    function magnitudes(){

        $this->load->view('header');                               // Carga el encabezado

        $this->load->view('configuracion/servicios/magnitudes');  // Vista de magnitudes

    }

    // Vista para configuración de correo en servicios

    function correo(){

        $this->load->view('header');                               // Carga el encabezado

        $this->load->view('configuracion/servicios/correo');      // Vista de correo

    }

    // ====== COMPRAS ======

    // Vista para dirección de envío

    function shipping\_address() {

        $this->load->view('header');                               // Carga el encabezado

        $this->load->view('configuracion/compras/shipping\_address'); // Dirección de envío

    }

    // Vista para dirección de facturación

    function billing\_address() {

        $this->load->view('header');                               // Carga el encabezado

        $this->load->view('configuracion/compras/billing\_address'); // Dirección de facturación

    }

    // Vista para opciones de pago

    function pagos() {

        $this->load->view('header');                               // Carga el encabezado

        $this->load->view('configuracion/compras/pagos');         // Vista de pagos

    }

    // ====== NOTIFICACIONES ======

    // Vista de notificaciones con datos desde modelo de privilegios

    function notificaiones() {

        $datos['noti'] = $this->privilegios\_model->getNotificaciones(); // Obtiene notificaciones

        $this->load->view('header');                               // Carga el encabezado

        $this->load->view('configuracion/notificaciones', $datos); // Vista con notificaciones

    }

}

    // Obtiene claves de precio desde la base de datos (por ID si se proporciona)

    function ajax\_getClavesPrecio() {

    $id = 0;

        if(isset($\_POST['id'])) {

            $id = $this->input->post('id');  // Recupera ID si está presente en POST

        }

        $query = "SELECT \* from claves\_precio";

        $row = $id > 0;

    // Si se especificó un ID, se agrega condición WHERE

        if($row) {

            $query .= " where id = $id";

        }

        $res = $this->Conexion->consultar($query, $row);

    // Devuelve resultado en JSON o "0" si falla

        if($res) {

         echo json\_encode($res);

        } else {

         echo "0";

        }

}

// Actualiza valores de bajo y alto en la tabla claves\_precio

function ajax\_setClavesPrecio() {

    $id = $this->input->post('id');          // ID del registro a actualizar

    $bajo = $this->input->post('bajo');      // Valor bajo

    $alto = $this->input->post('alto');      // Valor alto

    $datos['bajo'] = $bajo;

    $datos['alto'] = $alto;

    $where['id'] = $id;

    $this->Conexion->modificar('claves\_precio', $datos, null, $where); // Realiza la actualización

    echo "1"; // Éxito

}

// Obtiene registros de magnitudes desde la base de datos (opcionalmente por ID)

function ajax\_getMagnitudes() {

    $id = 0;

    if(isset($\_POST['id'])) {

        $id = $this->input->post('id');  // Recupera ID si está presente en POST

    }

    $query = "SELECT \* from magnitudes";

    $row = $id > 0;

    // Si se especificó un ID, se agrega condición WHERE

    if($row) {

        $query .= " where id = $id";

    }

    $res = $this->Conexion->consultar($query, $row);

    // Devuelve resultado en JSON o "0" si falla

    if($res) {

        echo json\_encode($res);

    } else {

        echo "0";

    }

}

// Obtiene textos de correo activos, con nombre del usuario asociado (opcionalmente por ID)

function ajax\_getTexto\_correo() {

    $id = 0;

    if(isset($\_POST['id'])) {

        $id = $this->input->post('id');  // Recupera ID si está presente en POST

    }

    // Consulta que une texto\_correo con usuarios y filtra solo activos

    $query = "SELECT tc.\*, concat(u.nombre, ' ', u.paterno) as us

              FROM texto\_correo tc

              JOIN usuarios u on u.id = tc.id\_us

              WHERE tc.activo = 1";

    $row = $id > 0;

    // Si se especificó un ID, se agrega condición adicional

    if($row) {

        $query .= " and tc.id = $id";

    }

    $res = $this->Conexion->consultar($query, $row);

    // Devuelve resultado en JSON o "0" si falla

    if($res) {

        echo json\_encode($res);

    } else {

        echo "0";

    }

}

// Inserta o actualiza una magnitud

function ajax\_setMagnitudes() {

    $id = 0;

    if(isset($\_POST['id'])) {

        $id = $this->input->post('id'); // ID del registro (si aplica)

    }

    $magnitud = $this->input->post('magnitud'); // Nombre de la magnitud

    $prefijo = $this->input->post('prefijo');   // Prefijo asociado

    $datos['magnitud'] = $magnitud;

    $datos['prefijo'] = $prefijo;

    $res = FALSE;

    // Si el ID es 0, se inserta nuevo registro

    if($id == 0) {

        $res = $this->Conexion->insertar('magnitudes', $datos) > 0;

    } else {

        // Si ya existe, se actualiza el registro

        $where['id'] = $id;

        $res = $this->Conexion->modificar('magnitudes', $datos, null, $where) >= 0;

    }

    if($res) {

        echo "1"; // Éxito

    }

}

// Elimina una magnitud por ID

function ajax\_deleteMagnitudes() {

    $id = $this->input->post('id');     // ID del registro a eliminar

    $where['id'] = $id;

    // Elimina y devuelve "1" si tiene éxito

    if($this->Conexion->eliminar('magnitudes', $where)) {

        echo "1";

    }

}

// Verifica si un prefijo ya existe en la tabla magnitudes

function ajax\_prefijoExists() {

    $prefijo = $this->input->post('prefijo'); // Prefijo a verificar

    $query = "SELECT count(M.prefijo) as Qty

              FROM magnitudes M

              WHERE 1 = 1 AND prefijo = '$prefijo'";

    $res = $this->Conexion->consultar($query, TRUE);

    // Devuelve cantidad de coincidencias o vacío si falla

    if($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

// Obtiene direcciones de envío (shipping\_address), por ID si se especifica

function ajax\_getShippingAddresses() {

    $id = 0;

    if(isset($\_POST['id'])) {

        $id = $this->input->post('id'); // ID del registro si aplica

    }

    $query = "SELECT \* from shipping\_address";

    $row = $id > 0;

    // Si hay ID, se agrega condición WHERE

    if($row) {

        $query .= " where id = $id";

    }

    $res = $this->Conexion->consultar($query, $row);

    // Devuelve resultados en JSON o "0" si no hay resultados

    if($res) {

        echo json\_encode($res);

    } else {

        echo "0";

    }

}

// Inserta o actualiza una dirección de envío (shipping\_address)

function ajax\_setShippingAddresses() {

    $id = 0;

    if(isset($\_POST['id'])) {

        $id = $this->input->post('id'); // ID si se va a actualizar

    }

    $nombre = $this->input->post('nombre');       // Nombre del destinatario

    $direccion = $this->input->post('direccion'); // Dirección de envío

    $pais = $this->input->post('pais');           // País

    $datos['nombre'] = $nombre;

    $datos['direccion'] = $direccion;

    $datos['pais'] = $pais;

    $res = FALSE;

    // Inserta nuevo registro si ID es 0

    if($id == 0) {

        $datos['default'] = '0'; // No es default por defecto

        $res = $this->Conexion->insertar('shipping\_address', $datos) > 0;

    } else {

        // Actualiza dirección existente

        $where['id'] = $id;

        $res = $this->Conexion->modificar('shipping\_address', $datos, null, $where) >= 0;

    }

    if($res) {

        echo "1"; // Éxito

    }

}

// Elimina una dirección de envío por ID

function ajax\_deleteShippingAddresses() {

    $id = $this->input->post('id');     // ID de la dirección a eliminar

    $where['id'] = $id;

    // Elimina el registro y responde "1" si tuvo éxito

    if($this->Conexion->eliminar('shipping\_address', $where)) {

        echo "1";

    }

}

// Establece una dirección como predeterminada para un país específico

function ajax\_setDefaultShippingAddresses() {

    $where['pais'] = $this->input->post('pais');    // País al que pertenece la dirección

    $datos['default'] = '0';

    // Primero quita el estatus "default" de todas las direcciones del país

    $this->Conexion->modificar('shipping\_address', $datos, null, $where);

    // Luego asigna como default la dirección indicada por ID

    $where['id'] = $this->input->post('id');

    $datos['default'] = '1';

    $this->Conexion->modificar('shipping\_address', $datos, null, $where);

    echo "1"; // Confirmación de éxito

}

// Obtiene direcciones de facturación (billing\_address), por ID si se especifica

function ajax\_getBillingAddresses() {

    $id = 0;

    if(isset($\_POST['id'])) {

        $id = $this->input->post('id'); // ID si se desea un registro específico

    }

    $query = "SELECT \* from billing\_address";

    $row = $id > 0;

    // Si se especificó ID, agrega condición WHERE

    if($row) {

        $query .= " where id = $id";

    }

    $res = $this->Conexion->consultar($query, $row);

    // Devuelve resultado en JSON o "0" si no hay datos

    if($res) {

        echo json\_encode($res);

    } else {

        echo "0";

    }

}

// Inserta o actualiza una dirección de facturación (billing\_address)

function ajax\_setBillingAddresses() {

    $id = 0;

    if(isset($\_POST['id'])) {

        $id = $this->input->post('id'); // ID si es una actualización

    }

    $nombre = $this->input->post('nombre');         // Nombre del cliente

    $direccion = $this->input->post('direccion');   // Dirección fiscal

    $datos['nombre'] = $nombre;

    $datos['direccion'] = $direccion;

    $res = FALSE;

    // Si no hay ID, se inserta nuevo registro

    if($id == 0) {

        $datos['default'] = '0'; // No es default por defecto

        $res = $this->Conexion->insertar('billing\_address', $datos) > 0;

    } else {

        // Actualiza registro existente

        $where['id'] = $id;

        $res = $this->Conexion->modificar('billing\_address', $datos, null, $where) >= 0;

    }

    if($res) {

        echo "1"; // Éxito

    }

}

    // Elimina una dirección de facturación por ID

function ajax\_deleteBillingAddresses() {

    $id = $this->input->post('id');     // ID del registro a eliminar

    $where['id'] = $id;

    // Ejecuta la eliminación y devuelve "1" si tiene éxito

    if($this->Conexion->eliminar('billing\_address', $where)) {

        echo "1";

    }

}

// Establece una dirección de facturación como predeterminada

function ajax\_setDefaultBillingAddresses() {

    $where['id'] = $this->input->post('id'); // ID de la dirección que será default

    $datos['default'] = '0';

    // Pone todas las direcciones como no predeterminadas

    $this->Conexion->modificar('billing\_address', $datos, null, null);

    // Establece como predeterminada la dirección especificada

    $datos['default'] = '1';

    $this->Conexion->modificar('billing\_address', $datos, null, $where);

    echo "1"; // Éxito

}

// Obtiene métodos de pago (empresa\_metodos\_pago), por ID si se proporciona

function ajax\_getPagos() {

    $id = 0;

    if(isset($\_POST['id'])) {

        $id = $this->input->post('id'); // ID del método de pago a obtener

    }

    $query = "SELECT \* from empresa\_metodos\_pago";

    $row = $id > 0;

    // Aplica filtro si se especificó un ID

    if($row) {

        $query .= " where id = $id";

    }

    $res = $this->Conexion->consultar($query, $row);

    // Devuelve los datos en JSON o "0" si no hay resultados

    if($res) {

        echo json\_encode($res);

    } else {

        echo "0";

    }

}

// Inserta o actualiza un método de pago

function ajax\_setPagos() {

    $metodo = json\_decode($this->input->post('metodo\_pago'));  // Decodifica objeto desde JSON

    $metodo->fecha\_vencimiento = date("Y-m-d", strtotime($metodo->fecha\_vencimiento)); // Formatea fecha

    $res = FALSE;

    // Inserta nuevo método si el ID es 0

    if($metodo->id == 0) {

        $res = $this->Conexion->insertar('empresa\_metodos\_pago', $metodo) > 0;

    } else {

        // Actualiza método existente

        $where['id'] = $metodo->id;

        $res = $this->Conexion->modificar('empresa\_metodos\_pago', $metodo, null, $where) >= 0;

    }

    if($res) {

        echo "1"; // Éxito

    }

}

    // Elimina un método de pago por ID

function ajax\_deletePagos() {

    $id = $this->input->post('id');      // ID del método a eliminar

    $where['id'] = $id;

    if($this->Conexion->eliminar('empresa\_metodos\_pago', $where)) {

        echo "1"; // Éxito

    }

}

//////////// R E Q U E R I M I E N T O S ////////////

// Vista del evaluador predeterminado de requerimientos

function evaluador\_requerimiento() {

    $this->load->view('header');                                           // Carga el header

    $this->load->view('configuracion/requerimientos/evaluador');          // Vista del evaluador

}

// Obtiene el evaluador predeterminado desde la configuración

function ajax\_getEvaluador() {

    $query = "SELECT evaluador\_default FROM requerimientos\_config WHERE id = 1";

    $res = $this->Conexion->consultar($query, TRUE);

    echo json\_encode($res); // Devuelve resultado en formato JSON

}

// Establece un nuevo evaluador predeterminado

function ajax\_setEvaluador() {

    $datos['evaluador\_default'] = $this->input->post('id'); // ID del nuevo evaluador

    // Actualiza el campo evaluador\_default donde id = 1

    $res = $this->Conexion->modificar('requerimientos\_config', $datos, null, array('id' => 1)) >= 0;

    if($res) {

        echo "1"; // Éxito

    }

}

// Guarda o actualiza preferencias de notificaciones del usuario

function guardarNot() {

    $query = "SELECT \* FROM notificaciones WHERE idus = " . $this->session->id;

    $res = $this->Conexion->consultar($query);

    if (empty($res)) {

        // Si no existen preferencias, se insertan

        $not = array(

            'idUs' => $this->session->id,

            'qr' => $this->input->post('qr'),

            'tickets' => $this->input->post('tickets'),

            'pr' => $this->input->post('pr'),

            'po' => $this->input->post('po'),

            'cotizaciones' => $this->input->post('cot'),

            'facturas' => $this->input->post('fact'),

            'agenda' => $this->input->post('agenda'),

            'tool' => $this->input->post('tool'),

        );

        $this->privilegios\_model->guardarNot($not);

    } else {

        // Si ya existen, se actualizan

        $not = array(

            'qr' => $this->input->post('qr'),

            'tickets' => $this->input->post('tickets'),

            'pr' => $this->input->post('pr'),

            'po' => $this->input->post('po'),

            'cotizaciones' => $this->input->post('cot'),

            'facturas' => $this->input->post('fact'),

            'agenda' => $this->input->post('agenda'),

            'tool' => $this->input->post('tool'),

        );

        $this->privilegios\_model->updateguardarNot($not);

    }

    // Redirige a la vista de notificaciones

    redirect(base\_url('configuracion/notificaiones'));

}

// Inserta o actualiza un texto de correo

function ajax\_setTexto\_correo() {

    $id = 0;

    if($this->input->post('id')) {

        $id = $this->input->post('id'); // ID del texto a modificar

    }

    $datos['texto'] = $this->input->post('texto');      // Contenido del texto

    $datos['activo'] = $this->input->post('activo');    // Estado (activo/inactivo)

    $datos['id\_us'] = $this->session->id;               // ID del usuario que lo modifica

    $res = FALSE;

    if($id == 0) {

        // Inserta nuevo texto

        $res = $this->Conexion->insertar('texto\_correo', $datos) > 0;

    } else {

        // Actualiza texto existente

        $where['id'] = $id;

        $res = $this->Conexion->modificar('texto\_correo', $datos, null, $where) >= 0;

    }

    if($res) {

        echo "1"; // Éxito

    }

}

// "Elimina" un texto de correo marcándolo como inactivo

function ajax\_deleteTexto() {

    $id = $this->input->post('id');      // ID del texto a desactivar

    $where['id'] = $id;

    $datos['activo'] = 0;                // Lo marca como inactivo

    if($this->Conexion->modificar('texto\_correo', $datos, null, $where)) {

        echo "1"; // Éxito

    }

}

## Correo.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

// Controlador para pruebas de envío de correo

class Correo extends CI\_Controller {

    // Constructor vacío

    function \_\_construct() {

        parent::\_\_construct();

    }

    // Prueba de envío de correo usando librería 'correos'

    function index() {

        $this->load->library('correos');                 // Carga librería personalizada

        $datos['titulo'] = 'Mantenimiento de Rutina';    // Título del ticket

        $datos['id'] = 'AT000023';                       // ID del ticket

        $datos['prefijo'] = 'AT';                        // Prefijo del ticket

        $datos['fecha'] = '17-Ene-2019';                 // Fecha de creación

        $datos['usuario'] = 'ALEJANDRO ORTIZ';           // Usuario responsable

        $this->correos->creacionTicket($datos);          // Envía correo de creación de ticket

    }

    // Prueba de múltiples configuraciones SMTP y envío con MailJet

    function prueba() {

        // Configuración para Gmail (no usada)

        $configGOOGLE = Array(

            'smtp\_host' => 'smtp.gmail.com',

            'smtp\_port' => '465',

            'smtp\_user' => 'aleksrocknlove@gmail.com',

            'smtp\_pass' => 'Alekssdr14',

            'mailtype'  => 'html',

            'newline'   => '\r\n',

            'crlf'      => '\r\n',

            'protocol'  => 'smtp',

        );

        // Configuración para GoDaddy/masmetrologia (no usada)

        $configMAS = Array(

            'smtp\_host' => 'smtpout.secureserver.net',

            'smtp\_port' => '80',

            'smtp\_user' => 'tickets@masmetrologia.mx',

            'smtp\_pass' => 'Soporte2018@',

            'mailtype'  => 'html',

            'newline'   => '\r\n',

            'crlf'      => '\r\n',

            'protocol'  => 'smtp',

        );

        // Configuración para MailJet (usada en esta prueba)

        $configMJET = Array(

            'smtp\_host' => 'in-v3.mailjet.com',

            'smtp\_port' => '587',

            'smtp\_user' => '1f55d1e5c5da7c2c10ee96e0a5d166af',

            'smtp\_pass' => '0c68495c162a80a883412b1106045cb3',

            'mailtype'  => 'html',

            'newline'   => '\r\n',

            'crlf'      => '\r\n',

            'protocol'  => 'smtp',

        );

        $this->load->library('email', $configMJET); // Usa configuración de MailJet

        $this->email->from('aortiz@masmetrologia.com', 'PRUEBA MAIL-JET'); // Remitente

        $this->email->to('alejandro\_ortiz426@hotmail.com');                // Destinatario

        $logo = base\_url('template/images/logo.png');

        // Mensaje con logotipo (comentado)

        $mensaje = <<<EOD

        <a href='#'><img width='400' src='$logo'><br></a>

        <h1><font face="Arial">SIGA-MAS</font></h1>

EOD;

        $this->email->subject('SIGA MAILJET');

        //$this->email->message($mensaje); // Envío de mensaje con HTML

        $this->email->message("PRUEBA MAIL-JET"); // Mensaje de prueba (texto simple)

        $this->email->send();

        echo $this->email->print\_debugger(); // Muestra resultado del envío

    }

    // Envío de correo con formato de ticket de servicio

    public function ticket($datos) {

        $logo = base\_url('template/images/logo.png');   // Ruta del logotipo

        $titulo = $datos['titulo'];                     // Título del ticket

        // Cuerpo del correo en HTML

        $mensaje = <<<EOD

               <img width='500' src='$logo'>

               <h1>Ticket de Servicio</h1>

               <h3>Titulo: $titulo</h3>

EOD;

        $this->load->library('email');

        $this->email->from('alejandro\_ortiz426@hotmail.com', 'Aleks Ortiz'); // Remitente

        $this->email->to('aortiz@masmetrologia.com');                         // Destinatario

        $this->email->subject('EXIaaaaaTO'); // Asunto del correo

        $this->email->message($mensaje);     // Cuerpo HTML

        $this->email->send(); // Enviar

        echo $this->email->print\_debugger(); // Muestra resultado del envío

    }

}

## Cotizaciones.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Cotizaciones extends CI\_Controller {

public $idsub=null;

// Constructor de la clase

function \_\_construct() {

parent::\_\_construct();

$this->load->library('correos\_cotizaciones');

$this->load->model('privilegios\_model');

}

// Vista principal del catálogo de cotizaciones

function index($estatus = 'TODO', $user = "") {

$us = null;

$data['estatus'] = strtoupper($estatus); // Convierte el estatus a mayúsculas

$data['user'] = strtoupper($user); // Convierte el nombre de usuario a mayúsculas

$this->load->model('inicio\_model');

$this->load->view('header');

$this->load->view('cotizaciones/catalogo', $data);

}

// Carga el formulario para crear una nueva cotización

function crear\_cotizacion() {

$usd = $this->aos\_funciones->getUSD(); // Obtiene tipos de cambio USD

$data["id"] = 0;

$data['USD'] = $usd[0];

$data['USD\_ACT'] = $usd[1];

$data['COPY'] = 0;

if (isset($\_POST["id"])) {

$id = $this->input->post('id');

$rev = $this->input->post('rev');

$data['COPY'] = $id . '-' . $rev; // Marca como copia de una cotización existente

}

$this->load->view('header');

$this->load->view('cotizaciones/generar', $data);

}

// Carga una cotización específica en modo visualización/edición

function ver\_cotizacion($id = 0) {

$this->load->model('usuarios\_model');

$data['sub'] = $this->usuarios\_model->userCots(); // Obtiene cotizaciones del usuario

$data['COPY'] = 0;

$usd = $this->aos\_funciones->getUSD();

$data['USD'] = $usd[0];

$data['USD\_ACT'] = $usd[1];

if (isset($\_POST["id"])) {

$id = $this->input->post('id'); // Prioriza el ID enviado por POST

} else if ($id == 0) {

redirect(base\_url('inicio')); // Redirige si no hay ID válido

}

$data["id"] = $id;

$this->load->view('header');

$this->load->view('cotizaciones/generar', $data);

}

// Muestra el dashboard de cotizaciones

function dashboard() {

$this->load->model('usuarios\_model');

$datos['sub'] = $this->usuarios\_model->userCots(); // Carga cotizaciones del usuario

$this->load->view('header');

$this->load->view('cotizaciones/dashboard', $datos);

}

function cotizacion\_pdf($param, $modo = 'I') {

// Separa el parámetro compuesto (id-rev)

$param = explode('-', $param);

$id = $param[0];

$rev = $param[1];

// Construcción del query SQL para obtener los datos de la cotización y conceptos

$query = "SELECT

(SELECT ifnull(max(CC2.revision), 0)

FROM cotizaciones\_conceptos CC2

WHERE CC2.cotizacion = $id) as UltRev,

C.fecha, C.moneda, C.tipo, C.impuesto\_factor, C.impuesto\_nombre,

C.aprobador, C.estatus, E.razon\_social, E.calle, E.numero,

E.numero\_interior, E.colonia, E.ciudad, E.estado, E.pais, E.rfc,

EC.nombre, EC.telefono, EC.correo,

(if(E.credito\_cliente = 1, concat(E.credito\_cliente\_plazo, ' Días'), 'Contado')) as Credito,

concat(R.nombre, ' ', R.paterno) as Resp, R.correo as RespCorreo,

CC.\*, C.planta,

ifnull(EP.nombre, 'N/A') as PlantaNombre,

ifnull(EP.calle, 'N/A') as PlantaCalle,

ifnull(EP.colonia, 'N/A') as PlantaColonia,

ifnull(EP.ciudad, 'N/A') as PlantaCiudad,

ifnull(EP.estado, 'N/A') as PlantaEstado

FROM cotizaciones\_conceptos CC

INNER JOIN cotizaciones C ON C.id = CC.cotizacion

LEFT JOIN empresa\_plantas EP ON EP.id = C.planta

INNER JOIN empresas E ON E.id = C.empresa

INNER JOIN empresas\_contactos EC ON EC.id = C.contactos->'$[0]'

INNER JOIN usuarios R ON R.id = C.responsable

WHERE CC.cotizacion = $id AND CC.revision = $rev";

// Ejecuta la consulta

$res = $this->Conexion->consultar($query);

// Inicialización de variables utilizadas en el PDF

$conceptos = [];

$APROB = -1;

$ESTATUS = "";

$SERVICIOS = new stdClass;

$SUBTOTAL = 0;

$IMPUESTO = 0;

$TOTAL = 0;

foreach ($res as $i => $elem) {

// Determina si la revisión es obsoleta

$OBS = $elem->UltRev != $rev ? " OBSOLETA" : "";

// Captura información general de la cotización

$APROB = $elem->aprobador;

$ESTATUS = $elem->estatus;

$COT = 'COT-' . str\_pad($elem->cotizacion, 6, "0", STR\_PAD\_LEFT) . " Rev: " . $rev . $OBS;

$FECHA = date\_format(date\_create($elem->fecha), 'd/m/Y h:i A');

$CLIENTE = $elem->razon\_social;

$DOMICILIO = $elem->calle . ' ' . $elem->numero . ' ' . $elem->numero\_interior;

$COLONIA = $elem->colonia;

$RFC = $elem->rfc;

// Datos de contacto

$CONTACTO = $elem->nombre;

$TELEFONO = $elem->telefono;

$CORREO = $elem->correo;

$UBICACION = $elem->ciudad . ', ' . $elem->estado . ', ' . $elem->pais;

// Responsable de la cotización

$RESPONSABLE = $elem->Resp;

$RESPONSABLE\_CORREO = $elem->RespCorreo;

// Planta asociada

$ID\_PLANTA = $elem->planta;

$NOMBRE\_PLANTA = $elem->PlantaNombre;

$CALLE\_PLANTA = $elem->PlantaCalle;

$COLONIA\_PLANTA = $elem->PlantaColonia;

$CIUDAD\_PLANTA = $elem->PlantaCiudad;

$ESTADO\_PLANTA = $elem->PlantaEstado;

// Construcción del objeto concepto

$concepto = new stdClass;

$concepto->cantidad = $elem->cantidad;

$concepto->atributos = json\_decode($elem->atributos);

$concepto->descripcion = $elem->descripcion;

$concepto->servicios = json\_decode($elem->servicios);

// Acumula servicios únicos

foreach ($concepto->servicios as $key => $value) {

$cod = $value[1];

if ($cod != "N/A") {

$SERVICIOS->$cod = $value[2];

}

}

// Campos adicionales del concepto

$concepto->comentarios = $elem->comentarios;

$concepto->tiempo\_entrega = $elem->tiempo\_entrega;

$concepto->sitio = $elem->sitio;

$concepto->precio\_unitario = $elem->precio\_unitario;

$concepto->importe = floatval($elem->precio\_unitario) \* floatval($elem->cantidad);

// Agrega el concepto al arreglo

array\_push($conceptos, $concepto);

// Cálculos de totales

$IMPUESTO\_NOMBRE = "IVA / TAX [" . ($elem->impuesto\_factor) \* 100 . "%]";

$MONEDA = $elem->moneda;

$CREDITO = $elem->Credito;

$SUBTOTAL += $concepto->importe;

$IMPUESTO += $concepto->importe \* ($elem->impuesto\_factor);

$TOTAL += $concepto->importe \* ($elem->impuesto\_factor + 1);

// Inicializa notas y margen por tipo de servicio

$NOTAS = "TERMINOS Y CONDICIONES:";

switch ($elem->tipo) {

case 'CALIBRACION':

$Margin = 195;

$NOTAS .= ''

. "\n1.- El servicio se programa con la orden de compra correspondiente. Si su orden de compra incluye más de un equipo se requiere que los mismos se calibren dentro de un periodo de 30 días máximo."

. "\n2.- La Cotización es válida por 120 días a partir de la fecha de elaboración."

. "\n3.- El servicio de calibración descrito se ofrece usando los métodos y procedimientos internos del laboratorio. Si existen requisitos específicos del cliente, se iniciará nuevamente el proceso de cotización."

. "\n4.- El costo del servicio es aplicable aun cuando su equipo no pase la calibración o no responda al proceso de ajuste. En ese caso, se entregará un reporte detallado."

. "\n5.- No hay garantía de que el equipo mantendrá las tolerancias especificadas durante el intervalo de calibración."

. "\n6.- Identificación y frecuencia de calibración son asignadas por el cliente."

. "\n7.- El tiempo de entrega depende de la programación y empieza tras la aprobación y recepción física del instrumento."

. "\n8.- Si el cliente no proporciona regla de decisión, nuestros certificados no consideran la incertidumbre salvo que se indique lo contrario."

. "\nESTE SERVICIO CONSTA DE:"

. "\n1.- Revisión y limpieza externa del equipo."

. "\n2.- Calibración del equipo.";

break;

case 'ESTUDIO DIMENSIONAL':

$Margin = 215;

$NOTAS .= ''

. "\n1.- Se requiere orden de compra para iniciar el servicio. Una vez recibida, se confirma la fecha de entrega."

. "\n2.- Generar la orden de compra con base en esta cotización y sus políticas internas."

. "\n3.- Cotización válida por 120 días desde la fecha de elaboración."

. "\n4.- Se requieren planos legibles."

. "\n5.- Puede ser necesario seccionar piezas; proveer cantidad suficiente.";

break;

case 'RENTA':

$Margin = 215;

$NOTAS .= ''

. "\n1.- Se requiere orden de compra para iniciar el servicio."

. "\n2.- Generar la orden de compra conforme a esta cotización y políticas internas."

. "\n3.- Cotización válida por 30 días desde la fecha de elaboración."

. "\nESTE SERVICIO CONSTA DE:"

. "\n1.- Entrega del equipo en planta."

. "\n2.- Instrucción sobre el uso correcto del equipo."

. "\n3.- Instalación en sitio (solo básculas y contadores).";

break;

case 'REPARACION':

$Margin = 215;

$NOTAS .= ''

. "\n1.- Se requiere orden de compra para iniciar el servicio."

. "\n2.- Generar la orden de compra conforme a esta cotización y políticas internas."

. "\n3.- Cotización válida por 15 días desde la fecha de elaboración.";

break;

case 'VENTA':

$Margin = 215;

$NOTAS .= ''

. "\n1.- Generar la orden de compra conforme a esta cotización y políticas internas."

. "\n2.- Cotización válida por 30 días desde la fecha de elaboración."

. "\n3.- Pedido no cancelable."

. "\n4.- Disponibilidad sujeta a existencias si no hay orden de compra."

. "\n5.- Se requiere orden de compra por escrito para entrega."

. "\n6.- Al recibir la orden, se confirma la fecha de entrega.";

break;

case 'SOPORTE':

$Margin = 215;

$NOTAS .= ''

. "\n1.- Se requiere orden de compra para iniciar el servicio."

. "\n2.- Generar la orden de compra conforme a esta cotización y políticas internas."

. "\n3.- Cotización válida por 30 días desde la fecha de elaboración.";

break;

case 'CALIBRACION EXTERNA':

$Margin = 205;

$NOTAS .= ''

. "\n1.- El servicio se programa con orden de compra y equipo disponible para Metrología Aplicada."

. "\n2.- Cotización válida por 30 días desde la fecha de elaboración."

. "\n3.- Si el cliente solicita otro proveedor, se iniciará nuevo proceso de cotización."

. "\n4.- Costo aplicable incluso si el equipo no pasa calibración."

. "\n5.- No hay garantía de estabilidad en tolerancias por factores externos."

. "\n6.- Tiempos de entrega pueden cambiar, se notificará con anticipación."

. "\n7.- El tiempo empieza a correr desde la aprobación y recepción del equipo."

. "\nESTE SERVICIO CONSTA DE:"

. "\n1.- Revisión y limpieza externa del equipo."

. "\n2.- Calibración del equipo.";

break;

case 'MAPEO':

$Margin = 215;

$NOTAS .= ''

. "\n1.- El servicio se programa con orden de compra."

. "\n2.- El equipo debe estar disponible en planta para el servicio."

. "\n3.- La cotización cubre el tiempo estimado. Excesos se cotizan por separado."

. "\n4.- Cambios en requerimientos se cotizan y requieren autorización."

. "\n5.- Cotización válida por 60 días desde su elaboración.";

break;

case 'LISTA PRECIOS':

$Margin = 210;

$NOTAS .= ''

. "\n1.- Se requiere orden de compra para programar el servicio."

. "\n2.- Calibración según procedimientos internos. Requerimientos especiales implican nueva cotización."

. "\n3.- Costo aplicable aunque el equipo no pase calibración."

. "\n4.- No se garantiza estabilidad del equipo debido a factores externos."

. "\n5.- Vigencia establecida en contrato o acuerdo correspondiente."

. "\nESTE SERVICIO CONSTA DE:"

. "\n1.- Revisión y limpieza externa del equipo."

. "\n2.- Calibración del equipo.";

break;

}

}

// Oculta errores para evitar que interfieran con la salida del PDF

ini\_set('display\_errors', 0);

// Carga la librería personalizada para generar PDF

$this->load->library('pdfview');

// Selección de plantilla según estatus de la cotización

if ($ESTATUS == "CANCELADA") {

$pdf = new pdfview\_CANCEL(PDF\_PAGE\_ORIENTATION, PDF\_UNIT, PDF\_PAGE\_FORMAT, true, 'UTF-8', false);

} else {

if ($APROB > 0) {

$pdf = new pdfview(PDF\_PAGE\_ORIENTATION, PDF\_UNIT, PDF\_PAGE\_FORMAT, true, 'UTF-8', false);

} else {

$pdf = new pdfview\_NA(PDF\_PAGE\_ORIENTATION, PDF\_UNIT, PDF\_PAGE\_FORMAT, true, 'UTF-8', false);

}

}

// Configuración general del PDF

$pdf->SetCreator(PDF\_CREATOR);

$pdf->SetAuthor('AleksOrtiz');

$pdf->SetTitle('Masmetrologia');

$pdf->SetSubject('Formato Cotización');

// Texto de cabecera personalizado

$spc = " ";

$head = "Metrologia Aplicada y Servicios, S. de R.L. de C.V. $spc Cotización / $COT";

$txt = "Av. Ramón Rayón #1520 Int-9 Fecha: " . $FECHA;

$txt .= "\nCol. Rio Bravo Ejecutivo: " . $RESPONSABLE;

$txt .= "\nCd. Juárez, Chih. C.P. 32550 Correo: " . $RESPONSABLE\_CORREO;

$txt .= "\nRFC: MAS080825EE7";

$pdf->SetHeaderData(PDF\_HEADER\_LOGO\_ORIGINAL, '40', $head, $txt);

// Fuentes y márgenes

$pdf->setHeaderFont(Array(PDF\_FONT\_NAME\_MAIN, '', 9));

$pdf->setFooterFont(Array(PDF\_FONT\_NAME\_DATA, '', PDF\_FONT\_SIZE\_DATA));

$pdf->SetDefaultMonospacedFont(PDF\_FONT\_MONOSPACED);

$pdf->SetMargins(8, PDF\_MARGIN\_TOP, 8); // Márgenes modificados manualmente

$pdf->SetHeaderMargin(PDF\_MARGIN\_HEADER);

$pdf->SetFooterMargin(PDF\_MARGIN\_FOOTER);

$pdf->SetAutoPageBreak(TRUE, PDF\_MARGIN\_BOTTOM);

$pdf->setImageScale(PDF\_IMAGE\_SCALE\_RATIO);

// Carga idioma si existe

if (@file\_exists(dirname(\_\_FILE\_\_) . '/lang/eng.php')) {

require\_once(dirname(\_\_FILE\_\_) . '/lang/eng.php');

$pdf->setLanguageArray($l);

}

// Fuente por defecto y nueva página

$pdf->SetFont('times', '', 8);

$pdf->AddPage();

// Construcción de la tabla de cabecera con datos del cliente

$tbl = <<<EOD

<table border="0">

<tr>

<td>

<b>Cliente/Client:</b><br>

$CLIENTE<br><br>

<b>Dirección / Address:</b><br>

$DOMICILIO<br>

$COLONIA<br>

$UBICACION<br>

RFC: $RFC<br>

</td>

<td>

EOD;

// Si existe planta asociada, incluir sus datos

if ($ID\_PLANTA > 0) {

$tbl .= <<<EOD

<b>Planta/Plant:</b><br>

$NOMBRE\_PLANTA<br>

$CALLE\_PLANTA<br>

$COLONIA\_PLANTA<br>

$CIUDAD\_PLANTA $ESTADO\_PLANTA<br><br>

EOD;

}

// Sección de contacto

$tbl .= <<<EOD

<b>Contacto / Contact:</b><br>

$CONTACTO<br>

$TELEFONO<br>

$CORREO<br>

</td>

</tr>

</table>

EOD;

// Imprime la tabla de cabecera

$pdf->writeHTML($tbl, false, false, false, false, '');

// Ancho de columnas para tabla de conceptos

$w = array(8, 125, 12, 24, 24);

// Encabezado de tabla de conceptos

$pdf->SetFont('helvetica', 'B', 9);

$pdf->SetTextColor(255); // Blanco

$pdf->SetTextColor(0); // Negro

// Nombres de columnas

$pdf->Cell($w[0], 6, "#", 1, 0, 'C');

$pdf->Cell($w[1], 6, "Descripción", 1, 0, 'C');

$pdf->Cell($w[2], 6, "Cant.", 1, 0, 'C');

$pdf->Cell($w[3], 6, "Precio Unit.", 1, 0, 'C');

$pdf->Cell($w[4], 6, "Importe", 1, 1, 'C');

// Fuente base para los conceptos

$pdf->SetFont('', '', 7);

// Inicializa índice de conceptos

$i = 0;

// Altura por renglón base para celdas

$RenSpace = 2.8;

foreach ($conceptos as $indice => $concepto) {

$pdf->SetFont('', '', 7);

// Si el concepto tiene atributo "otro" (CABEZAL o SEPARADOR)

if (array\_key\_exists("otro", $concepto->atributos)) {

// Espacio en blanco para separación visual

for ($j = 0; $j < 5; $j++) {

$pdf->writeHTMLCell($w[$j], 0, '', '', '', "LR", ($j == 4), 0, true, 'J', false);

}

$startX = $pdf->GetX();

$startY = $pdf->GetY();

$cellcount = [];

if ($concepto->atributos->otro == 'CABEZAL') {

$pdf->SetFont('', 'B', 8);

$pdf->MultiCell($w[0], '', '', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[1], '', $concepto->descripcion, 'LR', 'C', 0, 0);

$pdf->MultiCell($w[2], '', '', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[3], '', '', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[4], '', '', 'LR', 'C', 0, 1);

$h = (max($cellcount)) \* $RenSpace;

$pdf->SetXY($startX, $startY);

for ($j = 0; $j < 5; $j++) {

$pdf->writeHTMLCell($w[$j], $h, '', '', '', "LR", ($j == 4), 0, true, 'J', false);

}

$pdf->SetFont('', '', 7);

} elseif ($concepto->atributos->otro == 'SEPARADOR') {

$pdf->MultiCell($w[0], '', '===', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[1], '', str\_repeat('=', 84), 'LR', 'C', 0, 0);

$pdf->MultiCell($w[2], '', '=====', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[3], '', str\_repeat('=', 12), 'LR', 'C', 0, 0);

$pdf->MultiCell($w[4], '', str\_repeat('=', 12), 'LR', 'C', 0, 1);

$h = (max($cellcount)) \* $RenSpace;

$pdf->SetXY($startX, $startY);

for ($j = 0; $j < 5; $j++) {

$pdf->writeHTMLCell($w[$j], $h, '', '', '', "LR", ($j == 4), 0, true, 'J', false);

}

}

} else {

// Concepto estándar

$i++;

// Espacio inicial

for ($j = 0; $j < 5; $j++) {

$pdf->writeHTMLCell($w[$j], 0, '', '', '', "LR", ($j == 4), 0, true, 'J', false);

}

// Sitio y tiempo de entrega

$dicSitios = ['OS' => 'En Planta', 'LAB' => 'LAB', 'EXT' => 'Prov. Externo'];

$sitio = $concepto->sitio != "N/A" ? ", Donde: " . $dicSitios[$concepto->sitio] : "";

$tEntrega = $concepto->tiempo\_entrega > 0

? "T. Entrega: {$concepto->tiempo\_entrega} Día" . ($concepto->tiempo\_entrega > 1 ? "s" : "") . $sitio

: "T. Entrega: N/A" . $sitio;

// Si tiene servicios: mostrarlos con descripción extendida

if (count($concepto->servicios) > 0) {

$codigos = "";

foreach ($concepto->servicios as $value) {

$codigos .= ($value[1] == "N/A" ? $value[2] : $value[1]) . ", ";

}

$startX = $pdf->GetX();

$startY = $pdf->GetY();

$cellcount = [];

$cellcount[] = $pdf->MultiCell($w[0], '', $indice + 1, 0, 'C', 0, 0);

$cellcount[] = $pdf->MultiCell($w[1], '', "Servicios: $codigos $tEntrega", 0, 'L', 0, 0);

$cellcount[] = $pdf->MultiCell($w[2], '', number\_format($concepto->cantidad, 0), 0, 'C', 0, 0);

$cellcount[] = $pdf->MultiCell($w[3], '', "$" . number\_format($concepto->precio\_unitario, 2), 0, 'R', 0, 0);

$cellcount[] = $pdf->MultiCell($w[4], '', "$" . number\_format($concepto->importe, 2), 0, 'R', 0, 1);

$h = max($cellcount) \* $RenSpace;

$pdf->SetXY($startX, $startY);

for ($j = 0; $j < 5; $j++) {

$pdf->writeHTMLCell($w[$j], $h, '', '', '', "LR", ($j == 4), 0, true, 'J', false);

}

}

// Datos del equipo desde los atributos

$att = "";

foreach (['ID', 'Marca', 'Modelo', 'Serie'] as $key) {

if (isset($concepto->atributos->$key)) {

$att .= "$key: {$concepto->atributos->$key}, ";

}

}

foreach ($concepto->atributos as $key => $value) {

if (!in\_array($key, ['ID', 'Marca', 'Modelo', 'Serie'])) {

$att .= "$key: $value, ";

}

}

$att = rtrim($att, ', ');

$startX = $pdf->GetX();

$startY = $pdf->GetY();

$cellcount = [];

$cellcount[] = $pdf->MultiCell($w[0], '', (count($concepto->servicios) > 0 ? "" : $indice + 1), 0, 'C', 0, 0);

$cellcount[] = $pdf->MultiCell($w[1], '', $concepto->descripcion . (count($concepto->servicios) > 0 ? " $att" : " $tEntrega"), 0, 'L', 0, 0);

$cellcount[] = $pdf->MultiCell($w[2], '', (count($concepto->servicios) > 0 ? "" : $concepto->cantidad), 0, 'C', 0, 0);

$cellcount[] = $pdf->MultiCell($w[3], '', (count($concepto->servicios) > 0 ? "" : "$" . number\_format($concepto->precio\_unitario, 2)), 0, 'R', 0, 0);

$cellcount[] = $pdf->MultiCell($w[4], '', (count($concepto->servicios) > 0 ? "" : "$" . number\_format($concepto->importe, 2)), 0, 'R', 0, 1);

$h = max($cellcount) \* $RenSpace;

$pdf->SetXY($startX, $startY);

for ($j = 0; $j < 5; $j++) {

$pdf->writeHTMLCell($w[$j], $h, '', '', '', "LR", ($j == 4), 0, true, 'J', false);

}

// Comentarios del concepto

if (!empty($concepto->comentarios)) {

$startX = $pdf->GetX();

$startY = $pdf->GetY();

$cellcount = [];

$cellcount[] = $pdf->MultiCell($w[0], '', "", 0, 'C', 0, 0);

$cellcount[] = $pdf->MultiCell($w[1], '', "\*\* " . $concepto->comentarios, 0, 'L', 0, 0);

for ($j = 2; $j < 5; $j++) {

$cellcount[] = $pdf->MultiCell($w[$j], '', "", 0, 'R', 0, 0);

}

$h = max($cellcount) \* $RenSpace;

$pdf->SetXY($startX, $startY);

for ($j = 0; $j < 5; $j++) {

$pdf->writeHTMLCell($w[$j], $h, '', '', '', "LR", ($j == 4), 0, true, 'J', false);

}

}

// Salto de página si se rebasa el límite vertical

if ($pdf->GetY() > 260) {

for ($j = 0; $j < 5; $j++) {

$pdf->writeHTMLCell($w[$j], '', '', '', '', "LRB", ($j == 4), 0, true, 'J', false);

}

$pdf->AddPage();

$pdf->SetFont('times', '', 8);

$pdf->writeHTML($tbl, false, false, false, false, '');

if ($i < count($conceptos)) {

$pdf->SetFont('helvetica', 'B', 9);

$pdf->SetTextColor(0);

$pdf->Cell($w[0], 6, "#", 1, 0, 'C');

$pdf->Cell($w[1], 6, "Descripción", 1, 0, 'C');

$pdf->Cell($w[2], 6, "Cant.", 1, 0, 'C');

$pdf->Cell($w[3], 6, "Precio Unit.", 1, 0, 'C');

$pdf->Cell($w[4], 6, "Importe", 1, 1, 'C');

for ($j = 0; $j < 5; $j++) {

$pdf->writeHTMLCell($w[$j], '', '', '', '', "LRT", ($j == 4), 0, true, 'J', false);

}

}

}

}

}

// DESCRIPCIÓN DE CÓDIGOS DE SERVICIO (solo si existen servicios únicos)

if (count((array)$SERVICIOS) > 0) {

// Línea separadora visual

$pdf->MultiCell($w[0], '', '===', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[1], '', str\_repeat('=', 84), 'LR', 'C', 0, 0);

$pdf->MultiCell($w[2], '', '=====', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[3], '', str\_repeat('=', 12), 'LR', 'C', 0, 0);

$pdf->MultiCell($w[4], '', str\_repeat('=', 12), 'LR', 'C', 0, 1);

// Encabezado de sección

$pdf->SetFont('', 'B', 7);

$pdf->MultiCell($w[0], '', '', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[1], '', 'Descripción de Servicios:', 0, 'L', 0, 0);

$pdf->MultiCell($w[2], '', '', 'LR', 'R', 0, 0);

$pdf->MultiCell($w[3], '', '', 'LR', 'R', 0, 0);

$pdf->MultiCell($w[4], '', '', 'LR', 'R', 0, 1);

// Cuerpo de descripción de servicios

$pdf->SetFont('', '', 7);

foreach ($SERVICIOS as $codigo => $servicio) {

$startX = $pdf->GetX();

$startY = $pdf->GetY();

$cellcount = [];

// Calcula número de líneas necesarias por celda

$cellcount[] = $pdf->MultiCell($w[0], '', '', 0, 'C', 0, 0);

$cellcount[] = $pdf->MultiCell($w[1], '', "$codigo: $servicio", 0, 'L', 0, 0);

$cellcount[] = $pdf->MultiCell($w[2], '', '', 0, 'R', 0, 0);

$cellcount[] = $pdf->MultiCell($w[3], '', '', 0, 'R', 0, 0);

$cellcount[] = $pdf->MultiCell($w[4], '', '', 0, 'R', 0, 0);

// Regresa cursor y escribe celdas con altura unificada

$pdf->SetXY($startX, $startY);

$h = (max($cellcount) + 1) \* $RenSpace;

$pdf->MultiCell($w[0], $h, '', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[1], $h, '', 'LR', 'L', 0, 0);

$pdf->MultiCell($w[2], $h, '', 'LR', 'R', 0, 0);

$pdf->MultiCell($w[3], $h, '', 'LR', 'R', 0, 0);

$pdf->MultiCell($w[4], $h, '', 'LR', 'R', 0, 1);

}

}

// Rellena hasta el margen para separar los totales

$startY = $pdf->GetY();

for ($startY; $startY < $Margin; $startY += 6) {

$pdf->MultiCell($w[0], 6, '----', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[1], 6, str\_repeat('-', 120), 'LR', 'C', 0, 0);

$pdf->MultiCell($w[2], 6, '-------', 'LR', 'C', 0, 0);

$pdf->MultiCell($w[3], 6, str\_repeat('-', 19), 'LR', 'C', 0, 0);

$pdf->MultiCell($w[4], 6, str\_repeat('-', 19), 'LR', 'C', 0, 0);

$pdf->Ln();

}

// Notas de servicio

$pdf->MultiCell($w[0] + $w[1] + $w[2], 6, $NOTAS, 'T', 'M', 0, 0);

// Subtotal

$pdf->SetFont('', '', 8);

$pdf->MultiCell($w[3], 4, 'Sub-Total', 'T', 'C', 0, 0);

$pdf->MultiCell($w[4], 4, "$" . number\_format($SUBTOTAL, 2), 1, 'R', 0, 0);

$pdf->Ln();

// Impuesto

$pdf->MultiCell($w[0] + $w[1] + $w[2], 6, '', 0, 'L', 0, 0);

$pdf->MultiCell($w[3], 4, $IMPUESTO\_NOMBRE, 0, 'C', 0, 0);

$pdf->MultiCell($w[4], 4, "$" . number\_format($IMPUESTO, 2), 1, 'R', 0, 0);

$pdf->Ln();

// Total

$pdf->MultiCell($w[0] + $w[1] + $w[2], 6, '', 0, 'M', 0, 0);

$pdf->MultiCell($w[3], 4, 'Total', 0, 'C', 0, 0);

$pdf->MultiCell($w[4], 4, "$" . number\_format($TOTAL, 2), 1, 'R', 0, 0);

$pdf->Ln();

// Moneda

$pdf->MultiCell($w[0] + $w[1] + $w[2], 6, '', 0, 'M', 0, 0);

$pdf->MultiCell($w[3] + $w[4], 6, '\* Moneda / Currency: ' . $MONEDA, 0, 'C', 0, 0);

$pdf->Ln();

// Espacios de separación

$pdf->MultiCell(180, 6, '', 0, 'M', 0, 0);

$pdf->Ln();

$pdf->MultiCell(180, 6, '', 0, 'M', 0, 0);

$pdf->Ln();

$pdf->Ln();

// Salida del PDF: como string ('S') o mostrar en navegador ('I')

if ($modo === 'S') {

return $pdf->Output('', 'S'); // Devuelve como string (por ejemplo, para adjuntar a correo)

} else {

$cot = str\_pad($id, 6, '0', STR\_PAD\_LEFT) . '\_Rev\_' . $rev;

$pdf->Output("COT-$cot.pdf", 'I'); // Muestra en el navegador

}

}

function ajax\_setCotizacion() {

$id\_cotizacion = null;

$id\_cotizacion\_nueva = null;

// Decodifica los objetos JSON recibidos por POST

$cotizacion = json\_decode($this->input->post("cotizacion"));

$conceptos = json\_decode($this->input->post("conceptos"));

// Si se copia desde una cotización existente, guarda el ID de origen

if (isset($cotizacion->copiar\_desde)) {

$id\_cotizacion = $cotizacion->copiar\_desde;

}

// Si es una nueva cotización

if ($cotizacion->id == 0) {

$cotizacion->usuario = $this->session->id;

// Inserta la nueva cotización con timestamp automático

$cotizacion->id = $this->Conexion->insertar(

'cotizaciones',

$cotizacion,

array('fecha' => 'CURRENT\_TIMESTAMP()')

);

$id\_cotizacion\_nueva = $cotizacion->id;

// Verifica si la empresa es un prospecto para marcarlo

$res = $this->Conexion->consultar(

'SELECT prospecto FROM empresas WHERE id = ' . $cotizacion->empresa,

TRUE

);

if ($res->prospecto == '1') {

$prospectoCot['prospectoEmp'] = '1';

$this->Conexion->modificar(

'cotizaciones',

$prospectoCot,

null,

array('id' => $cotizacion->id)

);

}

} else {

// M O D I F I C A C I Ó N D E C O T I Z A C I Ó N

$func = null;

// Si se aprueba, se registra aprobador y fecha

if ($cotizacion->estatus == 'AUTORIZADA') {

$cotizacion->aprobador = $this->session->id;

$func['fecha\_aprobacion'] = 'CURRENT\_TIMESTAMP()';

}

// Si se confirma, se registra la fecha de confirmación

if ($cotizacion->estatus == 'CONFIRMADA') {

$func['fecha\_confirmacion'] = 'CURRENT\_TIMESTAMP()';

}

// Si es aprobado total, actualiza prospecto de la empresa y cancela seguimientos

if ($cotizacion->estatus == 'APROBADO TOTAL') {

$res = $this->Conexion->consultar(

'SELECT prospecto FROM empresas WHERE id = ' . $cotizacion->empresa,

TRUE

);

if ($res->prospecto == '1') {

$empresa['prospecto'] = '0';

$this->Conexion->modificar(

'empresas',

$empresa,

null,

array('id' => $cotizacion->empresa)

);

}

$seg = $this->Conexion->consultar(

"SELECT ca.\*, u.correo

FROM cot\_acciones ca

JOIN usuarios u ON u.id = ca.usuario

WHERE estatus = 'PENDIENTE' AND idCot = " . $cotizacion->id

);

if ($seg) {

foreach ($seg as $value) {

$seguimiento['estatus'] = 'CANCELADA';

$this->Conexion->modificar('cot\_acciones', $seguimiento, null, array('id' => $value->id));

$datos['id'] = $value->idCot;

$datos['fecha\_limite'] = $value->fecha\_limite;

$datos['accion'] = $value->accion;

$datos['correos'] = $value->correo;

$datos['correo\_cancelar'] = $this->session->correo;

$this->correos\_cotizaciones->cancelarSeguimiento($datos);

}

}

}

// Actualiza los datos de la cotización

$this->Conexion->modificar(

'cotizaciones',

$cotizacion,

$func,

array('id' => $cotizacion->id)

);

// Si la cotización fue cancelada, cancela seguimientos relacionados

if ($cotizacion->estatus == 'CANCELADA') {

$seg = $this->Conexion->consultar(

"SELECT ca.\*, u.correo

FROM cot\_acciones ca

JOIN usuarios u ON u.id = ca.usuario

WHERE estatus = 'PENDIENTE' AND idCot = " . $cotizacion->id

);

if ($seg) {

foreach ($seg as $value) {

$seguimiento['estatus'] = 'CANCELADA';

$this->Conexion->modificar('cot\_acciones', $seguimiento, null, array('id' => $value->id));

$datos['id'] = $value->idCot;

$datos['fecha\_limite'] = $value->fecha\_limite;

$datos['accion'] = $value->accion;

$datos['correos'] = $value->correo;

$datos['correo\_cancelar'] = $this->session->correo;

$this->correos\_cotizaciones->cancelarSeguimiento($datos);

}

}

}

// E N V Í O D E C O R R E O S

// Si se envió un comentario desde el formulario

if (isset($\_POST['comentarios']) && $\_POST['comentarios']) {

$datos['comentarios'] = $this->input->post('comentarios');

}

// Si está pendiente de autorización, notifica al autorizador

if ($cotizacion->estatus == 'PENDIENTE AUTORIZACION') {

$res = $this->Conexion->consultar(

"SELECT

E.nombre AS NombreCliente,

EC.nombre AS NombreContacto,

CONCAT(R.nombre, ' ', R.paterno) AS Responsable,

A.correo

FROM cotizaciones C

INNER JOIN usuarios R ON R.id = C.responsable

INNER JOIN usuarios A ON R.autorizador\_cotizacion = A.id

INNER JOIN empresas E ON E.id = C.empresa

INNER JOIN empresas\_contactos EC ON EC.id = C.contactos->'$[0]'

WHERE C.id = $cotizacion->id",

TRUE

);

$datos['id'] = $cotizacion->id;

$datos['correos'] = $res->correo;

$datos['nombreCliente'] = $res->NombreCliente;

$datos['nombreContacto'] = $res->NombreContacto;

$datos['nombreResponsable'] = $res->Responsable;

if (isset($cotizacion->enviar\_autorizar) && $cotizacion->enviar\_autorizar == 1) {

// Lógica adicional si se requiere enviar autorización explícitamente (actualmente vacía)

}

// Envío de solicitud de aprobación

$this->correos\_cotizaciones->solicitarAprobacion($datos);

}

// Si fue rechazada, notificar al responsable

if ($cotizacion->estatus == 'RECHAZADA') {

$res = $this->Conexion->consultar(

"SELECT

E.nombre AS NombreCliente,

EC.nombre AS NombreContacto,

CONCAT(R.nombre, ' ', R.paterno) AS Responsable,

R.correo

FROM cotizaciones C

INNER JOIN usuarios R ON R.id = C.responsable

INNER JOIN empresas E ON E.id = C.empresa

INNER JOIN empresas\_contactos EC ON EC.id = C.contactos->'$[0]'

WHERE C.id = $cotizacion->id",

TRUE

);

$datos['id'] = $cotizacion->id;

$datos['correos'] = $res->correo;

$datos['nombreCliente'] = $res->NombreCliente;

$datos['nombreContacto'] = $res->NombreContacto;

$datos['nombreResponsable'] = $res->Responsable;

$this->correos\_cotizaciones->rechazoCotizacion($datos);

}

// Si fue autorizada, notificar también al responsable

if ($cotizacion->estatus == 'AUTORIZADA') {

$res = $this->Conexion->consultar(

"SELECT

E.nombre AS NombreCliente,

EC.nombre AS NombreContacto,

CONCAT(R.nombre, ' ', R.paterno) AS Responsable,

R.correo,

correo\_cc

FROM cotizaciones C

INNER JOIN usuarios R ON R.id = C.responsable

INNER JOIN empresas E ON E.id = C.empresa

INNER JOIN empresas\_contactos EC ON EC.id = C.contactos->'$[0]'

WHERE C.id = $cotizacion->id",

TRUE

);

$datos['id'] = $cotizacion->id;

$datos['correos'] = $res->correo;

$datos['nombreCliente'] = $res->NombreCliente;

$datos['nombreContacto'] = $res->NombreContacto;

$datos['nombreResponsable'] = $res->Responsable;

$this->correos\_cotizaciones->aprobacionCotizacion($datos);

}

}

// Procesamiento de los conceptos asociados a la cotización

foreach ($conceptos as $key => $elem) {

$elem->cotizacion = $cotizacion->id;

// Inserta nuevo concepto

if ($elem->id == 0) {

$this->Conexion->insertar('cotizaciones\_conceptos', $elem);

}

// Elimina concepto si el ID es negativo (marcado para eliminar)

if ($elem->id < 0) {

$this->Conexion->eliminar('cotizaciones\_conceptos', array('id' => (intval($elem->id) \* -1)));

}

// Actualiza o re-inserta concepto si ya existía

else if ($elem->id != 0) {

$cot = $this->Conexion->consultar(

"SELECT cotizacion FROM cotizaciones\_conceptos WHERE cotizacion = '$elem->cotizacion'",

TRUE

);

if ($cot) {

$id\_cotizacion = $cot;

$this->Conexion->modificar('cotizaciones\_conceptos', $elem, null, array('id' => $elem->id));

} else {

unset($elem->id); // Elimina el ID para forzar una inserción nueva

$this->Conexion->insertar('cotizaciones\_conceptos', $elem);

}

}

}

// Si no hay estatus (es decir, la cotización aún no ha sido enviada formalmente)

if (!isset($cotizacion->estatus)) {

if ($id\_cotizacion) {

// Comentario en la nueva cotización indicando de cuál se copió

$comentarios = array(

'cotizacion' => $id\_cotizacion\_nueva,

'usuario' => $this->session->id,

'comentario' => 'Esta cotización es copia de: #' . $id\_cotizacion,

);

$this->Conexion->insertar('cotizaciones\_comentarios', $comentarios, array('fecha' => 'CURRENT\_TIMESTAMP()'));

// Comentario en la cotización original indicando que fue copiada

$comentarios2 = array(

'cotizacion' => $id\_cotizacion,

'usuario' => $this->session->id,

'comentario' => 'Esta cotización fue copiada para la cotización: #' . $id\_cotizacion\_nueva,

);

$this->Conexion->insertar('cotizaciones\_comentarios', $comentarios2, array('fecha' => 'CURRENT\_TIMESTAMP()'));

}

}

// Si se envió un comentario general en POST, agregarlo como comentario nuevo

if (isset($\_POST['comentarios']) && $\_POST['comentarios']) {

$comentario = new stdClass;

$comentario->cotizacion = $cotizacion->id;

$comentario->usuario = $this->session->id;

$comentario->comentario = $this->input->post('comentarios');

$this->Conexion->insertar('cotizaciones\_comentarios', $comentario, array('fecha' => 'CURRENT\_TIMESTAMP()'));

}

// Devuelve el ID de la cotización como respuesta (por ejemplo, para JS)

echo $cotizacion->id;

}

function ajax\_getCotizaciones() {

// Limpia la caché de la vista de catálogo

$this->output->delete\_cache('cotizaciones/catalogo');

// Obtiene los filtros enviados por POST

$id = $this->input->post('id');

$texto = $this->input->post('texto');

$parametro = $this->input->post('parametro');

$cliente = $this->input->post('cliente');

$estatus = $this->input->post('estatus');

$tipo = $this->input->post('tipo');

$cerradas = $this->input->post('cerradas');

$fecha1 = $this->input->post('fecha1');

$fecha2 = $this->input->post('fecha2');

// Formateo de fechas para rango de búsqueda

$f1 = strval($fecha1) . ' 00:00:00';

$f2 = strval($fecha2) . ' 23:59:59';

// Filtro de fecha por defecto (último año)

$Limitante\_Fecha = " AND (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR))";

// Bandera para agrupar resultados

$group = TRUE;

// Query base

$query = "SELECT C.\*, IFNULL(MAX(CC.revision), 0) AS UltRev, E.nombre AS Cliente, EC.nombre AS Contacto, CONCAT(R.nombre, ' ', R.paterno) AS Responsable, E.prospecto FROM cotizaciones C LEFT JOIN empresas E ON E.id = C.empresa LEFT JOIN empresas\_contactos EC ON EC.id = C.contactos->'$[0]' LEFT JOIN usuarios R Oid = C.responsable LEFT JOIN cotizaciones\_conceptos CC ON C.id = CC.cotizacion WHERE 1 = 1

";

// Si se busca por ID específico

if ($id) {

$query .= " AND C.id = '$id'";

} else {

// Filtro por cliente seleccionado

if (!empty($cliente) && $cliente != 0) {

$query .= " AND C.empresa = '$cliente'";

}

// Filtro por tipo de cotización

if (!empty($tipo) && $tipo != "TODOS") {

$query .= " AND C.tipo = '$tipo'";

}

// Filtro por estatus

if (!empty($estatus) && $estatus != 'TODOS') {

$query .= " AND C.estatus = '$estatus'";

} else {

// Si no se muestran cerradas

if ($cerradas == "0") {

$query .= " AND (C.estatus != 'CERRADO TOTAL' AND C.estatus != 'CERRADO PARCIAL' AND C.estatus != 'CANCELADA')";

}

}

// Filtro por texto (folio, ID, marca, etc.)

if (!empty($texto)) {

if ($parametro == "folio") {

$query .= " AND C.id = '$texto'";

}

if ($parametro == "id") {

$query .= " AND UPPER(CC.atributos->'$.ID') LIKE '%" . strtoupper($texto) . "%'";

}

if ($parametro == "marca") {

$query .= " AND UPPER(CC.atributos->'$.Marca') LIKE '%" . strtoupper($texto) . "%'";

}

if ($parametro == "serie") {

$query .= " AND UPPER(CC.atributos->'$.Serie') LIKE '%" . strtoupper($texto) . "%'";

}

if ($parametro == "modelo") {

$query .= " AND UPPER(CC.atributos->'$.Modelo') LIKE '%" . strtoupper($texto) . "%'";

}

if ($parametro == "contenido") {

$query .= " AND CC.descripcion LIKE '%$texto%'";

}

// Si el filtro es por responsable, requiere agrupamiento especial

if ($parametro == "responsable") {

$query .= $Limitante\_Fecha;

$group = FALSE;

$query .= " GROUP BY C.id";

$query .= " HAVING Responsable LIKE '%$texto%' ";

}

}

}

// Filtro adicional por rango de fechas (si ambos campos están presentes)

if (!empty($fecha1) && !empty($fecha2)) {

$query .= " AND C.fecha BETWEEN '$f1' AND '$f2'";

}

// Aplica agrupamiento si está habilitado

if ($group) {

$query .= $Limitante\_Fecha;

$query .= " GROUP BY C.id";

}

// Orden descendente por ID

$query .= " ORDER BY C.id DESC";

// Ejecuta la consulta

$res = $this->Conexion->consultar($query, $id);

// Devuelve los resultados en formato JSON

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getClientesCotizaciones() {

$texto = $this->input->post('texto');

// Consulta que devuelve clientes con número de cotizaciones

$query = "SELECT E.id, E.nombre, COUNT(C.id) AS NumCot FROM cotizaciones C INNER JOIN empresas E ON E.id = C.empresa

";

// Filtro por nombre de cliente si se proporcionó texto

if ($texto) {

$query .= " WHERE E.nombre LIKE '%$texto%'";

}

$query .= " GROUP BY E.id";

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getCotizacionConceptos() {

$coti = $this->input->post('cotizacion');

$rev = $this->input->post('revision');

// Consulta conceptos de una cotización (opcionalmente por revisión)

$query = "SELECT CC.\* FROM cotizaciones\_conceptos CC WHERE cotizacion = $coti";

// Si se envió la revisión como parámetro, agregarla al filtro

if (isset($\_POST["revision"])) {

$query .= " AND revision = $rev";

}

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_setRevision() {

$cotizacion = json\_decode($this->input->post("cotizacion"));

$conceptos = json\_decode($this->input->post("conceptos"));

$id = $cotizacion->id;

// Elimina el ID de la cotización (solo se usará como referencia, no se modificará)

unset($cotizacion->id);

// Obtiene el siguiente número de revisión

$rv = $this->Conexion->consultar(

"SELECT (MAX(CC.revision) + 1) AS Rev

FROM cotizaciones\_conceptos CC

WHERE CC.cotizacion = $id",

TRUE

);

// Inserta todos los conceptos con la nueva revisión

foreach ($conceptos as $key => $elem) {

$elem->cotizacion = $id;

$elem->revision = $rv->Rev;

$this->Conexion->insertar('cotizaciones\_conceptos', $elem);

}

// Actualiza la cabecera de la cotización (excepto el ID)

$this->Conexion->modificar('cotizaciones', $cotizacion, null, array('id' => $id));

echo "1";

}

function ajax\_setComentarios() {

$comentario = json\_decode($this->input->post('comentario'));

// Asigna al comentario el usuario actual y la fecha actual

$comentario->usuario = $this->session->id;

$funciones = array('fecha' => 'CURRENT\_TIMESTAMP()');

// Inserta el comentario

$res = $this->Conexion->insertar('cotizaciones\_comentarios', $comentario, $funciones);

if ($res > 0) {

echo "1";

}

}

function ajax\_getComentarios() {

$id = $this->input->post('id');

// Consulta los comentarios de una cotización y el nombre del usuario que los hizo

$query = "SELECT C.\*, CONCAT(U.nombre, ' ', U.paterno) AS User FROM cotizaciones\_comentarios C INNER JOIN usuarios U ON U.id = C.usuario WHERE 1 = 1

";

// Filtra por ID de cotización si se proporciona

if ($id) {

$query .= " AND C.cotizacion = '$id'";

}

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getSAData() {

// Permite acceso desde cualquier origen (CORS)

header("Access-Control-Allow-Origin: \*");

// Obtiene el ID del equipo desde POST

$id\_equipo = $this->input->post('id\_equipo');

// Ejecuta un archivo externo .exe pasando el ID como parámetro

$res = shell\_exec("C:/xampp/htdocs/sa\_reader/sa\_reader.exe \"$id\_equipo\"");

echo $res;

}

function ajax\_getClientes() {

$id = $this->input->post('id');

$nombre = $this->input->post('nombre');

// Consulta empresas que sean clientes y tengan moneda de cotización

$query = "SELECT E.\* FROM empresas E WHERE E.cliente = 1 AND JSON\_LENGTH(E.moneda\_cotizacion) > 0

";

// Filtro por ID si se proporciona

if ($id) {

$query .= " AND E.id = '$id'";

} else {

// Filtro por nombre parcial si se proporciona

if ($nombre) {

$query .= " AND E.nombre LIKE '%$nombre%'";

}

}

$res = $this->Conexion->consultar($query, $id);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getPlanta() {

$id = $this->input->post('id');

// Consulta la planta de la empresa por ID

$query = "SELECT \* FROM empresa\_plantas WHERE id = $id";

$res = $this->Conexion->consultar($query, TRUE);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getbuscarAutores() {

$id = $this->input->post('id');

// Consulta usuarios activos con privilegio para administrar cotizaciones

$query = "SELECT U.id, CONCAT(U.nombre, ' ', U.paterno) AS Nombre, P.puesto AS Puesto, U.correo FROM usuarios U INNER JOIN puestos P ON U.puesto = P.id INNER JOIN privilegios PR ON PR.usuario = U.id WHERE U.activo = 1 AND PR.administrar\_cotizaciones = 1";

// Filtro por ID si se proporciona

if ($id) {

$query .= " AND U.id = '$id'";

}

$res = $this->Conexion->consultar($query, $id);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getContactos() {

$id = $this->input->post('id');

$empresa = $this->input->post('id\_cliente');

// Consulta contactos activos y cotizables

$query = "SELECT \* FROM empresas\_contactos WHERE activo = 1 AND cotizable = 1";

// Filtro por ID si se proporciona

if ($id) {

$query .= " AND id = '$id'";

} else {

// Filtro por empresa si se proporciona

if ($empresa) {

$query .= " AND empresa = '$empresa'";

}

}

$res = $this->Conexion->consultar($query, $id);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getNombresUsuarios() {

$ids = $this->input->post('ids');

// Convierte el array en sintaxis SQL válida para IN

$ids = str\_replace('[', '(', $ids);

$ids = str\_replace(']', ')', $ids);

// Consulta nombres de los usuarios dados sus IDs

$query = "SELECT id, CONCAT(nombre, ' ', paterno) AS User FROM usuarios WHERE id IN $ids";

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getServicio() {

$codigo = $this->input->post('codigo');

// Consulta servicio por código y su precio

$query = "SELECT S.id, S.codigo, S.sitio, S.descripcion AS DescripcionServicio, CP.alto\_a AS Precio FROM servicios S INNER JOIN claves\_precio CP ON S.clave\_precio = CP.id WHERE S.codigo = '$codigo'";

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getServicioMarMod() {

$fabricante = $this->input->post('fabricante');

$modelo = $this->input->post('modelo');

// Consulta requerimiento catalogado por marca y modelo

$query = "SELECT R.id, R.descripcion, R.servicio FROM requerimientos R WHERE R.catalogado = '1' AND UPPER(TRIM(R.fabricante)) = '$fabricante' AND UPPER(TRIM(R.modelo)) = '$modelo' LIMIT 1";

$res = $this->Conexion->consultar($query, TRUE);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_enviarCorreo() {

// Carga la librería encargada del envío de correos con archivos adjuntos

$this->load->library('correos\_archivos');

// Decodifica la cotización recibida por POST

$cotizacion = json\_decode($this->input->post("cotizacion"));

$id\_cot = $cotizacion->id;

$param = $cotizacion->id . "-" . $cotizacion->UltRev;

// Genera el PDF de la cotización

$pdf\_content = $this->cotizacion\_pdf($param, 'S');

// Consulta los correos del contacto, responsable, creador y copia

$res = $this->Conexion->consultar("SELECT ec.correo AS contacto, r.correo AS responsable, u.correo AS creador, c.correo\_cc FROM cotizaciones c JOIN empresas\_contactos ec ON JSON\_CONTAINS(c.contactos, CAST(ec.id AS JSON), '$') JOIN usuarios r ON c.responsable = r.id JOIN usuarios u ON c.usuario = u.id WHERE c.id = $id\_cot",TRUE );

// Consulta el cuerpo del correo activo

$body = $this->Conexion->consultar("SELECT \* FROM texto\_correo WHERE activo = 1", TRUE);

// Obtiene la firma del usuario en sesión

$firma = $this->Conexion->consultar("SELECT foto\_firma FROM usuarios WHERE id = " . $this->session->id, TRUE);

// Guarda la firma en un archivo temporal

$nombre\_archivo = 'firma\_' . $this->session->id . '.png';

$ruta\_temporal = sys\_get\_temp\_dir() . '/' . $nombre\_archivo;

file\_put\_contents($ruta\_temporal, $firma->foto\_firma);

// Construye los datos del correo

$datos['asunto'] = "Cotización Masmetrologia – #" . $id\_cot;

$datos['pdf'] = $pdf\_content;

$datos['pdf\_nombre'] = "Cotización - #" . $param . ".pdf";

$datos['body'] = $body->texto;

$datos['firma'] = $ruta\_temporal;

$datos['firma\_nombre'] = $nombre\_archivo;

$datos['contacto'] = $res->contacto;

$datos['responsable'] = $res->responsable;

$datos['creador'] = $res->creador;

$datos['correo\_cc'] = $res->correo\_cc;

// Envía el correo y actualiza estatus si es AUTORIZADA

if ($this->correos\_archivos->Solicitar\_Autorización($datos)) {

if ($datos['estatus'] == "AUTORIZADA") {

$this->Conexion->modificar(

'cotizaciones',

array(

'estatus' => 'ENVIADA',

'bitacora\_estatus' => $cotizacion->bitacora\_estatus

),

null,

array('id' => $datos['id'])

);

}

echo "1";

}

}

// Carga la vista del calendario de cotizaciones con los datos del subusuario actual

function calendario() {

$this->load->model('usuarios\_model');

$datos['sub'] = $this->usuarios\_model->userCots();

$this->load->view('header');

$this->load->view('cotizaciones/calendario', $datos);

}

// Obtiene las acciones programadas para mostrarlas en el calendario

function ajax\_getAcciones\_calendar() {

$idsub = $this->input->get('idSub');

$query = "SELECT A.\*, concat('Responsable: ',U.nombre, ' ', U.paterno,'\r\n','Cotizacion: ', A.idCot ) as title, A.fecha\_limite as start, A.fecha\_limite + interval 1 hour as end, concat(U.nombre, ' ', U.paterno) as User,";

$query .= " if(estatus = 'CANCELADA', 'gray', if(estatus = 'PENDIENTE' and current\_timestamp() > A.fecha\_limite, 'red', if(estatus = 'PENDIENTE' and current\_timestamp() <= A.fecha\_limite, '#f0ad4e', if(estatus = 'REALIZADA' and A.fecha\_realizada > A.fecha\_limite, '#76A874', if(estatus = 'REALIZADA' and A.fecha\_realizada <= A.fecha\_limite, 'green', ''))))) as color";

$query .= " from cot\_acciones A inner join usuarios U on U.id = A.usuario";

if (!empty($idsub)) {

$query .= " where U.id=" . $idsub;

}

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

// Registra una nueva acción de seguimiento para una cotización y envía un evento de calendario por correo

function ajax\_setAccion() {

$this->load->library('correos\_archivos');

$accion = json\_decode($this->input->post('data'));

$rev = $this->input->post('rev');

$responsable = $this->input->post('responsable');

$enviar\_contacto = $this->input->post('enviar\_contacto');

$idUser = null;

$contacto = null;

$param = $accion->idCot . "-" . $rev;

$pdf\_content = $this->cotizacion\_pdf($param, 'S');

if (empty($responsable)) {

$accion->usuario = $this->session->id;

$idUser = $this->session->id;

} else {

$accion->usuario = $responsable;

$idUser = $responsable;

}

$accion->estatus = "PENDIENTE";

$func['fecha\_creacion'] = 'CURRENT\_TIMESTAMP()';

$id = $this->Conexion->insertar('cot\_acciones', $accion, $func);

$query = "SELECT concat(nombre , '', paterno) as name, correo from usuarios where id = " . $idUser;

$res = $this->Conexion->consultar($query, TRUE);

if ($enviar\_contacto == 1) {

$contacto = $this->Conexion->consultar(

"SELECT ec.correo

FROM cotizaciones c

JOIN empresas\_contactos ec

ON JSON\_CONTAINS(c.contactos, CAST(ec.id AS JSON), '$')

WHERE c.id = $accion->idCot",

TRUE

);

}

// Generar contenido para evento de calendario

$fecha = strtotime($accion->fecha\_limite);

$name = "Cotización : " . $accion->idCot;

$start = date('Ymd', $fecha) . 'T' . date('His', $fecha);

$end = date('Ymd', $fecha) . 'T' . date('His', $fecha);

$description = $accion->accion;

$ical\_content = "BEGIN:VCALENDAR

VERSION:2.0

PRODID:-//LearnPHP.co//NONSGML {$res->name}//EN

METHOD:REQUEST

BEGIN:VEVENT

ORGANIZER;CN=" . $this->session->nombre . ":tickets@masmetrologia.com

UID:" . date('Ymd') . 'T' . date('His') . rand() . "-learnphp.co

DTSTAMP:" . date('Ymd') . 'T' . date('His') . "

DTSTART:{$start}

DTEND:{$end}

SUMMARY:{$name}

DESCRIPTION:{$description}

END:VEVENT

END:VCALENDAR";

// Datos del correo

$datos['nombre'] = "Seguimiento-" . $accion->idCot . ".ics";

$datos['cuerpo'] = "Esto es un evento de seguimiento para la cotización - " . $accion->idCot;

$datos['cal'] = $ical\_content;

$datos['correo'] = $res->correo;

$datos['accion'] = $description;

$datos['contacto'] = $contacto->correo;

$datos['pdf'] = $pdf\_content;

$datos['pdf\_nombre'] = "Cotización-" . $param . ".pdf";

$datos['correoResponsable'] = $this->session->correo;

$this->correos\_archivos->evento\_cotizaciones($datos);

if ($id > 0) {

echo $id;

}

}

// Obtiene las acciones asociadas a una cotización específica

function ajax\_getAcciones() {

$po = $this->input->post('idCot');

$res = $this->Conexion->consultar(

"SELECT A.\*, concat(U.nombre, ' ', U.paterno) as User

FROM cot\_acciones A

INNER JOIN usuarios U ON U.id = A.usuario

WHERE A.idCot != 0 AND A.idCot = $po"

);

if ($res) {

echo json\_encode($res);

}

}

// Marca una acción como realizada, registrando la fecha actual

function ajax\_setAccionRealizada() {

$accion = json\_decode($this->input->post('data'));

$func['fecha\_realizada'] = 'CURRENT\_TIMESTAMP()';

$id = $this->Conexion->modificar(

'cot\_acciones',

$accion,

$func,

array('id' => $accion->id)

);

}

// Actualiza los datos de una acción existente

function ajax\_updateAccion() {

$accion = json\_decode($this->input->post('data'));

$id = $this->Conexion->modificar('cot\_acciones',$accion,null,array('id' => $accion->id)

);

}

// Inserta un comentario asociado a una acción

function ajax\_setAccionComentario() {

$comment = json\_decode($this->input->post('data'));

$comment->usuario = $this->session->id;

$func['fecha'] = 'CURRENT\_TIMESTAMP()';

$id = $this->Conexion->insertar('cot\_acciones\_comentarios', $comment, $func);

if ($id > 0) {

echo $id;

}

}

// Obtiene los comentarios asociados a una acción específica

function ajax\_getAccionComentarios() {

$accion = $this->input->post('accion');

$query = "SELECT C.\*, concat(U.nombre, ' ', U.paterno, ' ', U.materno) as User FROM cot\_acciones\_comentarios C INNER JOIN usuarios U ON U.id = C.usuario WHERE C.accion = " . $accion;

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function exportar() {

$parametro = $this->input->post('rbBusqueda');

$texto = $this->input->post('txtBusqueda');

$cliente = $this->input->post('txtClienteId');

$tipoCot = $this->input->post('opTipoCotizacion');

$estatus = $this->input->post('opEstatus');

$cerradas = $this->input->post('cbCerradasCanceladas');

$fecha1 = $this->input->post('fecha1');

$fecha2 = $this->input->post('fecha2');

$f1 = strval($fecha1) . ' 00:00:00';

$f2 = strval($fecha2) . ' 23:59:59';

$query2 = "";

if (empty($cerradas)) {

$close = " and (C.estatus != 'CERRADO TOTAL' and C.estatus != 'CERRADO PARCIAL' and C.estatus != 'CANCELADA')";

} else {

$close = "";

}

$sep = '",';

$coma = "'";

$url = base\_url('cotizaciones/ver\_cotizacion/');

$con = $coma . '=HYPERLINK("' . $url . '' . $coma;

$query = "SELECT C.id as Cotizacion, C.fecha as Fecha, C.tipo as Servicio, C.estatus as Estatus, EC.nombre as Contacto, EC.telefono as Telefono, EC.celular as Celular, EC.correo as Correo, E.nombre as Cliente, concat(R.nombre, ' ', R.paterno) as Responsable, (SELECT fecha\_creacion from cot\_acciones WHERE idCot= C.id ORDER BY id DESC limit 1) as Fecha\_Seguimiento, (SELECT accion from cot\_acciones WHERE idCot= C.id ORDER BY id DESC limit 1) as Accion\_Seguimiento, C.prospectoEmp from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos->'$[0]' left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 " . $close . "and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ";

if (!empty($cliente) && $cliente != 0) {

$query .= " and C.empresa = '$cliente'";

}

if (!empty($tipoCot) && $tipoCot != "TODOS") {

$query .= " and C.tipo = '$tipoCot'";

}

if (!empty($estatus) && $estatus != 'TODOS') {

$query .= " and C.estatus = '$estatus'";

}

if (!empty($texto)) {

if ($parametro == "folio") {

$query .= " and C.id = '$texto'";

}

if ($parametro == "id") {

$query .= " and UPPER(CC.atributos->'$.ID') like '%" . strtoupper($texto) . "%'";

}

if ($parametro == "marca") {

$query .= " and UPPER(CC.atributos->'$.Marca') like '%" . strtoupper($texto) . "%'";

}

if ($parametro == "serie") {

$query .= " and UPPER(CC.atributos->'$.Serie') like '%" . strtoupper($texto) . "%'";

}

if ($parametro == "modelo") {

$query .= " and UPPER(CC.atributos->'$.Modelo') like '%" . strtoupper($texto) . "%'";

}

if ($parametro == "contenido") {

$query .= " and CC.descripcion like '%$texto%'";

}

if ($parametro == "responsable") {

$query2 = " having Responsable like '%$texto%' ";

}

}

if (!empty($fecha1) && !empty($fecha2)) {

$query .= " and C.fecha BETWEEN '" . $f1 . "' AND '" . $f2 . "'";

}

$query .= ' group by C.id ' . $query2 . ' order by C.fecha desc';

$result = $this->Conexion->consultar($query);

$salida = '';

$salida .= '<table style="border: 1px solid black; border-collapse: collapse;">

<thead>

<tr>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Cotizacion</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Fecha</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Servicio</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Estatus</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Contacto</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Telefono</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Celular</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Correo</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Cliente</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Prospecto</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Responsable</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Fecha Seguimiento</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Accion Seguimiento</th>

</tr>

</thead>

<tbody>';

foreach ($result as $row) {

$prospecto = ($row->prospectoEmp == 1) ? 'Si' : 'No';

$salida .= '

<tr>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Cotizacion . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Fecha . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Servicio . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Estatus . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Contacto . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Telefono . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Celular . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Correo . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Cliente . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $prospecto . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Responsable . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Fecha\_Seguimiento . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->Accion\_Seguimiento . '</td>

</tr>';

}

$salida .= '</tbody></table>';

$timestamp = date('m/d/Y', time());

$filename = 'Cotizaciones' . $timestamp . '.xls';

header("Content-Type: application/vnd.ms-excel");

header("Content-Disposition: attachment; filename=\"$filename\"");

header("Pragma: no-cache");

header("Expires: 0");

header('Content-Transfer-Encoding: binary');

echo $salida;

}

function ajax\_getDashboard(){

$sub = $this->input->post('idSub');

$query = "SELECT C.\*, ifnull(max(CC.revision), 0) as UltRev, E.nombre as Cliente, EC.nombre as Contacto, concat(R.nombre, ' ', R.paterno) as Responsable from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 and (C.estatus != 'CERRADO TOTAL' and C.estatus != 'CERRADO PARCIAL' and C.estatus != 'CANCELADA') and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ";

if ($this->session->privilegios['cotDashboard']) {

$query .= " ";

} elseif ($this->session->privilegios['generar\_cotizaciones']) {

$query .= " and C.usuario = ".$this->session->id;

}

if ($sub != 'TODOS') {

$query .= " and R.nombre like '%".$sub."%'";

}

$query .= " group by C.id order by C.fecha desc";

$res = $this->Conexion->consultar($query);

$res = count($res);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getDashboardCreadas(){

$sub = $this->input->post('idSub');

// Consulta para obtener cotizaciones con estatus "CREADA" del último año

$query = 'SELECT C.\*, ifnull(max(CC.revision), 0) as UltRev, E.nombre as Cliente, EC.nombre as Contacto, concat(R.nombre, " ", R.paterno) as Responsable from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos->"$[0]" left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 and C.estatus = "CREADA" and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ';

// Filtro según privilegios de sesión

if ($this->session->privilegios['cotDashboard']) {

$query .= " ";

} elseif ($this->session->privilegios['generar\_cotizaciones']) {

$query .= " and C.usuario = ".$this->session->id;

}

// Filtro por responsable si se indica un subusuario específico

if ($sub != 'TODOS') {

$query .= " and R.nombre like '%".$sub."%'";

}

$query .= ' group by C.id order by C.fecha desc';

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getDashboardRechazadas(){

$sub = $this->input->post('idSub');

// Consulta para obtener cotizaciones RECHAZADAS del último año

$query = 'SELECT C.\*, ifnull(max(CC.revision), 0) as UltRev, E.nombre as Cliente, EC.nombre as Contacto, concat(R.nombre, " ", R.paterno) as Responsable from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos->"$[0]" left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 and C.estatus = "RECHAZADA" and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ';

// Filtro según privilegios del usuario en sesión

if ($this->session->privilegios['cotDashboard']) {

$query .= " ";

} elseif ($this->session->privilegios['generar\_cotizaciones']) {

$query .= " and C.usuario = " . $this->session->id;

}

// Filtro por subusuario si aplica

if ($sub != 'TODOS') {

$query .= " and R.nombre like '%" . $sub . "%'";

}

// Agrupación y orden de resultados

$query .= ' group by C.id order by C.fecha desc';

// Depuración: se muestra la consulta y se detiene la ejecución

echo $query; die();

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getDashboardPendientes(){

$sub = $this->input->post('idSub');

// Consulta para obtener cotizaciones en estado "PENDIENTE AUTORIZACION" del último año

$query = 'SELECT C.\*, ifnull(max(CC.revision), 0) as UltRev, E.nombre as Cliente, EC.nombre as Contacto, concat(R.nombre, " ", R.paterno) as Responsable from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos->"$[0]" left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 and C.estatus = "PENDIENTE AUTORIZACION" and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ';

// Filtro según los privilegios del usuario en sesión

if ($this->session->privilegios['cotDashboard']) {

$query .= " ";

} elseif ($this->session->privilegios['generar\_cotizaciones']) {

$query .= " and C.usuario = " . $this->session->id;

}

// Filtro adicional si se seleccionó un subusuario específico

if ($sub != 'TODOS') {

$query .= " and R.nombre like '%" . $sub . "%'";

}

// Agrupación por cotización y orden descendente por fecha

$query .= ' group by C.id order by C.fecha desc';

// Ejecución de la consulta

$res = $this->Conexion->consultar($query);

// Si hay resultados, se regresan en formato JSON

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getDashboardRevision(){

$sub = $this->input->post('idSub');

// Consulta para obtener cotizaciones en estado "EN REVISION" del último año

$query = 'SELECT C.\*, ifnull(max(CC.revision), 0) as UltRev, E.nombre as Cliente, EC.nombre as Contacto, concat(R.nombre, " ", R.paterno) as Responsable from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos->"$[0]" left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 and C.estatus = "EN REVISION" and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ';

// Filtro según privilegios del usuario en sesión

if ($this->session->privilegios['cotDashboard']) {

$query .= " ";

} elseif ($this->session->privilegios['generar\_cotizaciones']) {

$query .= " and C.usuario = " . $this->session->id;

}

// Filtro por subusuario si se especifica uno

if ($sub != 'TODOS') {

$query .= " and R.nombre like '%" . $sub . "%'";

}

// Agrupamiento y ordenamiento

$query .= ' group by C.id order by C.fecha desc';

// Ejecución de la consulta

$res = $this->Conexion->consultar($query);

// Devolver resultados en formato JSON

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getDashboardAutorizadas(){

$sub = $this->input->post('idSub');

// Consulta para obtener cotizaciones con estatus "AUTORIZADA" del último año

$query = 'SELECT C.\*, ifnull(max(CC.revision), 0) as UltRev, E.nombre as Cliente, EC.nombre as Contacto, concat(R.nombre, " ", R.paterno) as Responsable from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos->"$[0]" left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 and C.estatus = "AUTORIZADA" and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ';

// Filtro según privilegios del usuario en sesión

if ($this->session->privilegios['cotDashboard']) {

$query .= " ";

} elseif ($this->session->privilegios['generar\_cotizaciones']) {

$query .= " and C.usuario = " . $this->session->id;

}

// Filtro por subusuario si se especifica uno

if ($sub != 'TODOS') {

$query .= " and R.nombre like '%" . $sub . "%'";

}

// Agrupamiento y ordenamiento

$query .= ' group by C.id order by C.fecha desc';

// Ejecución de la consulta

$res = $this->Conexion->consultar($query);

// Devolver resultados en formato JSON si hay datos

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getDashboardConfirmadas(){

$sub = $this->input->post('idSub');

// Consulta principal para cotizaciones con estatus "CONFIRMADA"

$query = 'SELECT C.\*, ifnull(max(CC.revision), 0) as UltRev, E.nombre as Cliente, EC.nombre as Contacto, concat(R.nombre, " ", R.paterno) as Responsable from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos->"$[0]" left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 and C.estatus = "CONFIRMADA" and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ';

// Filtro por privilegios del usuario

if ($this->session->privilegios['cotDashboard']) {

$query .= " ";

} elseif ($this->session->privilegios['generar\_cotizaciones']) {

$query .= " and C.usuario = " . $this->session->id;

}

// Filtro por subusuario si se indicó alguno

if ($sub != 'TODOS') {

$query .= " and R.nombre like '%" . $sub . "%'";

}

// Agrupamiento y orden por fecha descendente

$query .= ' group by C.id order by C.fecha desc';

// Ejecutar consulta y retornar resultado si existe

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getDashboardAprobacion(){

$sub = $this->input->post('idSub');

// Consulta para obtener cotizaciones con estatus "EN APROBACION" del último año

$query = 'SELECT C.\*, ifnull(max(CC.revision), 0) as UltRev, E.nombre as Cliente, EC.nombre as Contacto, concat(R.nombre, " ", R.paterno) as Responsable from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos->"$[0]" left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 and C.estatus = "EN APROBACION" and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ';

// Filtro según privilegios del usuario

if ($this->session->privilegios['cotDashboard']) {

$query .= " ";

} elseif ($this->session->privilegios['generar\_cotizaciones']) {

$query .= " and C.usuario = " . $this->session->id;

}

// Filtro por subusuario si corresponde

if ($sub != 'TODOS') {

$query .= " and R.nombre like '%" . $sub . "%'";

}

// Agrupamiento y orden final

$query .= ' group by C.id order by C.fecha desc';

// Ejecutar consulta y retornar resultado

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getDashboardApParcial(){

$sub = $this->input->post('idSub');

// Consulta para obtener cotizaciones con estatus "APROBADO PARCIAL" del último año

$query = 'SELECT C.\*, ifnull(max(CC.revision), 0) as UltRev, E.nombre as Cliente, EC.nombre as Contacto, concat(R.nombre, " ", R.paterno) as Responsable from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos->"$[0]" left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 and C.estatus = "APROBADO PARCIAL" and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ';

// Filtro según privilegios del usuario

if ($this->session->privilegios['cotDashboard']) {

$query .= " ";

} elseif ($this->session->privilegios['generar\_cotizaciones']) {

$query .= " and C.usuario = " . $this->session->id;

}

// Filtro por subusuario si corresponde

if ($sub != 'TODOS') {

$query .= " and R.nombre like '%" . $sub . "%'";

}

// Agrupamiento y orden final

$query .= ' group by C.id order by C.fecha desc';

// Ejecutar consulta y retornar resultado

$res = $this->Conexion->consultar($query);

if ($res) {

echo json\_encode($res);

}

}

function ajax\_getDashboardApTotal(){

$sub = $this->input->post('idSub');

// Consulta base: obtiene cotizaciones con estatus "APROBADO TOTAL" del último año

$query = 'SELECT C.\*, ifnull(max(CC.revision), 0) as UltRev, E.nombre as Cliente, EC.nombre as Contacto, concat(R.nombre, " ", R.paterno) as Responsable from cotizaciones C left join empresas E on E.id = C.empresa left join empresas\_contactos EC on EC.id = C.contactos->"$[0]" left join usuarios R on R.id = C.responsable inner join cotizaciones\_conceptos CC on C.id = CC.cotizacion where 1 = 1 and C.estatus = "APROBADO TOTAL" and (C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) ';

// Filtro por privilegios del usuario

if ($this->session->privilegios['cotDashboard']) {

$query .= " ";

} elseif ($this->session->privilegios['generar\_cotizaciones']) {

$query .= " and C.usuario = " . $this->session->id;

}

// Filtro por subusuario si se especifica

if ($sub != 'TODOS') {

$query .= " and R.nombre like '%" . $sub . "%'";

}

// Agrupar por ID de cotización y ordenar por fecha descendente

$query .= ' group by C.id order by C.fecha desc';

// Ejecutar la consulta

$res = $this->Conexion->consultar($query);

// Si hay resultados, devolverlos como JSON

if ($res) {

echo json\_encode($res);

}

}

function excel()

{

// Consulta todos los conceptos de la cotización con ID 16747

$query = "SELECT CC.\* from cotizaciones\_conceptos CC where cotizacion = 16747";

$result = $this->Conexion->consultar($query);

$salida = '';

// Encabezado de la tabla con estilos en línea

$salida .= '<table style="border: 1px solid black; border-collapse: collapse;">

<thead>

<tr>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Cantidad</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Concepto</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Servicio</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Comentarios</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Servicio a Realizarse</th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">T. Entrega </th>

<th style="background-color: #F3F1F1; color: black; border: 1px solid black; border-collapse: collapse">Precio Unitario</th>

</tr>

</thead>

<tbody>';

$d = 2;

// Cuerpo de la tabla con los datos obtenidos

foreach ($result as $row) {

$salida .= '

<tr>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->cantidad . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->descripcion . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->servicios . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->comentarios . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->sitio . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->tiempo\_entrega . '</td>

<td style="color: $444; border: 1px solid black; border-collapse: collapse">' . $row->precio\_unitario . '</td>

</tr>';

$d = $d + 1;

}

$salida .= '</tbody>

</table>';

// Encabezados para forzar la descarga como archivo Excel

$timestamp = date('m/d/Y', time());

$filename = 'QR\_' . $timestamp . '.xls';

header("Content-Type: application/vnd.ms-excel");

header("Content-Disposition: attachment; filename=\"$filename\"");

header("Pragma: no-cache");

header("Expires: 0");

header('Content-Transfer-Encoding: binary');

// Salida final del archivo

echo $salida;

}

function buscarQrs()

{

// Obtiene el valor ingresado para búsqueda del QR

$qr = $this->input->post('txtBuscarQr');

// Consulta el ID y estatus de la requisición que coincide con el QR

$query = "SELECT id, estatus FROM requisiciones\_cotizacion where id = " . $qr;

$res = $this->Conexion->consultar($query);

// Devuelve los resultados en formato JSON si existen

if ($res)

{

echo json\_encode($res);

}

}

function ajax\_setQr()

{

// Recibe los datos enviados por POST

$qr = $this->input->post('qr');

$id\_cotizacion = $this->input->post('id\_cotizacion');

// Crea el arreglo de datos a actualizar

$data = array('id\_cotizacion' => $id\_cotizacion);

// Actualiza la tabla requisiciones\_cotizacion con el ID de cotización

$res = $this->Conexion->modificar('requisiciones\_cotizacion', $data, null, array('id' => $qr));

// Devuelve 1 si la operación fue exitosa

if ($res)

{

echo 1;

}

}

function ajax\_getQrs()

{

// Recibe el ID de cotización desde POST

$id\_cotizacion = $this->input->post('id\_cotizacion');

// Consulta los QRs asociados a la cotización

$query = "SELECT id FROM requisiciones\_cotizacion where id\_cotizacion = " . $id\_cotizacion;

$res = $this->Conexion->consultar($query);

// Devuelve los IDs en formato JSON si existen resultados

if ($res)

{

echo json\_encode($res);

}

}

function ajax\_getReporteCotizaciones()

{

$requisitor = $this->input->post('asignado');

$us=null;

$usF=null;

if ($requisitor != 'TODO') {

$us =' and C1.usuario ='.$requisitor;

$usF =' and C.usuario ='.$requisitor;

}

$query="SELECT

COUNT(DISTINCT C.id) AS Total,

(SELECT COUNT(DISTINCT C1.id)

FROM cotizaciones C1

WHERE C1.estatus = 'CREADA'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS TotalCreadas,

(SELECT MIN(C1.fecha)

FROM cotizaciones C1

WHERE C1.estatus = 'CREADA'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS UltFechaCreadas,

(SELECT COUNT(DISTINCT C1.id)

FROM cotizaciones C1

WHERE C1.estatus = 'RECHAZADA'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS TotalRechazadas,

(SELECT MIN(C1.fecha)

FROM cotizaciones C1

WHERE C1.estatus = 'RECHAZADA'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS UltFechaRechazadas,

(SELECT COUNT(DISTINCT C1.id)

FROM cotizaciones C1

WHERE C1.estatus = 'PENDIENTE AUTORIZACION'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS TotalPendienteAutorizacion,

(SELECT MIN(C1.fecha)

FROM cotizaciones C1

WHERE C1.estatus = 'PENDIENTE AUTORIZACION'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS UltFechaPendienteAutorizacion,

(SELECT COUNT(DISTINCT C1.id)

FROM cotizaciones C1

WHERE C1.estatus = 'EN REVISION'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS TotalEnRevision,

(SELECT MIN(C1.fecha)

FROM cotizaciones C1

WHERE C1.estatus = 'EN REVISION'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS UltFechaEnRevision,

(SELECT COUNT(DISTINCT C1.id)

FROM cotizaciones C1

WHERE C1.estatus = 'AUTORIZADA'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS TotalAutorizadas,

(SELECT MIN(C1.fecha)

FROM cotizaciones C1

WHERE C1.estatus = 'AUTORIZADA'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS UltFechaAutorizadas,

(SELECT COUNT(DISTINCT C1.id)

FROM cotizaciones C1

WHERE C1.estatus = 'CONFIRMADA'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS TotalConfirmada,

(SELECT MIN(C1.fecha)

FROM cotizaciones C1

WHERE C1.estatus = 'CONFIRMADA'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS UltFechaConfirmada,

(SELECT COUNT(DISTINCT C1.id)

FROM cotizaciones C1

WHERE C1.estatus = 'EN APROBACION'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS TotalEnAprobacion,

(SELECT MIN(C1.fecha)

FROM cotizaciones C1

WHERE C1.estatus = 'EN APROBACION'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS UltFechaEnAprobacion,

(SELECT COUNT(DISTINCT C1.id)

FROM cotizaciones C1

WHERE C1.estatus = 'APROBADO TOTAL'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS TotalAprobadoTotal,

(SELECT MIN(C1.fecha)

FROM cotizaciones C1

WHERE C1.estatus = 'APROBADO TOTAL'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS UltFechaAprobadoTotal,

(SELECT COUNT(DISTINCT C1.id)

FROM cotizaciones C1

WHERE C1.estatus = 'APROBADO PARCIAL'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS TotalAprobadoParcial,

(SELECT MIN(C1.fecha)

FROM cotizaciones C1

WHERE C1.estatus = 'APROBADO PARCIAL'".$us."

AND C1.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C1.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)) AS UltFechaAprobadoParcial

FROM cotizaciones C

WHERE C.estatus NOT IN ('CERRADO TOTAL', 'CERRADO PARCIAL', 'CANCELADA')

AND C.fecha > (CURRENT\_DATE() - INTERVAL 1 YEAR)".$usF;

$res = $this->Conexion->consultar($query, TRUE);

//echo $query;die

echo json\_encode($res);

}

}

## Descargas.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Descargas extends CI\_Controller {

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('descargas\_model', 'Modelo'); // Modelo principal para descargas

        $this->load->helper('download'); // Helper para forzar descargas

        $this->load->model('MLConexion\_model', 'MLConexion'); // Modelo para conexión con WO\_Master

    }

    public function index() {

        // Sin implementación; puede utilizarse como landing para el controlador si se requiere

    }

    // Descarga archivos adjuntos a tickets de autos

    function tickets\_AT($id) {

        $row = $this->Modelo->getFile($id, 'tickets\_autos\_archivos');

        $file = $row->archivo;

        $nombre = $row->nombre;

        force\_download($nombre, $file);

    }

    // Descarga archivos adjuntos a tickets de edificio

    function tickets\_ED($id) {

        $row = $this->Modelo->getFile($id, 'tickets\_edificio\_archivos');

        $file = $row->archivo;

        $nombre = $row->nombre;

        force\_download($nombre, $file);

    }

    // Descarga archivos adjuntos a tickets de sistemas

    function tickets\_IT($id) {

        $row = $this->Modelo->getFile($id, 'tickets\_sistemas\_archivos');

        $file = $row->archivo;

        $nombre = $row->nombre;

        force\_download($nombre, $file);

    }

    // Descarga archivos adjuntos a órdenes de trabajo (WO\_Master)

    function ordenes\_trabajo($id) {

        $row = $this->MLConexion->consultar("SELECT \* from WO\_Master where WorkOrder\_ID=".$id, true);

        $file = $row->archivo;

        $nombre = $row->nombre\_archivo;

        force\_download($nombre, $file);

    }

}

## Documentacion.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Documentacion extends CI\_Controller {

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('conexion\_model', 'Conexion'); // Carga el modelo de conexión general

    }

    // Muestra la vista del catálogo de manuales

    function manuales() {

        $this->load->view('header');

        $this->load->view('documentacion/catalogo');

    }

    // Muestra la vista del mapa del sitio

    function mapa\_sitio() {

        $this->load->view('header');

        $this->load->view('documentacion/mapa\_sitio');

    }

}

## Empresas.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Empresas extends CI\_Controller {

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('empresas\_model', 'Modelo');

        $this->db->db\_debug = FALSE; // Desactiva los errores detallados de la base de datos

    }

    // Carga la vista principal del catálogo de empresas

    function index() {

        // $this->output->enable\_profiler(TRUE); // Línea comentada para depuración

        // $datos['empresas'] = $this->Modelo->getEmpresas(); // Comentado, no se usa en la vista actual

        $this->load->view('header');

        $this->load->view('empresas/catalogo');

    }

    // Muestra el formulario para dar de alta una nueva empresa

    function alta() {

        $datos['paises'] = $this->Modelo->listadopaises(); // Lista de países para el formulario

        $this->load->view('header');

        $this->load->view('empresas/alta', $datos);

    }

    // Muestra los detalles de una empresa específica

    function ver($id) {

        $datos['empresa'] = $this->Modelo->getEmpresa($id); // Datos generales de la empresa

        $datos['proveedor'] = $this->Modelo->getProveedor($id); // Información si es proveedor

        $datos['contactos'] = $this->Modelo->getContactos($id); // Contactos asociados

        $datos['archivos'] = $this->Modelo->getArchivos($id); // Archivos relacionados

        $datos['requisitos'] = $this->Modelo->getRequisitos\_empresa($id); // Requisitos pendientes

        $datos['paises'] = $this->Modelo->listadopaises(); // Nuevamente países

        $datos['documento'] = $this->Modelo->documento(); // Documentos de referencia

        $this->load->view('header');

        $this->load->view('empresas/ver', $datos);

    }

    function registrar() {

    $ACIERTOS = array();

    $ERRORES = array();

    // Recolección y limpieza de datos del formulario

    $data = array(

        'nombre' => trim($this->input->post('nombre')),

        'nombre\_corto' => trim($this->input->post('nombre\_corto')),

        'razon\_social' => trim($this->input->post('razon\_social')),

        'giro' => trim($this->input->post('giro')),

        'clasificacion' => trim($this->input->post('clasificacion')),

        'rfc' => trim($this->input->post('rfc')),

        'calle' => trim($this->input->post('calle')),

        'numero' => trim($this->input->post('numero')),

        'numero\_interior' => trim($this->input->post('numero\_interior')),

        'colonia' => trim($this->input->post('colonia')),

        'calles\_aux' => trim($this->input->post('calles\_aux')),

        'pais' => trim($this->input->post('pais')),

        'estado' => trim($this->input->post('estado')),

        'ciudad' => trim($this->input->post('ciudad')),

        'cp' => trim($this->input->post('cp')),

        'foto' => 'default.png', // Imagen por defecto

        'ubicacion' => '',

        'cliente' => $this->input->post('cliente') != NULL ? '1' : '0',

        'proveedor' => $this->input->post('proveedor') != NULL ? '1' : '0',

        'credito\_cliente' => '0',

        'credito\_cliente\_plazo' => '0',

        'documentos\_facturacion' => '[]',

        'codigo\_impresion' => '',

        'comentarios' => '',

        'moneda\_cotizacion' => '[]',

        'iva\_cotizacion' => '[]',

        'notas\_cotizacion' => '',

        'contacto\_cotizacion' => '[]',

        'requisitos\_logisticos' => '',

        'requisitos\_documento' => '',

        'factura\_ejemplo' => '',

        'dejar\_factura' => 0,

        'prospecto' => $this->input->post('prospecto') != NULL ? '1' : '0',

    );

    // Intento de inserción de nueva empresa

    if ($this->Modelo->crear\_empresa($data)) {

        $acierto = array('titulo' => 'Agregar Empresa', 'detalle' => 'Se ha agregado Empresa con Éxito');

        array\_push($ACIERTOS, $acierto);

        // Recupera el ID recién insertado

        $res = $this->Conexion->consultar('SELECT MAX(id) as id FROM empresas', TRUE);

        // Registra en bitácora

        $data = array(

            'id\_empresa' => $res->id,

            'user' => $this->session->id,

            'estatus' => 'ALTA',

        );

        $this->Modelo->bitacoraEmpresas($data);

    } else {

        // En caso de error al insertar

        $error = array('titulo' => 'ERROR', 'detalle' => 'Error al agregar Empresa');

        array\_push($ERRORES, $error);

    }

    // Se guardan mensajes en sesión

    $this->session->aciertos = $ACIERTOS;

    $this->session->errores = $ERRORES;

    // Redirige al catálogo de empresas

    redirect(base\_url('empresas'));

}

    function editar() {

    $ACIERTOS = array();

    $ERRORES = array();

    $id = $this->input->post('id');

    // Recolección y limpieza de datos del formulario

    $data = array(

        'id' => trim($id),

        'nombre' => trim($this->input->post('nombre')),

        'giro' => trim($this->input->post('giro')),

        'razon\_social' => trim($this->input->post('razon\_social')),

        'rfc' => trim($this->input->post('rfc')),

        'calle' => trim($this->input->post('calle')),

        'numero' => trim($this->input->post('numero')),

        'numero\_interior' => trim($this->input->post('numero\_interior')),

        'colonia' => trim($this->input->post('colonia')),

        'calles\_aux' => trim($this->input->post('calles\_aux')),

        'cp' => trim($this->input->post('cp')),

        'pais' => trim($this->input->post('pais')),

        'estado' => trim($this->input->post('estado')),

        'ciudad' => trim($this->input->post('ciudad')),

        'ubicacion' => '',

        'cliente' => $this->input->post('cliente') != NULL ? '1' : '0',

        'proveedor' => $this->input->post('proveedor') != NULL ? '1' : '0',

        'prospecto' => $this->input->post('prospecto') != NULL ? '1' : '0',

    );

    // Actualiza la empresa

    if ($this->Modelo->update($data)) {

        $acierto = array('titulo' => 'Editar Empresa', 'detalle' => 'Se ha editado Empresa con Éxito');

        array\_push($ACIERTOS, $acierto);

        // Registro en bitácora

        $data = array(

            'id\_empresa' => trim($id),

            'user' => $this->session->id,

            'estatus' => 'Editar Empresa',

        );

        $this->Modelo->bitacoraEmpresas($data);

    } else {

        // Error al actualizar

        $error = array('titulo' => 'ERROR', 'detalle' => 'Error al editar Empresa');

        array\_push($ERRORES, $error);

    }

    // Almacena los mensajes en la sesión

    $this->session->aciertos = $ACIERTOS;

    $this->session->errores = $ERRORES;

    // Redirige a la vista de la empresa

    redirect(base\_url('empresas/ver/' . $id));

}

    function paises() {

    // Carga la vista del catálogo de países

    $this->load->view('header');

    $this->load->view('empresas/catalogo\_paises');

}

function ajax\_getEmpresas() {

    $texto = trim($this->input->post('texto'));

    $parametro = $this->input->post('parametro');

    $cliente = $this->input->post('cliente');

    $proveedor = $this->input->post('proveedor');

    // Consulta base: une empresas con proveedores

    $query = "SELECT E.\* FROM empresas E JOIN proveedores P ON E.id = P.empresa WHERE 1 = 1";

    // Filtro según tipo (cliente o proveedor)

    if ($cliente != $proveedor) {

        if ($cliente == "1") {

            $query .= " AND E.cliente = '1'";

        } else {

            $query .= " AND E.proveedor = '1'";

        }

    }

    // Filtros de búsqueda por parámetro

    if (!empty($texto)) {

        if ($parametro == "nombre") {

            $query .= " AND (E.nombre LIKE '%$texto%' OR E.razon\_social LIKE '%$texto%')";

        } else if ($parametro == "id") {

            $query .= " AND E.id = '$texto'";

        } else if ($parametro == "tag") {

            $query .= " AND P.tags LIKE '%$texto%'";

        }

    }

    // Ejecuta la consulta y retorna el resultado

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

    function ajax\_getContactos() {

    // Obtiene el ID de la empresa desde el POST

    $id = $this->input->post('id');

    $where['empresa'] = $id;

    // Consulta los contactos asociados a la empresa

    $res = $this->Conexion->get('empresas\_contactos', $where);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

function ajax\_getContacto() {

    // Obtiene el ID del contacto desde el POST

    $id = $this->input->post('id');

    // Consulta la información del contacto junto con el nombre de la planta si existe

    $res = $this->Conexion->consultar(

        "SELECT EC.\*, IFNULL(Pl.nombre, 'NO DEFINIDO') AS Planta

         FROM empresas\_contactos EC

         LEFT JOIN empresa\_plantas Pl ON Pl.id = EC.planta

         WHERE EC.id = $id",

         TRUE

    );

    if ($res) {

        echo json\_encode($res);

    }

}

function ajax\_getArchivos() {

    // Obtiene el ID de la empresa y texto de búsqueda desde el POST

    $empresa = $this->input->post('empresa');

    $texto = $this->input->post('texto');

    // Consulta base para obtener los archivos de la empresa con información del usuario y tipo de documento

    $query = "SELECT A.\*, CONCAT(U.nombre, ' ', U.paterno) AS User, D.tipo, D.clave

              FROM empresas\_archivos A

              INNER JOIN usuarios U ON U.id = A.usuario

              LEFT JOIN documentosEmpresas D ON D.id = A.id\_documento

              WHERE A.empresa = $empresa";

    // Filtro por texto si se proporciona

    if ($texto) {

        $query .= " AND (A.nombre LIKE '%$texto%' OR A.comentarios LIKE '%$texto%')";

    }

    // Ejecuta la consulta y responde con los resultados

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    }

}

    function ajax\_getFacturaEjemplo() {

    // Obtiene el ID de la empresa desde POST

    $empresa = $this->input->post('empresa');

    // Consulta el nombre del archivo de factura de ejemplo

    $query = "SELECT factura\_ejemplo FROM empresas WHERE id = $empresa";

    $res = $this->Conexion->consultar($query, TRUE);

    // Devuelve el nombre del archivo como respuesta

    echo $res->factura\_ejemplo;

}

function ajax\_setFacturaEjemplo() {

    // Obtiene datos desde POST

    $id = $this->input->post('empresa');

    $file = $this->input->post('file');

    // Si se recibió un archivo válido

    if ($file != "undefined") {

        $config['upload\_path'] = 'data/empresas/ejemplo\_facturas/';

        $config['allowed\_types'] = 'pdf';

        $config['encrypt\_name'] = TRUE;

        $this->load->library('upload', $config);

        // Intenta subir el archivo

        if ($this->upload->do\_upload('file')) {

            $data['factura\_ejemplo'] = $this->upload->data('file\_name');

            // Guarda el nombre del archivo en la base de datos

            $this->Conexion->modificar('empresas', $data, null, array('id' => $id));

        }

    }

}

function ajax\_deleteFacturaEjemplo() {

    // Obtiene ID de la empresa

    $id = $this->input->post('empresa');

    // Consulta el nombre del archivo actual

    $res = $this->Conexion->consultar("SELECT factura\_ejemplo FROM empresas WHERE id = $id", TRUE);

    // Elimina el archivo del servidor

    unlink('data/empresas/ejemplo\_facturas/' . $res->factura\_ejemplo);

    // Limpia el campo en la base de datos

    $data['factura\_ejemplo'] = "";

    $this->Conexion->modificar('empresas', $data, null, array('id' => $id));

}

    function editar\_otros\_datos() {

    // Inicializa arreglos para mensajes de éxito y error

    $ACIERTOS = array();

    $ERRORES = array();

    // Obtiene el ID de la empresa desde POST

    $id = $this->input->post('id');

    // Recoge los datos del formulario

    $data = array(

        'id' => $id,

        'horario\_facturas' => trim($this->input->post('horario\_facturas')),

        'ultimo\_dia\_facturas' => trim($this->input->post('ultimo\_dia\_facturas')),

        'requisitos\_logisticos' => trim($this->input->post('requisitos\_logisticos')),

        'requisitos\_documento' => trim($this->input->post('requisitos\_documento')),

        'comentarios' => trim($this->input->post('comentarios')),

        'dejar\_factura' => $this->input->post('dejar\_factura') == '1' ? '1' : '0',

    );

    // Intenta guardar los cambios

    if ($this->Modelo->update($data)) {

        array\_push($ACIERTOS, array('titulo' => 'Editar Empresa', 'detalle' => 'Se ha editado Empresa con Éxito'));

    } else {

        array\_push($ERRORES, array('titulo' => 'ERROR', 'detalle' => 'Error al editar Empresa'));

    }

    // Almacena mensajes y redirige

    $this->session->aciertos = $ACIERTOS;

    $this->session->errores = $ERRORES;

    redirect(base\_url('empresas/ver/' . $id));

}

// CONTACTOS

function agregarContacto() {

    // Recoge datos del formulario

    $empresa = $this->input->post('empresa');

    $data = array(

        'empresa' => trim($empresa),

        'nombre' => trim($this->input->post('nombre')),

        'telefono' => trim($this->input->post('telefono')),

        'ext' => trim($this->input->post('ext')),

        'celular' => trim($this->input->post('celular')),

        'celular2' => trim($this->input->post('celular2')),

        'correo' => trim($this->input->post('correo')),

        'puesto' => trim($this->input->post('puesto')),

        'red\_social' => trim($this->input->post('red\_social')),

        'activo' => $this->input->post('activo'),

        'cotizable' => $this->input->post('cotizable'),

        'planta' => $this->input->post('planta'),

    );

    // Inserta el nuevo contacto

    $res = $this->Modelo->insertContacto($data);

    // Registra en bitácora

    $r = $this->Conexion->consultar('SELECT \* FROM empresas\_contactos ORDER BY id DESC LIMIT 1', TRUE);

    $bitacora = array(

        'id\_contacto' => $r->id,

        'id\_empresa' => $r->empresa,

        'user' => $this->session->id,

        'estatus' => 'ALTA',

    );

    $this->Modelo->bitacoraContactosEmpresas($bitacora);

    // Devuelve el nuevo contacto en formato JSON

    if ($res) {

        $res = json\_encode($this->Modelo->getContacto($res));

    } else {

        $res = "";

    }

    echo $res;

}

    // Edita los datos de un contacto existente

function editarContacto() {

    $id = $this->input->post('id');

    // Prepara los datos recibidos del formulario

    $data = array(

        'id' => trim($id),

        'nombre' => trim($this->input->post('nombre')),

        'telefono' => trim($this->input->post('telefono')),

        'ext' => trim($this->input->post('ext')),

        'celular' => trim($this->input->post('celular')),

        'celular2' => trim($this->input->post('celular2')),

        'correo' => trim($this->input->post('correo')),

        'puesto' => trim($this->input->post('puesto')),

        'red\_social' => trim($this->input->post('red\_social')),

        'activo' => $this->input->post('activo'),

        'cotizable' => $this->input->post('cotizable'),

        'planta' => $this->input->post('planta'),

    );

    // Consulta la empresa del contacto para registrar en bitácora

    $r = $this->Conexion->consultar('SELECT \* FROM empresas\_contactos WHERE id = '.trim($id), TRUE);

    $dataBit = array(

        'id\_contacto' => trim($id),

        'id\_empresa' => $r->empresa,

        'user' => $this->session->id,

        'estatus' => 'Editar Contacto',

    );

    $this->Modelo->bitacoraContactosEmpresas($dataBit);

    // Actualiza el contacto y responde con el objeto actualizado

    if ($this->Modelo->updateContacto($data)) {

        $res = json\_encode($this->Modelo->getContacto($id));

    } else {

        $res = "";

    }

    echo $res;

}

// Retorna un contacto en formato JSON

function getContacto\_json(){

    $contact = $this->Modelo->getContacto($this->input->post('id'));

    if($contact){

        echo json\_encode($contact);

    } else {

        echo "";

    }

}

// Elimina un contacto y retorna "1" si fue exitoso, o cadena vacía si falló

function deleteContacto\_json(){

    if($this->Modelo->deleteContacto($this->input->post('id'))){

        echo "1";

    } else {

        echo ""; // jQuery interpreta como FALSE

    }

}

    // Sube o reemplaza la foto de una empresa

function subir\_foto() {

    $id = $this->input->post('id\_empresa');            // ID de la empresa

    $foto = $this->input->post('fotoActual');          // Nombre del archivo actual

    $file = $this->input->post('iptFoto');             // Archivo recibido

    if ($file != "undefined") {

        // Configuración para la carga del archivo

        $config['upload\_path'] = 'data/empresas/fotos/';

        $config['allowed\_types'] = 'gif|jpg|png';

        $config['encrypt\_name'] = TRUE;

        $this->load->library('upload', $config);

        // Si la carga del archivo fue exitosa

        if ($this->upload->do\_upload('iptFoto')) {

            $data['id'] = $id;

            $data['foto'] = $this->upload->data('file\_name');

            // Actualiza la foto en la base de datos

            if ($this->Modelo->update($data)) {

                // Elimina la foto anterior si no es la por defecto

                if ($foto != "default.png") {

                    unlink('data/empresas/fotos/' . $foto);

                }

                echo $data['foto'];

            }

        } else {

            echo "";

        }

    } else {

        // Si no se sube un archivo, se asigna la imagen por defecto

        $data['id'] = $id;

        $data['foto'] = 'default.png';

        if ($this->Modelo->update($data)) {

            if ($foto != "default.png") {

                unlink('data/empresas/fotos/' . $foto);

            }

            echo $data['foto'];

        }

    }

}

    // Sube un archivo relacionado a una empresa y lo registra en la base de datos

function subir\_archivo() {

    $id = $this->input->post('id');                        // ID de la empresa

    $comentarios = $this->input->post('comentarios');      // Comentarios del archivo

    $documento = $this->input->post('documento');          // Tipo de documento

    // Crea el directorio de la empresa si no existe

    if (!is\_dir('data/empresas/archivos/' . $id)) {

        mkdir('data/empresas/archivos/' . $id, 0777, TRUE);

    }

    // Configuración de subida

    $config['upload\_path'] = 'data/empresas/archivos/' . $id;

    $config['allowed\_types'] = '\*';

    $this->load->library('upload', $config);

    // Si la carga del archivo fue exitosa

    if ($this->upload->do\_upload('userfile')) {

        $data['empresa'] = $id;

        $data['usuario'] = $this->session->id;

        $data['nombre'] = $this->upload->data('file\_name');

        $data['comentarios'] = $comentarios;

        $data['id\_documento'] = $documento;

        $idFile = $this->Modelo->insertArchivo($data);

        if ($idFile) {

            $arreglo['id'] = $idFile;

            $arreglo['nombre'] = $data['nombre'];

            $arreglo['fecha'] = date('d/m/Y h:i A');

            $arreglo['icono'] = $this->aos\_funciones->file\_image($data['nombre']);

            $arreglo['error'] = '';

            echo json\_encode($arreglo);

        }

    } else {

        $arreglo['id'] = '0';

        $arreglo['nombre'] = 'ERROR';

        $arreglo['fecha'] = 'ERROR';

        // Se intenta generar un ícono aunque no haya nombre (puede generar warning si no hay archivo)

        $arreglo['icono'] = $this->aos\_funciones->file\_image($this->upload->data('file\_name') ?? '');

        $arreglo['error'] = $this->upload->display\_errors();

        echo json\_encode($arreglo);

    }

}

    // Elimina un archivo físico y su registro en la base de datos

function deleteArchivo\_json() {

    $id\_file = $this->input->post('id');

    $id\_empresa = $this->input->post('id\_empresa');

    $nombre = $this->input->post('nombre\_archivo');

    if ($this->Modelo->deleteArchivo($id\_file)) {

        unlink('data/empresas/archivos/' . $id\_empresa . '/' . $nombre);

        echo "1";

    } else {

        echo ""; // jQuery interpreta como FALSE (no 0)

    }

}

// Edita los metadatos de un archivo (comentarios y tipo de documento)

function editArchivo\_json() {

    $data['id'] = $this->input->post('id');

    $data['comentarios'] = trim($this->input->post('comentarios'));

    $data['id\_documento'] = $this->input->post('documento');

    if ($this->Modelo->updateArchivo($data)) {

        echo $data['comentarios'];

    } else {

        echo ""; // jQuery interpreta como FALSE (no 0)

    }

}

//////// REQUISITOS DE FACTURACIÓN //////////

// Carga la vista del catálogo de requisitos

function requisitos() {

    $data['requisitos'] = $this->Modelo->getRequisitos('');

    $this->load->view('header');

    $this->load->view('empresas/catalogo\_requisitos', $data);

}

// Retorna los requisitos de facturación filtrados por tipo

function getRequisitos\_json() {

    $tipo = $this->input->post('tipo');

    $requisitos = $this->Modelo->getRequisitos($tipo);

    if ($requisitos) {

        echo json\_encode($requisitos->result());

    } else {

        echo "";

    }

}

      // Asocia un requisito existente a una empresa

function setRequisitos\_json() {

    $data['empresa'] = $this->input->post('id\_empresa');

    $data['requisito'] = $this->input->post('requisito');

    $data['tipo'] = $this->input->post('tipo');

    $data['detalles'] = $this->input->post('detalles');

    $id = $this->Modelo->setRequisitos($data);

    if ($id) {

        $res['id'] = $id;

        $res['requisito'] = $data['requisito'];

        $res['detalles'] = $data['detalles'];

        echo json\_encode($res);

    } else {

        echo "";

    }

}

// Elimina un requisito vinculado a una empresa

function deleteRequisito\_json() {

    if ($this->Modelo->deleteRequisito($this->input->post('id'))) {

        echo "1";

    } else {

        echo ""; // jQuery interpreta como FALSE (no 0)

    }

}

// Agrega un nuevo requisito genérico al catálogo

function agregarRequisito\_ajax() {

    $data = array(

        'requisito' => strtoupper(trim($this->input->post('requisito'))),

        'tipo' => strtoupper(trim($this->input->post('tipo'))),

        'detalle' => $this->input->post('detalle'),

    );

    $res = $this->Modelo->insertRequisito($data);

    if ($res) {

        $data['id'] = $res;

        echo json\_encode($data);

    } else {

        echo "";

    }

}

    // Edita un requisito del catálogo general

function editarRequisito\_ajax() {

    $data = array(

        'id' => strtoupper(trim($this->input->post('id'))),

        'requisito' => strtoupper(trim($this->input->post('requisito'))),

        'tipo' => strtoupper(trim($this->input->post('tipo'))),

        'detalle' => $this->input->post('detalle'),

    );

    $res = $this->Modelo->updateRequisito($data);

    if ($res) {

        echo json\_encode($data);

    } else {

        echo "";

    }

}

// Elimina un requisito del catálogo general

function eliminarRequisito\_ajax() {

    $data = array(

        'id' => trim($this->input->post('id')),

    );

    $res = $this->Modelo->deleteRequisitoCatalogo($data);

    if ($res) {

        echo "1";

    } else {

        echo "";

    }

}

// Actualiza la ubicación geográfica de la empresa (latitud, longitud y nivel de zoom)

function editarUbicacion\_ajax() {

    $data = array(

        'id' => $this->input->post('id'),

        'lat' => $this->input->post('lat'),

        'lng' => $this->input->post('lng'),

        'zoom' => $this->input->post('zoom'),

    );

    $res = $this->Modelo->update($data);

    if ($res) {

        echo "1";

    } else {

        echo "";

    }

}

    // Actualiza el listado de documentos y código de impresión de una empresa

function ajax\_setListadoDocumentos(){

    $where['id'] = $this->input->post('id');

    $data['documentos\_facturacion'] = $this->input->post('documentos');

    $data['codigo\_impresion'] = strtoupper($this->input->post('codigo'));

    $this->Conexion->modificar('empresas', $data, null, $where);

    echo "1";

}

// Registra o actualiza los datos del proveedor asociados a la empresa

function ajax\_setProveedor(){

    $data = array(

        'empresa' => $this->input->post('id\_empresa'),

        'aprobado' => $this->input->post('aprobado'),

        'clasificacion\_proveedor' => $this->input->post('clasificacion\_proveedor'),

        'tipo' => $this->input->post('tipo'),

        'credito' => $this->input->post('credito'),

        'monto\_credito' => $this->input->post('monto\_credito'),

        'moneda\_credito' => $this->input->post('moneda\_credito'),

        'terminos\_pago' => $this->input->post('terminos\_pago'),

        'tags' => "," . $this->input->post('tags') . ",",

        'formas\_pago' => $this->input->post('formas\_pago'),

        'formas\_compra' => $this->input->post('formas\_compra'),

        'entrega' => $this->input->post('entrega'),

        'pasos\_cotizacion' => $this->input->post('pasos\_cotizacion'),

        'pasos\_compra' => $this->input->post('pasos\_compra'),

        'rma\_requerido' => $this->input->post('rma\_requerido'),

        'valResico' => $this->input->post('valResico'),

        'persona' => $this->input->post('persona'),

    );

    $res = $this->Modelo->setProveedor($data);

    // Registrar en bitácora la modificación del proveedor

    $data = array(

        'id\_empresa' => $this->input->post('id\_empresa'),

        'user' => $this->session->id,

        'estatus' => 'Editar Proveedor',

    );

    $this->Modelo->bitacoraEmpresas($data);

    if ($res) {

        echo "1";

    } else {

        echo "";

    }

}

    // Devuelve clientes cuyo nombre coincida parcialmente con el texto ingresado

function ajax\_getClientes(){

    $texto = $this->input->post('texto');

    $query = "SELECT E.\* FROM empresas E WHERE E.nombre LIKE '%" . $texto . "%'";

    $res = $this->Modelo->consulta($query);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

// Devuelve proveedores cuyo nombre o tags coincidan parcialmente con el texto ingresado

function ajax\_getProveedores(){

    $texto = $this->input->post('texto');

    $query = "SELECT E.\* FROM empresas E INNER JOIN proveedores P ON E.id = P.empresa WHERE E.proveedor = 1 AND (P.tags LIKE '%," . $texto . ",%' OR E.nombre LIKE '%" . $texto . "%')";

    $res = $this->Modelo->consulta($query);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

// Devuelve los datos del proveedor a partir de su ID

function ajax\_getProveedor(){

    $id = $this->input->post('id');

    $query = "SELECT P.\*, E.nombre FROM proveedores P INNER JOIN empresas E ON E.id = P.empresa WHERE P.empresa = '" . $id . "'";

    $res = $this->Modelo->consulta($query, TRUE);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

// Devuelve la lista de ciudades registradas en empresas (sin duplicados)

function ajax\_getCiudades(){

    $query = "SELECT DISTINCT ciudad FROM empresas";

    $res = $this->Modelo->consulta($query);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

    // Guarda la información de cotización (moneda, IVA, etc.) de una empresa

function ajax\_setInfoCotizaciones(){

    $datos = json\_decode($this->input->post('datos'));

    $datos->moneda\_cotizacion = json\_encode($datos->moneda\_cotizacion);

    $datos->iva\_cotizacion = json\_encode($datos->iva\_cotizacion);

    $this->Conexion->modificar('empresas', $datos, null, array('id' => $datos->id));

    echo "1";

}

// Obtiene los contactos cotizables de una empresa específica

function ajax\_getContactosCotizacion(){

    $empresa = $this->input->post('empresa');

    $res = $this->Conexion->consultar("SELECT EC.\*, IFNULL(Pl.nombre, 'NO DEFINIDO') as Planta FROM empresas\_contactos EC LEFT JOIN empresa\_plantas Pl ON Pl.id = EC.planta WHERE EC.empresa = '$empresa' AND EC.cotizable = 1");

    echo json\_encode($res);

}

// Crea o edita una planta asociada a la empresa

function ajax\_setPlanta(){

    $planta = json\_decode($this->input->post('planta'));

    if ($planta->id) {

        $this->Conexion->modificar('empresa\_plantas', $planta, null, array('id' => $planta->id));

    } else {

        $this->Conexion->insertar('empresa\_plantas', $planta);

    }

}

// Obtiene las plantas asociadas a una empresa

function ajax\_getPlantas(){

    $empresa = $this->input->post('empresa');

    $query = "SELECT \* FROM empresa\_plantas WHERE empresa = $empresa";

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    }

}

// Elimina una planta por su ID

function ajax\_deletePlanta(){

    $id = $this->input->post('id');

    $this->Conexion->eliminar('empresa\_plantas', array('id' => $id));

}

    // Devuelve el listado de estados del país seleccionado en formato <option>

function estados() {

    $pais = $this->input->post('paisid');

    if ($pais) {

        $edo = $this->Modelo->getestados($pais);

        foreach ($edo as $fila) {

            echo '<option value="'. $fila->estadonombre .'">'. $fila->estadonombre .'</option>';

        }

    } else {

        echo '<option value="0">Estados</option>';

    }

}

// Obtiene la bitácora de cambios de una empresa específica

function ajax\_getBitacoraEmpresas(){

    $ids = $this->input->post('id');

    $query = "SELECT CONCAT(u.nombre, ' ', u.paterno) as user, b.fecha, b.estatus FROM bitacora\_empresas b JOIN usuarios u ON b.user = u.id WHERE id\_empresa = ". $ids;

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    }

}

// Obtiene la bitácora de cambios de contactos de empresa

function ajax\_getBitacoraEmpresasContactos(){

    $ids = $this->input->post('id');

    $query = "SELECT CONCAT(u.nombre, ' ', u.paterno) as user, b.fecha, b.estatus FROM bitacora\_contactos\_empresas b JOIN usuarios u ON b.user = u.id WHERE id\_contacto = ". $ids;

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    }

}

// Devuelve lista de países según texto buscado, y si se desea solo los activos

function ajax\_paises(){

    $texto = trim($this->input->post('texto'));

    $activo = $this->input->post('activo');

    $all = null;

    if ($activo == 0) {

        $all = " AND activo = 1";

    }

    $query = "SELECT \* FROM pais WHERE 1 = 1 " . $all;

    if (!empty($texto)) {

        $query .= " AND paisnombre LIKE '%" . $texto . "%'";

    }

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

    // Activa un país estableciendo el campo 'activo' en 1

function ajax\_Activarpaises()

{

    $id = $this->input->post('id');

    $data['activo'] = '1';

    $this->Conexion->modificar('pais', $data, null, array('id' => $id));

    echo "1";

}

// Desactiva un país estableciendo el campo 'activo' en 0

function ajax\_Desactivarpaises()

{

    $id = $this->input->post('id');

    $data['activo'] = '0';

    $this->Conexion->modificar('pais', $data, null, array('id' => $id));

    echo "1";

}

// Obtiene todos los estados pertenecientes a un país específico

function ajax\_Estados()

{

    $id = $this->input->post('id');

    $query = "SELECT \* FROM estado WHERE paisid = " . $id;

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

// Asigna un estado como predeterminado (defecto = 1) dentro de un país

function ajax\_AsignarEstados()

{

    $id = $this->input->post('id');

    $idp = $this->input->post('idp');

    // Desactiva el estado predeterminado anterior del país

    $dato['defecto'] = '0';

    $this->Conexion->modificar('estado', $dato, null, array('paisid' => $idp));

    // Activa el nuevo estado predeterminado

    $data['defecto'] = '1';

    $this->Conexion->modificar('estado', $data, null, array('id' => $id));

    // Retorna el nuevo estado marcado como predeterminado

    $query = "SELECT \* FROM estado WHERE defecto = 1 AND id = " . $id;

    $res = $this->Conexion->consultar($query, TRUE);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

    // Retorna el estado marcado como defecto (predeterminado) para un país dado

function ajax\_setEstados()

{

    $id = $this->input->post('id');

    $query = "SELECT \* FROM estado WHERE defecto = 1 AND paisid = " . $id;

    $res = $this->Conexion->consultar($query, TRUE);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

  function excel\_empresas(){

        $texto = $this->input->post('texto');

        $texto = trim($texto);

        $parametro = $this->input->post('rbBusqueda');

        $cliente = $this->input->post('cliente');

        $proveedor = $this->input->post('proveedor');

        $c=0;

        $p=0;

        $fecha1 = $this->input->post('fecha1');

        $fecha2 = $this->input->post('fecha2');

        $f1=strval($fecha1).' 00:00:00';

        $f2=strval($fecha2).' 23:59:59' ;

        if ($cliente == 'cliente') {

                $c=1;

            }

            if ($proveedor == 'proveedor') {

                $p=1;

            }

$query = "SELECT c.id as cot, c.fecha,concat(uc.nombre, ' ', uc.paterno) usercot, c.tipo,P.entrega, c.estatus as estatus\_cot, e.calle, e.numero, e.estado, e.pais, e.nombre, e.razon\_social, e.ciudad, e.cliente, e.proveedor, (SELECT estatus from bitacora\_empresas WHERE id\_empresa = e.id AND estatus = 'ALTA') as estatus, (SELECT fecha from bitacora\_empresas WHERE id\_empresa = e.id AND estatus = 'ALTA') as fecha\_alta, (SELECT concat(u.nombre, ' ', u.paterno) from bitacora\_empresas be join usuarios u on u.id=be.user WHERE id\_empresa = e.id AND estatus = 'ALTA') as usuario\_alta FROM cotizaciones c JOIN empresas e on e.id=c.empresa JOIN proveedores P on e.id=P.empresa join usuarios uc on uc.id=c.responsable WHERE 1=1";

        if($c != $p)

        {

            if($cliente == "cliente")

            {

                $query .= " and e.cliente = '1'";

            }

            else if ($proveedor == "proveedor")

            {

                $query .= " and e.proveedor = '1'";

            }

        }

        if(!empty($texto))

        {

            if($parametro == "nombre")

            {

                $query .= " and (e.nombre like '%$texto%' or e.razon\_social like '%$texto%')";

            }

            else if($parametro == "id")

            {

                $query .= " and e.id = '$texto'";

            }

            else if($parametro == "tag")

            {

                $query .= " and P.tags like '%$texto%'";

            }

        }

        if (!empty($fecha1) && !empty($fecha2)) {

            $query .=" HAVING fecha\_alta BETWEEN '".$f1."' AND '".$f2."' ";

        }

        $query .= "  ORDER BY e.nombre ASC";

        $result= $this->Conexion->consultar($query);

        $salida='';

            $salida .= '<table style="border: 1px solid black; border-collapse: collapse;">

                            <thead>

                                <tr>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Nombre</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Razon Social</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Calle</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Ciduad</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Estado</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Pais</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Cliente</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Proveedor</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Entrega</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Fecha Alta</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Usuario Alta</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Cotizacion</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Fecha</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Estatus</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Responsable</th>

                                    <th style="background-color: #F3F1F1; color: black;  border: 1px solid black; border-collapse: collapse">Tipo</th>

                             </tr>

                            </thead>

                            <tbody>';

        foreach($result as $row){

            $ECLIENTE="";

            $EPROVEEDOR="";

            if ($row->cliente == 1) {

                $ECLIENTE="CLIENTE";

            }

            if ($row->proveedor == 1) {

                $EPROVEEDOR="PROVEEDOR";

            }

            $salida .='

                        <tr>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->nombre.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->razon\_social.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->calle." ".$row->numero.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->ciudad.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->estado.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->pais.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$ECLIENTE.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$EPROVEEDOR.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->entrega.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->fecha\_alta.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->usuario\_alta.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->cot.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->fecha.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->estatus\_cot.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->usercot.'</td>

                            <td style="color: $444;  border: 1px solid black; border-collapse: collapse">'.$row->tipo.'</td>

                        </tr>';

             }

                $salida .= '</tbody>

                </table>';

        $timestamp = date('m/d/Y', time());

        $filename='Empresas\_'.$timestamp.'.xls';

        header("Content-Type: application/vnd.ms-excel");

        header("Content-Disposition: attachment; filename=\"$filename\"");

        header("Pragma: no-cache");

        header("Expires: 0");

        header('Content-Transfer-Encoding: binary');

        echo $salida;

    }

}

## Equipos.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Equipos extends CI\_Controller {

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('tickets\_IT\_model','Modelo');

        $this->load->model('privilegios\_model');

    }

    // Vista principal del catálogo de equipos TI

    public function ti() {

        $datos['usuarios'] = $this->privilegios\_model->listadoJefes();

        $this->load->view('header');

        $this->load->view('equipos/ti/catalogo', $datos);

    }

    // Muestra historial de un equipo específico

    function historial($idequipo){

        $datosequipo = $this->Modelo->getEquipo($idequipo); // No usado en esta vista, podrías revisar si es necesario

        $datos['equipo'] = $this->Modelo->historial($idequipo);

        $this->load->view('header');

        $this->load->view('equipos/ti/historial', $datos);

    }

    // Vista general de revisiones de equipos

    function revisiones() {

        $datos['equipo'] = $this->Modelo->Equipos();

        $this->load->view('header');

        $this->load->view('equipos/ti/revisiones', $datos);

    }

    // Muestra la vista para realizar mantenimiento a un equipo específico

    function mantenimiento($idE) {

        $datos['equipo'] = $this->Modelo->getEquipos($idE);

        $this->load->view('header');

        $this->load->view('equipos/ti/mantenimiento', $datos);

    }

    // Muestra el historial de mantenimiento de un equipo

    function historialMantto($idE) {

        $datos['equipo'] = $this->Modelo->getEquipos($idE);

        $datos['manto'] = $this->Modelo->historialMantto($idE);

        $this->load->view('header');

        $this->load->view('equipos/ti/historialMantto', $datos);

    }

    // Vista de hallazgos para un mantenimiento específico

    public function hallazgos($iM) {

        $id = $datos['manto'] = $this->Modelo->Mantto($iM);

        $a = $id->idEquipo;

        $datos['equipo'] = $this->Modelo->getEquipos($a);

        $datos['fotos'] = $this->Modelo->fotosMantto($iM);

        $this->load->view('header');

        $this->load->view('equipos/ti/hallazgos', $datos);

    }

    function ajax\_setEquiposTI(){

    $equipo = json\_decode($this->input->post('equipo'));

    $file = $this->input->post('iptFoto');

    // Manejo de imagen anterior y subida de nueva imagen

    if($equipo->foto != "default.png") {

        unlink('data/equipos/ti/fotos/' . $equipo->foto);

    } else if($file == "undefined") {

        $equipo->foto = 'default.png';

    } else {

        $config['upload\_path'] = 'data/equipos/ti/fotos/';

        $config['allowed\_types'] = 'gif|jpg|png';

        $config['encrypt\_name'] = TRUE;

        $this->load->library('upload', $config);

        if ($this->upload->do\_upload('iptFoto')) {

            $equipo->foto = $this->upload->data('file\_name');

        }

    }

    // Tiempos de alta/asignación automáticos

    $funciones = array('fecha\_alta' => 'CURRENT\_TIMESTAMP()', 'fecha\_asignacion' => 'CURRENT\_TIMESTAMP()');

    if($equipo->id == 0) {

        // Inserta nuevo equipo

        $res = $this->Conexion->insertar('equipos\_it', $equipo, $funciones);

    } else {

        // Guarda histórico de asignación

        $query = 'INSERT INTO `equiposHis` (`idEqH`, `idequipo`, `idus`, `fecha`) VALUES (NULL, '.$equipo->id.', '.$equipo->asignado.', CURRENT\_TIMESTAMP);';

        $this->Conexion->comando($query);

        // Actualiza equipo existente

        $this->Conexion->modificar('equipos\_it', $equipo, $funciones, array('id' => $equipo->id));

    }

}

    function ajax\_getEquiposTI(){

    $texto = $this->input->post('texto');

    $tipo = $this->input->post('tipo');

    $inactivo = $this->input->post('inactivo');

    $asignado = $this->input->post('asignado');

    // Consulta base con unión para obtener el nombre del usuario asignado

    $query = "SELECT E.\*, ifnull(concat(U.nombre, ' ', U.paterno, ' ', U.materno), 'N/A') as Asignado from equipos\_it E left join usuarios U on U.id = E.asignado ";

    // Si se recibe un ID específico, se filtra directamente por él

    if(isset($\_POST['id'])){

        $id = $this->input->post('id');

        $query .= " where E.id = $id";

    }

    // Equipos inactivos por tipo

    else if ($inactivo == 1 && !empty($tipo)) {

        $query = "SELECT E.\*, ifnull(concat(U.nombre, ' ', U.paterno, ' ', U.materno), 'N/A') as Asignado from equipos\_it E left join usuarios U on U.id = E.asignado where E.activo = 0 and E.tipo like '".$tipo."'";

        if(isset($\_POST['id'])){

            $id = $this->input->post('id');

            $query .= " and E.id = $id";

        }

    }

    // Equipos inactivos por texto (buscando por ID)

    else if ($inactivo == 1 && !empty($texto)) {

        $query = "SELECT E.\*, ifnull(concat(U.nombre, ' ', U.paterno, ' ', U.materno), 'N/A') as Asignado from equipos\_it E left join usuarios U on U.id = E.asignado where E.activo = 0 and E.id =  '".$texto."'";

        if(isset($\_POST['id'])){

            $id = $this->input->post('id');

            $query .= " and E.id = $id";

        }

    }

    // Todos los equipos inactivos

    else if ($inactivo == 1) {

        $query = "SELECT E.\*, ifnull(concat(U.nombre, ' ', U.paterno, ' ', U.materno), 'N/A') as Asignado from equipos\_it E left join usuarios U on U.id = E.asignado where E.activo = 0";

        if(isset($\_POST['id'])){

            $id = $this->input->post('id');

            $query .= " and E.id = $id";

        }

    }

    // Búsqueda por campo JSON: No\_Inventario\_Interno o Serie (en caso de celulares)

    else if(!empty($texto)){

        $campo = '$.No\_Inventario\_Interno';

        if ($tipo == 'Celular') {

            $campo = '$.Serie';

        }

        $query = "SELECT E.\*, ifnull(concat(U.nombre, ' ', U.paterno, ' ', U.materno), 'N/A') as Asignado from equipos\_it E left join usuarios U on U.id = E.asignado where E.activo=1 and  JSON\_EXTRACT(campos, '".$campo."') = '".$texto."'";

        if(isset($\_POST['id'])){

            $id = $this->input->post('id');

            $query .= " and E.id = $id";

        }

    }

    // Filtro por tipo

    else if(!empty($tipo)){

        $query = "SELECT E.\*, ifnull(concat(U.nombre, ' ', U.paterno, ' ', U.materno), 'N/A') as Asignado from equipos\_it E left join usuarios U on U.id = E.asignado where E.activo=1 and E.tipo like '".$tipo."'";

        if(isset($\_POST['id'])){

            $id = $this->input->post('id');

            $query .= " and E.id = $id";

        }

    }

    // Filtro por usuario asignado

    else if(!empty($asignado)){

        $query = "SELECT E.\*, ifnull(concat(U.nombre, ' ', U.paterno, ' ', U.materno), 'N/A') as Asignado from equipos\_it E left join usuarios U on U.id = E.asignado where E.activo=1 and E.asignado = '".$asignado."'";

        if(isset($\_POST['id'])){

            $id = $this->input->post('id');

            $query .= " and E.id = $id";

        }

    }

    // Por defecto, mostrar todos los equipos activos

    else{

        $query .= " where E.activo=1";

    }

    // Ejecutar la consulta y retornar resultados en formato JSON

    $res = $this->Conexion->consultar($query, isset($\_POST['id']));

    echo json\_encode($res);

}

    function getEquiposTI(){

    $tipo = $this->input->post('tipo');

    $texto = $this->input->post('texto');

    // Consulta base con nombre del asignado y fecha de última revisión (Ultrev)

    $query = "SELECT E.\*, ifnull(concat(U.nombre, ' ', U.paterno, ' ', U.materno), 'N/A') as Asignado, ifnull((SELECT max(fecha) from manttoEquipos where idEquipo=E.id),'') as Ultrev FROM equipos\_it E LEFT JOIN usuarios U ON U.id = E.asignado where E.activo = 1";

    // Si se filtra por tipo, se construye la consulta según ese parámetro

    if(!empty($tipo)){

        $query = "SELECT E.\*, ifnull(concat(U.nombre, ' ', U.paterno, ' ', U.materno), 'N/A') as Asignado,  ifnull((SELECT max(fecha) from manttoEquipos where idEquipo=E.id),'') as Ultrev from equipos\_it E left join usuarios U on U.id = E.asignado where E.activo = 1 and E.tipo like '".$tipo."'";

    }

    // Se filtra por equipos cuya última revisión fue hace al menos 2 meses

    $query .= " HAVING `Ultrev` < last\_day(now()) + interval 1 day - interval 2 month";

    if(!empty($texto)){

        $campo = '$.No\_Inventario\_Interno';

        if ($tipo == 'Celular') {

            $campo = '$.Serie';

        }

        $query .= " and  JSON\_EXTRACT(campos, '".$campo."') = '".$texto."'";

    }

    $res = $this->Conexion->consultar($query);

    if($res){

        echo json\_encode($res);

    } else {

        echo "";

    }

}

    function equipos(){

    $texto = $this->input->post('texto');

    $texto = trim($texto);

    $inactivo = $this->input->post('inactivo');

    // Consulta base

    $query = "SELECT \* from equipos\_it";

    // Filtrado por estado de actividad

    if($activo!="1"){

        $query .="where activo=1";

    } elseif($activo=="1"){

        $query .="where activo=0";

    }

    $res = $this->Conexion->consultar($query);

    if($res)

    {

        echo json\_encode($res);

    }

    else{

        echo "";

    }

}

    function registrarMantto(){

    $equipo = $this->input->post('equipo');

    $id = null;

    $fototmp = $\_FILES['foto'];

    $fSize = intval(implode(".", $fototmp['size']));

    $countfoto = count($fototmp['name']);

    // Datos de mantenimiento recopilados del formulario

    $datos = array(

        'idEquipo' => $this->input->post('equipo'),

        'usMantto' => $this->session->id,

        'case' => $this->input->post('case'),

        'vidTem' => $this->input->post('vidTem'),

        'usb' => $this->input->post('usb'),

        'manosL' => $this->input->post('ml'),

        'bateria' => $this->input->post('bateria'),

        'datos' => $this->input->post('datos'),

        'comentarios' => $this->input->post('comentarios'),

        'disco' => $this->input->post('disco'),

        'cpu' => $this->input->post('cpu'),

        'tecMouse' => $this->input->post('tecMouse'),

        'abanicos' => $this->input->post('abanicos'),

    );

    $this->Modelo->registrarMantto($datos);

    echo var\_dump($datos);

    // Obtener el ID del mantenimiento recién registrado

    $query = "SELECT MAX(idME) as id FROM manttoEquipos WHERE idEquipo= '".$equipo."'";

    $res = $this->Conexion->consultar($query);

    foreach($res as $elem){

        $id = $elem->id;

    }

    // Guardar fotos si se cargaron

    if($fSize != 0){

        for ($i = 0; $i < $countfoto; $i++) {

            $fotofile = file\_get\_contents($fototmp['tmp\_name'][$i]);

            $data = array(

                'idMantto' => $id,

                'fotos' => $fotofile,

            );

            $id\_inserted = $this->Modelo->registrar($data);

        }

    }

    redirect(base\_url('equipos/revisiones'));

}

    function generador() {

    // Consulta registros del generador junto con el nombre del usuario asociado

    $datos['data'] = $this->Conexion->consultar("SELECT g.\*, concat(u.nombre, ' ', u.paterno) as name FROM generador g join usuarios u on u.id=g.usuario ORDER BY fecha DESC");

    $this->load->view('header');

    $this->load->view('equipos/ti/generador', $datos);

}

function registrar\_generador() {

    // Se obtiene la imagen del formulario como archivo binario

    $foto = file\_get\_contents($\_FILES['foto']['tmp\_name']);

    // Datos recopilados del formulario de registro

    $data = array(

        'fecha\_inicio' => $this->input->post('fecha\_inicio'),

        'fecha\_final' => $this->input->post('fecha\_final'),

        'duracion' => $this->input->post('duracion'),

        'foto' => $foto,

        'porcentaje' => $this->input->post('porcentaje') . "% capacidad",

        'amperaje' => $this->input->post('amperaje'),

        'usuario' => $this->session->id,

        'consumo' => $this->input->post('consumo') . " kW",

        'motor' => $this->input->post('motor'),

        'arranques' => $this->input->post('arranques'),

    );

    $this->Conexion->insertar('generador', $data);

    redirect(base\_url('equipos/generador'));

}

function uploadFoto() {

    // Actualiza la foto para un registro de generador específico

    $id = $this->input->post('id');

    $datos['foto'] = file\_get\_contents($\_FILES['file']['tmp\_name']);

    $this->Conexion->modificar('generador', $datos, null, array('id' => $id));

}

}

## Facturas.php

<?php

require\_once ('data/Merger/PDFMerger.php');

use PDFMerger\PDFMerger;

defined('BASEPATH') OR exit('No direct script access allowed');

class Facturas extends CI\_Controller {

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->library('correos\_facturacion'); // Carga librería para envío de correos de facturación

        $this->load->model('MLConexion\_model', 'MLConexion'); // Modelo para conexión con base de datos relacionada

    }

    function index(){

        $this->load->view('header');

        $this->load->view('facturas/catalogo\_facturas'); // Vista principal del catálogo de facturas

    }

    function solicitudes(){

        $this->load->view('header');

        $this->load->view('facturas/catalogo\_solicitudes'); // Vista del catálogo de solicitudes de factura

    }

    function solicitud\_factura(){

        $data["id"] = 0; // Para nueva solicitud, id = 0

        $this->load->view('header');

        $this->load->view('facturas/solicitud\_facturas', $data); // Vista para crear solicitud

    }

    function editar\_solicitud(){

        if (isset($\_POST["id"])) {

            $id = $this->input->post('id'); // Recupera ID desde POST

        } else if ($id == 0) {

            redirect(base\_url('inicio')); // Redirige si no hay ID válido

        }

        $data["id"] = $id;

        $data["editar"] = true; // Modo edición

        $this->load->view('header');

        $this->load->view('facturas/solicitud\_facturas', $data); // Vista para editar solicitud

    }

    function ver\_solicitud($id = 0){

        if (isset($\_POST["id"])) {

            $id = $this->input->post('id'); // Puede llegar también por POST

        } else if ($id == 0) {

            redirect(base\_url('inicio')); // Redirección si no hay ID

        }

        $data["id"] = $id;

        $this->load->view('header');

        $this->load->view('facturas/solicitud\_facturas', $data); // Vista para ver detalles de la solicitud

    }

    function documentos\_globales() {

    // Carga la vista de documentos globales de facturación

    $this->load->view('header');

    $this->load->view('facturas/documentos\_globales');

}

    function logistica() {

    // Carga la vista de logística dentro del módulo de facturación

    $this->load->view('header');

    $this->load->view('facturas/logistica');

}

    function programacion\_recorrido() {

    // Carga la vista de programación de recorrido de facturación

    $this->load->view('header');

    $this->load->view('facturas/programacion\_recorrido');

}

    function ajax\_setSolicitud() {

    $solicitud = json\_decode($this->input->post('solicitud'));

    $other = json\_decode($this->input->post('other'));

    $rs\_items = json\_decode($this->input->post('rs\_items'));

    // Archivos adjuntos (orden de compra y remisión)

    if (isset($\_FILES['f\_O'])) {

        $solicitud->f\_orden\_compra = file\_get\_contents($\_FILES['f\_O']['tmp\_name']);

    }

    if (isset($\_FILES['f\_R'])) {

        $solicitud->f\_remision = file\_get\_contents($\_FILES['f\_R']['tmp\_name']);

    }

    $funciones = array('fecha' => 'CURRENT\_TIMESTAMP()');

    $res = false;

    // Insertar o actualizar solicitud

    if ($solicitud->id == 0) {

        $solicitud->usuario = $this->session->id;

        $res = $this->Conexion->insertar('solicitudes\_facturas', $solicitud, $funciones);

        $solicitud->id = $res;

    } else {

        $res = $this->Conexion->modificar('solicitudes\_facturas', $solicitud, null, array('id' => $solicitud->id)) >= 0;

    }

    // Manejo de items relacionados

    $this->load->model('MLConexion\_model', 'MLConexion');

    foreach ($rs\_items as $item) {

        $borrar = $item->BORRAR;

        unset($item->BORRAR);

        if ($item->id == 0) {

            $rsitems = $this->MLConexion->consultar(

                "SELECT rs.folio\_id, rs.item\_id, rs.Fec\_CalibracionMT, s.DescripcionDeServicio, rs.HojadeDatos\_Id

                 FROM rsitems rs

                 JOIN catalogo\_servicios s ON s.servicio\_id = item\_servicio\_id

                 WHERE rs.item\_id = " . $item->item\_id,

                true

            );

            $item->servicio = $rsitems->DescripcionDeServicio;

            $item->HojadeDatos\_Id = $rsitems->HojadeDatos\_Id;

            $item->Fec\_CalibracionMT = $rsitems->Fec\_CalibracionMT;

            $item->id\_factura = $solicitud->id;

            $this->Conexion->insertar('rsitems\_facturas', $item);

            $this->MLConexion->modificar('rsitems', array('Solicitud\_ID' => $item->id\_factura), null, array('item\_id' => $item->item\_id));

        } else {

            if ($borrar) {

                $this->Conexion->eliminar('rsitems\_facturas', array('id' => $item->id));

                $this->MLConexion->modificar('rsitems', array('Solicitud\_ID' => 0), null, array('item\_id' => $item->item\_id));

            } else {

                $this->Conexion->modificar('rsitems\_facturas', $item, null, array('id' => $item->id));

            }

        }

    }

    // Preparar datos para enviar correo de notificación

    $solicitud->User = $other->User;

    $solicitud->Client = $other->Client;

    $solicitud->Contact = $other->Contact;

    $correos = [];

    $correos\_a = $this->Conexion->consultar("SELECT U.correo FROM privilegios P INNER JOIN usuarios U ON P.usuario = U.id WHERE P.responder\_facturas = 1");

    foreach ($correos\_a as $value) {

        array\_push($correos, $value->correo);

    }

    $solicitud->correos = array\_merge(array($this->session->correo), $correos);

    // Enviar correo

    $this->correos\_facturacion->solicitud($solicitud);

    if ($res) {

        echo "1";

    }

}

    function ajax\_getSolicitudes(){

        $id = $this->input->post('id');

        $aceptadas = $this->input->post('aceptadas');

        $cliente = $this->input->post('cliente');

        $ejecutivo = $this->input->post('ejecutivo');

        $texto = $this->input->post('texto');

        $parametro = $this->input->post('parametro');

        // Consulta base con joins a empresas y usuarios

        $query = "SELECT F.id, F.monto\_factura, F.fecha, F.usuario, F.ejecutivo, F.cliente, F.contacto, F.reporte\_servicio, F.orden\_compra, F.forma\_pago, F.pagada, F.urgente, F.conceptos, F.notas, F.estatus, F.documentos\_requeridos, F.serie, F.folio, F.codigo\_impresion, F.bitacora\_estatus, E.nombre as Cliente, concat(U.nombre, ' ', U.paterno) as User from solicitudes\_facturas F inner join empresas E on E.id = F.cliente inner join usuarios U on U.id = F.ejecutivo where 1 = 1";

        // Filtro por ID específico

        if($id)

        {

            $query .= " and F.id = '$id'";

        }

        else

        {

            // Búsqueda por texto en campo específico

            if($texto)

            {

                $query .= " and F.$parametro = '$texto'";

            }

            // Excluir solicitudes aceptadas o canceladas si $aceptadas es 0

            if($aceptadas == 0)

            {

                $query .= " and (F.estatus != 'ACEPTADO' && F.estatus != 'CANCELADO')";

            }

            // Filtro por cliente

            if($cliente && $cliente != 0)

            {

                $query .= " and F.cliente = '$cliente'";

            }

            // Filtro por ejecutivo

            if($ejecutivo && $ejecutivo != 0)

            {

                $query .= " and F.ejecutivo = '$ejecutivo'";

            }

        }

        // Filtro por estatus (si se manda por POST)

        if(isset($\_POST['estatus']))

        {

            $estatus = $this->input->post('estatus');

            $query .= " and F.estatus = '$estatus'";

        }

        // Filtro por logistica (si se manda por POST)

        if(isset($\_POST['logistica']))

        {

            $logistica = $this->input->post('logistica');

            $query .= " and F.logistica = '$logistica'";

        }

        $query .= " order by F.fecha desc";

        // Ejecutar y devolver resultados

        $res = $this->Conexion->consultar($query, $id);

        if($res)

        {

            echo json\_encode($res);

        }

    }

    function ajax\_editSolicitud(){

        $solicitud = json\_decode($this->input->post('solicitud'));

        $other = json\_decode($this->input->post('other'));

        // Procesar archivo de acuse (PDF)

        if(isset($\_FILES['f\_A']))

        {

            $solicitud->f\_acuse = file\_get\_contents($\_FILES['f\_A']['tmp\_name']);

        }

        // Procesar archivo de factura (PDF)

        if(isset($\_FILES['f\_F']))

        {

            $solicitud->f\_factura = file\_get\_contents($\_FILES['f\_F']['tmp\_name']);

            $solicitud->name\_factura = $this->input->post('f\_F\_name');

        }

        // Procesar archivo XML

        if(isset($\_FILES['f\_X']))

        {

            $solicitud->f\_xml = file\_get\_contents($\_FILES['f\_X']['tmp\_name']);

            $solicitud->name\_xml = $this->input->post('f\_X\_name');

        }

        $comentario = $this->input->post('comentario');

        // Actualizar solicitud en la base de datos

        $res = $this->Conexion->modificar('solicitudes\_facturas', $solicitud, null, array('id' => $solicitud->id));

        // Si la solicitud fue aceptada, actualiza la factura en los items relacionados

        if($solicitud->estatus\_factura == "ACEPTADO")

        {

            $this->load->model('MLConexion\_model', 'MLConexion');

            $this->MLConexion->comando("UPDATE rsitems set Factura = ifnull(Factura, $solicitud->folio) where Solicitud\_ID = $solicitud->id;");

        }

        if($res > 0)

        {

            // Si se agregó comentario, registrar en tabla de comentarios

            if(isset($\_POST['comentario']) && !empty($comentario))

            {

                $this->Conexion->insertar('solicitudes\_facturas\_comentarios', array('solicitud' => $solicitud->id, 'usuario' => $this->session->id, 'comentario' => $comentario), array('fecha' => 'CURRENT\_TIMESTAMP()'));

                $solicitud->comentario = $comentario;

            }

            // Obtener correos de los usuarios con privilegio para responder facturas

            $correos = [];

            $correos\_a = $this->Conexion->consultar("SELECT U.correo from privilegios P inner join usuarios U on P.usuario = U.id where P.responder\_facturas = 1");

            foreach ($correos\_a as $key => $value) {

                array\_push($correos, $value->correo);

            }

            // Preparar información adicional para el correo

            $solicitud->correos = array\_merge(array($this->session->correo), $correos);

            $solicitud->User = $other->User;

            $solicitud->Client = $other->Client;

            $solicitud->Contact = $other->Contact;

            $mail['id'] = $solicitud->id;

            $mail['estatus\_factura'] = $solicitud->estatus\_factura;

            $mail['comentario'] = $solicitud->comentario;

            $mail['User'] = $other->User;

            $mail['Client'] = $other->Client;

            $mail['Contact'] = $other->Contact;

            $mail['correos'] = array\_merge(array($this->session->correo), $correos);

            // Enviar notificación por correo

            $this->correos\_facturacion->editar\_solicitud($mail);

            echo "1";

        }

        else

        {

            echo "";

        }

}

    function ajax\_getReporteServicios() {

    $this->load->model('MLConexion\_model', 'MLConexion');

    $texto = $this->input->post('texto'); // Texto a buscar

    $rs = $this->input->post('rs'); // Folio del reporte de servicio

    // Consulta de items de un reporte de servicio con descripción extendida

    $query = "SELECT item\_id, folio\_id, descripcion, concat(descripcion, if(isnull(Fabricante), '', concat(' ', Fabricante)), if(isnull(Modelo), '', concat(' ', Modelo)), if(isnull(Serie), '', concat(' Serie: ', Serie)), if(isnull(Equipo\_ID), '', concat(' ID: ', Equipo\_ID))) as CadenaDescripcion, Solicitud\_ID, Monto from rsitems where 1=1 ";

    $query .= " and folio\_id = '$rs'";

    if($texto) {

        $query .= " having CadenaDescripcion like '%$texto%'";

    }

    $res = $this->MLConexion->Consultar($query);

    if($res) {

        echo json\_encode($res);

    }

}

function ajax\_getRSItems() {

    $id\_factura = $this->input->post('id\_factura'); // ID de la solicitud de factura

    $res = $this->Conexion->Consultar("SELECT \* from rsitems\_facturas where id\_factura = $id\_factura");

    if($res) {

        echo json\_encode($res);

    }

}

    function ajax\_setComentarios() {

    $comentario = json\_decode($this->input->post('comentario')); // Decodificar comentario recibido por POST

    $comentario->usuario = $this->session->id; // Agregar ID del usuario actual

    $funciones = array('fecha' => 'CURRENT\_TIMESTAMP()'); // Establecer fecha actual automáticamente

    // Insertar el comentario en la base de datos

    $res = $this->Conexion->insertar('solicitudes\_facturas\_comentarios', $comentario, $funciones);

    if($res > 0) {

        echo "1";

    } else {

        echo "";

    }

}

function ajax\_getComentarios() {

    $id = $this->input->post('id'); // ID de la solicitud de factura

    // Consulta de comentarios asociados a una solicitud

    $query = "SELECT C.\*, concat(U.nombre, ' ', U.paterno) as User from solicitudes\_facturas\_comentarios C inner join usuarios U on U.id = C.usuario where 1 = 1";

    if($id) {

        $query .= " and C.solicitud = '$id'";

    }

    $res = $this->Conexion->consultar($query);

    if($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

    function ajax\_getRequisitores() {

    $id = $this->input->post('id'); // ID opcional para filtrar un requisitor específico

    // Consulta a usuarios activos con privilegio para solicitar facturas

    $query = "SELECT U.id, concat(U.nombre, ' ', U.paterno) as Nombre, P.puesto as Puesto from usuarios U inner join puestos P on U.puesto = P.id inner join privilegios PR on PR.usuario = U.id where U.activo = 1 and PR.solicitar\_facturas = 1";

    if($id) {

        $query .= " and U.id = '$id'";

    }

    $res = $this->Conexion->consultar($query, $id);

    if($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

function ajax\_getVFPData() {

    $modelo = $this->input->post('modelo'); // Obtener modelo por POST

    // Ejecutar archivo externo de Visual FoxPro Reader con el modelo como argumento

    $res = shell\_exec("C:/xampp/htdocs/MASMetrologia/vfp\_reader/vfp\_reader.exe \"$modelo\"");

    echo $res;

}

    function archivo\_impresion() {

    // Inicializa el objeto para combinar PDFs

    $pdf = new PDFMerger;

    // Limpieza de búfer y configuración de errores

    ob\_start();

    error\_reporting(E\_ALL & ~E\_NOTICE);

    ini\_set('display\_errors', 0);

    ini\_set('log\_errors', 1);

    ob\_end\_clean();

    $id = $this->input->post('id');           // ID de la solicitud

    $codigo = $this->input->post('codigo');   // Códigos de documentos a incluir

    // Inicializa nuevamente el objeto PDFMerger por seguridad

    $pdf = new PDFMerger();

    // Consulta los datos de la solicitud

    $q = "SELECT SF.\* from solicitudes\_facturas SF where SF.id = $id";

    $res = $this->Conexion->consultar($q, TRUE);

    // Recorre el código de documentos y agrega los archivos PDF correspondientes

    for ($i = 0; $i < strlen($codigo); $i++) {

        switch (strtoupper($codigo[$i])) {

            case 'F': $campo = 'f\_factura'; break;

            case 'R': $campo = 'f\_remision'; break;

            case 'O': $campo = 'f\_orden\_compra'; break;

            case 'A': $campo = 'f\_acuse'; break;

            case 'P': $campo = 'OPINION'; break;

            case 'S': $campo = 'EMISI ON'; break;

            default:  $campo = null; break;

        }

        if ($campo != null) {

            if (substr($campo, 0, 2) == "f\_") {

                // Documentos adjuntos en la base de datos (binarios)

                $file = $res->$campo;

                $fichero = sys\_get\_temp\_dir() . '/' . $campo . '.pdf';

                file\_put\_contents($fichero, $file);

                $pdf->addPDF($fichero, 'all');

            } else {

                // Documentos globales predefinidos

                $fichero = "data/empresas/documentos\_globales/" . $campo . "\_000001.pdf";

                $pdf->addPDF($fichero, 'all');

            }

        }

    }

    ob\_end\_clean();

    $pdf->merge('browser'); // Genera y muestra el archivo combinado en el navegador

}

    function ajax\_getClientes() {

    // Consulta clientes activos marcados como tales

    $query = "SELECT C.id, C.nombre, C.razon\_social, C.foto, C.opinion\_positiva, C.emision\_sua from empresas C where C.cliente = 1";

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

function ajax\_getClientesSolicitudes() {

    $texto = $this->input->post('texto'); // Texto para filtrar por nombre de empresa

    // Consulta empresas con el número de solicitudes que tienen asociadas

    $query = "SELECT E.id, E.nombre, count(S.id) as NumSol from solicitudes\_facturas S inner join empresas E on E.id = S.cliente";

    if ($texto) {

        $query .= " where E.nombre like '%$texto%'";

    }

    $query .= " group by E.id;";

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    }

}

    function ajax\_getEjecutivosSolicitudes() {

    $texto = $this->input->post('texto'); // Texto para filtrar por nombre del ejecutivo

    // Consulta ejecutivos con el número de solicitudes que tienen asociadas

    $query = "SELECT U.id, concat(U.nombre, ' ', U.paterno) as Ejecutivo, count(S.id) as NumSol from solicitudes\_facturas S inner join usuarios U on U.id = S.ejecutivo";

    if ($texto) {

        $query .= " where concat(U.nombre, ' ', U.paterno) like '%$texto%'";

    }

    $query .= " group by U.id;";

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    }

}

    function ajax\_getDocumentosGlobales() {

    // Consulta los documentos globales (opinión positiva y emisión SUA)

    $query = "SELECT id, opinion\_positiva, emision\_sua from documentos\_globales where id = 1";

    $res = $this->Conexion->consultar($query);

    if ($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

function ajax\_filesExists($id) {

    $this->load->helper('file'); // Carga helper para manejo de archivos

    // Asegura que el ID tenga longitud de 6 caracteres, rellenando con ceros a la izquierda

    $id = str\_pad($id, 6, "0", STR\_PAD\_LEFT);

    // Verifica si existen los archivos PDF requeridos (ACUSE y EMISION)

    $acuse = read\_file(base\_url("data/empresas/documentos\_facturacion/ACUSE\_" . $id . ".pdf")) ? "1" : "0";

    $emision = read\_file(base\_url("data/empresas/documentos\_facturacion/EMISION\_" . $id . ".pdf")) ? "1" : "0";

    // Retorna el resultado como arreglo JSON (1 si existe, 0 si no)

    echo json\_encode(array($acuse, $emision));

}

    function ajax\_readXML() {

    // Crea un nuevo objeto DOMDocument para cargar y procesar el archivo XML

    $dom = new DomDocument;

    $dom->preserveWhiteSpace = FALSE; // Elimina espacios en blanco al procesar el XML

    $dom->loadXML(file\_get\_contents($\_FILES['f\_X']['tmp\_name'])); // Carga el contenido del archivo XML

    $comp = $dom->getElementsByTagName('Comprobante'); // Obtiene el nodo <Comprobante>

    $ext = 1; // Bandera para determinar si el folio ya existe

    $data = array(); // Arreglo para almacenar los atributos encontrados

    // Recorre los atributos del nodo <Comprobante>

    foreach ($comp[0]->attributes as $elem) {

        // Si el atributo es Serie, Folio o SubTotal, se agrega al arreglo $data

        if($elem->name == "Serie" | $elem->name == "Folio" | $elem->name == "SubTotal") {

            $e = array($elem->name => $elem->value);

            array\_push($data, $e);

        }

        // Si el atributo es Folio, se verifica si ya existe en la base de datos

        if($elem->name == "Folio") {

            $folio = $elem->value;

            $res = $this->Conexion->consultar("SELECT count(\*) as existe FROM solicitudes\_facturas where folio = '$folio'", TRUE);

            $ext = $res->existe;

        }

    }

    // Si el folio no existe, se devuelve el contenido extraído del XML

    if($ext == 0) {

        echo json\_encode($data);

    } else {

        echo "0"; // Si el folio ya existe, se devuelve "0"

    }

}

    function ajax\_setDocumentoFacturacion() {

    $file = $this->input->post('file');             // Nombre del archivo enviado

    $documento = $this->input->post('documento');   // Tipo de documento (EMISION u OPINION)

    $id = $this->input->post('empresa');            // ID de la empresa

    $id = str\_pad($id, 6, "0", STR\_PAD\_LEFT);        // Rellena el ID con ceros a la izquierda (mínimo 6 caracteres)

    // Verifica que se haya enviado un archivo válido

    if($file != "undefined") {

        // Configuración para la carga del archivo

        $config['upload\_path'] = 'data/empresas/documentos\_facturacion/';

        $config['allowed\_types'] = 'pdf';

        $config['overwrite'] = TRUE;

        $config['file\_name'] = $documento . '\_' . $id; // Nombre del archivo con prefijo del tipo de documento e ID

        $this->load->library('upload', $config);

        // Si la carga del archivo fue exitosa

        if ($this->upload->do\_upload('file')) {

            $where['id'] = $id;

            // Determina el campo correspondiente en la tabla según el tipo de documento

            switch($documento) {

                case "EMISION":

                    $campo = "emision\_sua";

                    break;

                case "OPINION":

                    $campo = "opinion\_positiva";

                    break;

            }

            // Guarda el nombre del archivo en el campo correspondiente de la base de datos

            $data[$campo] = $this->upload->data('file\_name');

            $this->Conexion->modificar('empresas', $data, null, $where);

            echo "1";

        }

    }

}

function ajax\_deleteDocumentoFacturacion(){

        $documento = $this->input->post('documento');                       // Tipo de documento: EMISION u OPINION

        $id = $this->input->post('empresa');                               // ID de la empresa

        $id = str\_pad($id, 6, "0", STR\_PAD\_LEFT);                          // Rellena el ID con ceros a la izquierda hasta tener 6 caracteres

        // Elimina el archivo físico del servidor

        unlink('data/empresas/documentos\_facturacion/' . $documento . '\_' . $id . '.pdf');

        $where['id'] = $id;

        // Determina el campo a limpiar en la base de datos según el tipo de documento

        switch($documento)

        {

            case "EMISION":

            $campo = "emision\_sua";

            break;

            case "OPINION":

            $campo = "opinion\_positiva";

            break;

        }

        // Limpia el campo correspondiente en la base de datos

        $data[$campo] = "";

        $this->Conexion->modificar('empresas', $data, null, $where);

}

    function ajax\_setDocumentoGlobal(){

        $file = $this->input->post('file');                                 // Archivo enviado desde el formulario

        $documento = $this->input->post('documento');                       // Tipo de documento: EMISION u OPINION

        $id = $this->input->post('empresa');                                // ID de la empresa

        $id = str\_pad($id, 6, "0", STR\_PAD\_LEFT);                           // Rellena con ceros a la izquierda hasta 6 dígitos

        if($file != "undefined")                                           // Verifica que haya un archivo definido

        {

            $config['upload\_path'] = 'data/empresas/documentos\_globales/'; // Ruta de almacenamiento

            $config['allowed\_types'] = 'pdf';                               // Solo permite archivos PDF

            $config['overwrite'] = TRUE;                                    // Sobrescribe si ya existe

            $config['file\_name'] = $documento . '\_' . $id;                  // Nombre del archivo final

            $this->load->library('upload', $config);                        // Carga la librería de subida

            if ($this->upload->do\_upload('file'))                           // Si se sube el archivo correctamente

            {

                $where['id'] = $id;

                // Determina qué campo actualizar en la tabla según el tipo de documento

                switch($documento)

                {

                    case "EMISION":

                    $campo = "emision\_sua";

                    break;

                    case "OPINION":

                    $campo = "opinion\_positiva";

                    break;

                }

                // Actualiza la base de datos con el nombre del archivo cargado

                $data[$campo] = $this->upload->data('file\_name');

                $this->Conexion->modificar('documentos\_globales', $data, null, $where);

                echo "1";

            }

        }

}

    function ajax\_deleteDocumentoGlobal(){

        $documento = $this->input->post('documento');                                  // Tipo de documento: EMISION u OPINION

        $id = $this->input->post('empresa');                                           // ID de la empresa

        $id = str\_pad($id, 6, "0", STR\_PAD\_LEFT);                                      // Rellena con ceros a la izquierda hasta 6 dígitos

        // Elimina el archivo PDF correspondiente en la carpeta de documentos globales

        unlink('data/empresas/documentos\_globales/' . $documento . '\_' . $id . '.pdf');

        $where['id'] = $id;

        // Determina qué campo de la base de datos se debe limpiar según el documento

        switch($documento)

        {

            case "EMISION":

            $campo = "emision\_sua";

            break;

            case "OPINION":

            $campo = "opinion\_positiva";

            break;

        }

        // Limpia el campo del documento en la base de datos

        $data[$campo] = "";

        $this->Conexion->modificar('documentos\_globales', $data, null, $where);

}

    function ajax\_enviarCorreo(){

        $id = $this->input->post('id');                                        // ID de la solicitud

        $body = $this->input->post('body');                                    // Cuerpo del correo

        $subject = $this->input->post('subject');                              // Asunto del correo

        $para = $this->input->post('para');                                    // Destinatario(s)

        $cc = $this->input->post('cc');                                        // Copia(s)

        $campos = json\_decode($this->input->post('campos'));                  // Campos a adjuntar (archivos)

        $archivos = [];

        $res = $this->Conexion->consultar("SELECT SF.\* from solicitudes\_facturas SF where SF.id = $id", TRUE);  // Consulta la solicitud

        foreach ($campos as $value) {

            if(substr($value, 0, 2 ) == "f\_")                                  // Archivos directamente en la tabla (PDF o XML)

            {

                $file = $res->$value;

                $fichero = sys\_get\_temp\_dir(). '/' . $value . ($value == "f\_xml" ? '.xml' : '.pdf');

                file\_put\_contents($fichero, $file);

            }

            else                                                               // Archivos globales por nombre fijo

            {

                switch ($value) {

                    case 'opinion\_positiva':

                        $value = 'OPINION';

                        break;

                    case 'emision\_sua':

                        $value = 'EMISION';

                        break;

                }

                $fichero = "data/empresas/documentos\_globales/" . $value . "\_000001.pdf";

            }

            array\_push($archivos, $fichero);                                   // Agrega ruta del archivo al array de adjuntos

        }

        // Arma el paquete de datos para enviar por correo

        $datos['id'] = $id;

        $datos['para'] = $para;

        $datos['cc'] = $cc;

        $datos['subject'] = $subject;

        $datos['body'] = $body;

        $datos['campos'] = $campos;

        $datos['archivos'] = $archivos;

        $this->correos\_facturacion->enviarCorreo($datos);

}

    ////////////////////////////////// F A C T U R A S //////////////////////////////////

function ajax\_getFacturas(){

        $id = $this->input->post('id');

        // Consulta de facturas con detalles de cliente, usuario y número de recorridos/envíos

        $query = "SELECT F.id, F.fecha, F.usuario, F.cliente, F.contacto, F.reporte\_servicio, F.orden\_compra, F.forma\_pago, F.pagada, F.conceptos, F.notas, F.estatus\_factura, F.documentos\_requeridos, F.serie, F.folio, F.codigo\_impresion, (SELECT count(id) from recorrido\_conceptos where id\_concepto = F.id and tipo = 'FACTURA') as Recorridos, (SELECT count(id) from envios\_factura where factura = F.id) as Envios, E.nombre as Cliente, concat(U.nombre, ' ', U.paterno) as User, U.correo, ifnull(EC.correo, 'N/A') as CorreoContacto from solicitudes\_facturas F inner join empresas E on E.id = F.cliente inner join usuarios U on U.id = F.usuario left join empresas\_contactos EC on EC.id = F.contacto";

        $query .= " where F.folio > 0 and F.estatus = 'ACEPTADO'"; // Solo facturas aceptadas con folio

        if($id)

        {

            $query .= " and F.id = '$id'";

        }

        if(isset($\_POST['estatus']))

        {

            $estatus = $this->input->post('estatus');

            $query .= " and F.estatus = '$estatus'";

        }

        $res = $this->Conexion->consultar($query, $id);

        if($res)

        {

            echo json\_encode($res);

        }

}

////////////////////////////////// L O G I S T I C A //////////////////////////////////

function ajax\_getMensajeros(){

        // Obtiene usuarios activos con privilegio de mensajero

        $query = "SELECT U.id, U.nombre, U.paterno, U.materno, U.no\_empleado, U.puesto, U.correo, U.ultima\_sesion, U.departamento, U.activo, U.jefe\_directo, U.autorizador\_compras, U.autorizador\_compras\_venta, concat(U.nombre, ' ', U.paterno) as User, concat(U.nombre, ' ', U.paterno, ' ', U.materno) as CompleteName from usuarios U inner join privilegios P on P.usuario = U.id where U.activo = '1'";

        $query .= " and P.mensajero = '1'";

        $query .= " order by User";

        $res = $this->Conexion->consultar($query);

        if($res)

        {

            echo json\_encode($res);

        }

}

    function ajax\_setRecorrido(){

        // Obtener datos desde POST

        $mensajero = $this->input->post('mensajero');

        $fecha = $this->input->post('fecha');

        $recorrido = json\_decode($this->input->post('recorrido'));

        // Insertar encabezado del recorrido

        $data['mensajero'] = $mensajero;

        $data['fecha\_recorrido'] = $fecha;

        $recorrido\_id = $this->Conexion->insertar('recorridos', $data);

        // Insertar detalles del recorrido y actualizar estatus de factura

        foreach ($recorrido as $value) {

            $data2['recorrido'] = $recorrido\_id;

            $data2['factura'] = $value[1];

            $data2['accion'] = $value[0];

            $data2['estatus'] = "EN RECORRIDO";

            $this->Conexion->insertar('recorrido\_conceptos', $data2);

            $this->Conexion->modificar('solicitudes\_facturas', array('estatus' => $data2['estatus']), null, array('id' => $data2['factura']));

        }

}

function ajax\_getRecorridos(){

        $pendientes = $this->input->post('pendientes');

        $factura = $this->input->post('factura');

        // Consulta de recorridos con detalles adicionales como comentarios, pendientes y nombre del cliente/mensajero

        $query = "SELECT RF.\*, R.mensajero, R.fecha\_recorrido, (SELECT count(RC.id) from recorrido\_comentarios RC where RC.recorrido\_factura = RF.id) as Comentarios, (SELECT count(RF2.id) from recorrido\_facturas RF2 where RF2.recorrido = RF.recorrido and RF2.estatus = 'EN RECORRIDO') as Pendientes, (SELECT E.nombre from solicitudes\_facturas F inner join empresas E on E.id = F.cliente where F.id = RF.factura) as Cliente, ifnull(concat(M.nombre, ' ', M.paterno), 'N/A') as Mensajero from recorrido\_facturas RF inner join recorridos R on R.id = RF.recorrido left join usuarios M on M.id = R.mensajero where 1 = 1";

        // Filtrar por factura si se proporciona

        if(isset($\_POST['factura']))

        {

            $query .= " and RF.factura = $factura";

        }

        // Mostrar solo recorridos con pendientes si se solicita

        if($pendientes == "1")

        {

            $query .= " having Pendientes > 0";

        }

        $query .= " order by R.fecha\_recorrido, R.id, RF.id asc";

        $res = $this->Conexion->consultar($query);

        echo json\_encode($res);

}

    function ajax\_updateRecorrido(){

        // Decodificar datos recibidos por POST

        $recorrido = json\_decode($this->input->post('recorrido'));

        $recolecta = $this->input->post('recolecta');

        $comentario = $this->input->post('comentario');

        // Actualizar registro de recorrido\_facturas

        $this->Conexion->modificar('recorrido\_facturas', $recorrido, null, array('id' => $recorrido->id));

        // Determinar estatus a aplicar en la tabla solicitudes\_facturas

        if(substr($recorrido->estatus, 0, 2) == "NO")

        {

            $estat = "PENDIENTE " . $recorrido->accion;

            $this->Conexion->modificar('solicitudes\_facturas', array('estatus' => $estat), null, array('id' => $recorrido->factura));

        }

        else

        {

            $estat = $recolecta == "1" ? "PENDIENTE RECOLECTA" : "CERRADO";

            $this->Conexion->modificar('solicitudes\_facturas', array('estatus' => $estat), null, array('id' => $recorrido->factura));

        }

        // Si se envió comentario, registrar con color y estatus

        if($comentario)

        {

            $color = substr($recorrido->estatus, 0, 2) == "NO" ? "red" : "green";

            $comentario = '<font color=' . $color . '><b>' . $recorrido->estatus . ':</b></font> ' . $comentario;

            $data\_com['recorrido\_factura'] = $recorrido->id;

            $data\_com['usuario'] = $this->session->id;

            $data\_com['comentario'] = $comentario;

            $func\_com['fecha'] = "CURRENT\_TIMESTAMP()";

            $this->Conexion->insertar('recorrido\_comentarios', $data\_com, $func\_com);

        }

}

    function ajax\_getComentariosRecorrido(){

        // Obtener el ID de la factura de recorrido desde POST

        $id = $this->input->post('id');

        // Construir consulta para obtener los comentarios junto con el nombre del usuario

        $query = "SELECT C.\*, concat(U.nombre, ' ', U.paterno) as User from recorrido\_comentarios C inner join usuarios U on U.id = C.usuario where 1 = 1";

        // Filtro por ID

        if($id)

        {

            $query .= " and C.recorrido\_factura = '$id'";

        }

        $query .= " order by C.fecha";

        $res = $this->Conexion->consultar($query);

        if($res)

        {

            echo json\_encode($res);

        }

    }

    function ver\_POD($id){

        // Consulta para obtener datos de la solicitud de factura

        $query = "SELECT sf.id, sf.folio,sf.serie, sf.reporte\_servicio, sf.fecha, sf.orden\_compra, concat(u.nombre, ' ', u.paterno) as responsable, e.razon\_social, concat(e.calle, ' ',e.numero, ' CP ',e.cp) as direccion,e.ciudad, e.estado, e.rfc,e.colonia, ec.nombre as contacto, ec.correo FROM solicitudes\_facturas sf join usuarios u on sf.ejecutivo = u.id JOIN empresas e on sf.cliente = e.id JOIN empresas\_contactos ec on sf.contacto = ec.id WHERE sf.id =$id";

        $factura = $this->Conexion->consultar($query, TRUE);

        // Consulta para obtener los items relacionados a la solicitud

        $query2 = "SELECT rs.descripcion, rs.Equipo\_ID, rs.Fec\_CalibracionMT, s.DescripcionDeServicio, concat(rs.descripcion,if(isnull(rs.Fabricante), '', concat(' ', rs.Fabricante)), if(isnull(rs.Modelo), '', concat(' ', rs.Modelo)), if(isnull(rs.Serie), '', concat(' Serie: ', rs.Serie)) ) as CadenaDescripcion from rsitems rs JOIN catalogo\_servicios s on s.servicio\_id = rs.item\_servicio\_id WHERE rs.Solicitud\_ID =".$id." and rs.Factura = '".$factura->folio."'";

        $rs = $this->MLConexion->consultar($query2);

        // Contar total de equipos

        $total=0;

        foreach($rs as $row){

            $total++;

        }

        ini\_set('display\_errors', 0);

        $this->load->library('pdfview');

        $pdf = new TCPDF(PDF\_PAGE\_ORIENTATION, PDF\_UNIT, PDF\_PAGE\_FORMAT, true, 'UTF-8', false);

        $f=$factura->serie . "-".$factura->folio;

        $pdf->SetCreator(PDF\_CREATOR);

        $pdf->SetAuthor('AleksOrtiz');

        $pdf->SetTitle('Masmetrologia');

        $pdf->SetSubject('Formato Cotización');

        // Encabezado del PDF

        $spc = "           ";

        $head = "$spc           Prueba de Entrega      $spc Folio: $id / Factura: $f";

        $txt = "                             Proof of delivery                       Responsable: " . $factura->responsable;

        $txt .= "\n                                                                                Fec. de elaboración:: " . $factura->fecha;

        $pdf->SetHeaderData(PDF\_HEADER\_LOGO\_ORIGINAL, '40', $head, $txt);

        // Configuración general de fuentes y márgenes

        $pdf->setHeaderFont(Array(PDF\_FONT\_NAME\_MAIN, '', 10));

        $pdf->setFooterFont(Array(PDF\_FONT\_NAME\_DATA, '', PDF\_FONT\_SIZE\_DATA));

        $pdf->SetDefaultMonospacedFont(PDF\_FONT\_MONOSPACED);

        $pdf->SetMargins(8, PDF\_MARGIN\_TOP, 8);

        $pdf->SetHeaderMargin(PDF\_MARGIN\_HEADER);

        $pdf->SetFooterMargin(PDF\_MARGIN\_FOOTER);

        $pdf->SetAutoPageBreak(TRUE, PDF\_MARGIN\_BOTTOM);

        $pdf->setImageScale(PDF\_IMAGE\_SCALE\_RATIO);

        // Soporte para idioma inglés

        if (@file\_exists(dirname(\_\_FILE\_\_).'/lang/eng.php')) {

            require\_once(dirname(\_\_FILE\_\_).'/lang/eng.php');

            $pdf->setLanguageArray($l);

        }

        $pdf->SetFont('helvetica', '', 8);

        $pdf->AddPage();

        $pdf->SetFillColor(255, 255, 255);

$pdf->SetFont('helvetica', '', 10);

        // Tabla con los datos del cliente

        $tbl = <<<EOD

        <br>

            <table border="0">

                <tr>

                    <td>

                        <b>Cliente/Customer:</b><br>

                        $factura->razon\_social<br>

                        $factura->direccion<br>

                        $factura->colonia<br>

                        $factura->ciudad, $factura->estado<br>

                        RFC: $factura->rfc<br>

                    </td>

                    <td>

EOD;

        $tbl .= <<<EOD

        <b>Orden de Compra: </b>

        $factura->orden\_compra<br>

        <b>Contacto / Contact: </b><br>

        $factura->contacto<br>

        $factura->correo<br>

        <br>

        Total de Equipo(s):  $total<br>

        </td>

        </tr>

    </table>

EOD;

        $pdf->writeHTML($tbl, false, false, false, false, '');

        $w = array(8, 125, 12, 24, 24);

        // Configuración de la tabla de servicios

        $pdf->SetFont('helvetica', 'B', 9);

        $pdf->SetTextColor(255);

        $pdf->Ln();$pdf->Ln();

        $w = array(20, 20, 125, 20);

        $pdf->SetTextColor(0);

        $pdf->SetFont('helvetica', '', 10);

            $pdf->Ln();

        $tabla\_items='';

            // Construcción HTML de la tabla con los equipos

            $tabla\_items .= '<table style=" ">

                            <thead>

                                <tr>

                                    <th style="border-bottom: 1px solid #000; text-align: center; font-weight: bold; width: 15%;">Servicio</th>

                                    <th style="border-bottom: 1px solid #000; text-align: center; font-weight: bold; width: 15%;">Equipo ID</th>

                                    <th style="border-bottom: 1px solid #000; text-align: center; font-weight: bold; width: 55%;">Descripcion</th>

                                    <th style="border-bottom: 1px solid #000; text-align: center; font-weight: bold; width: 15%;">Realizado</th>

                                </tr>

                            </thead>

                            <tbody>';

         foreach($rs as $row){

            $date=date\_create($row->Fec\_CalibracionMT);

            $tabla\_items .='

                        <tr>

                            <td style="border-bottom: 1px solid #000; text-align: center; width: 15%; tr:nth-child:background: #F8F8F8;">'.$row->DescripcionDeServicio .'</td>

                            <td style="border-bottom: 1px solid #000; text-align: center; width: 15%; tr:nth-child:background: #F8F8F8;">'.$row->Equipo\_ID.'</td>

                            <td style="border-bottom: 1px solid #000; text-align:center ; width: 55%; tr:nth-child:background: #F8F8F8;">'.$row->CadenaDescripcion.'</td>

                            <td style="border-bottom: 1px solid #000; text-align: center;width: 15%; tr:nth-child:background: #F8F8F8;">'.date\_format($date,'d/M/Y').'</td>

                        </tr>';

             }

             $tabla\_items .= '</tbody>

                </table>

                <br>

                <br>

                <br>

                <br>

                <br>

                <br> ';

        // Renderizar la tabla de items

        $pdf->writeHTML($tabla\_items, true, false, false, false, '');

        $pdf->Ln();

        $pdf->Ln();

        $pdf->SetFont('helvetica', '', 10);

        $pdf->MultiCell(150, 8,"Recibí el servicio a los equipos arriba listados", 0, 'L', 1, 0, '', '', true, 0, false, true, 0);

        $pdf->SetFont('helvetica', 'B', 10);

        $pdf->writeHTML("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", true, false, false, false, '');

        $pdf->MultiCell(150, 8,"Firma de recibido", 0, 'L', 1, 0, '', '', true, 0, false, true, 0);

        $pdf->Ln();

        // Nombre del archivo generado

        $name = "POD ".$id." - PO ".$factura->orden\_compra.'.pdf';

        // Salida del archivo PDF al navegador

        $pdf->Ln();

        $pdf->Output(sys\_get\_temp\_dir().'/'.$name, 'I');

    }

function ajax\_enviarCorreoPOD(){

        // Se obtienen datos del formulario (POST)

        $id = $this->input->post('id');

        $body = $this->input->post('body');

        $subject = $this->input->post('subject');

        $para = $this->input->post('para');

        $cc = $this->input->post('cc');

        $certificados=[];

        $pdfname=null;

        $xmlname =null;

        $campos = json\_decode($this->input->post('campos'));

        $archivos = [];

        $name\_files = [];

        // Se consulta la solicitud de factura correspondiente al ID

        $res = $this->Conexion->consultar("SELECT SF.\* from solicitudes\_facturas SF where SF.id = $id", TRUE);

        $pdfname=$res->name\_factura;

        $xmlname=$res->name\_xml;

        // Se preparan los archivos adjuntos

        foreach ($campos as $value) {

            if(substr($value, 0, 2 ) == "f\_")

            {

                $file = $res->$value;

                $fichero = sys\_get\_temp\_dir(). '/' . $value . ($value == "f\_xml" ? '.xml' : '.pdf');

                $name =$value == "f\_xml" ? $res->name\_xml : $res->name\_factura;

                 file\_put\_contents($fichero, $file);

            }

            else

            {

                switch ($value) {

                    case 'opinion\_positiva':

                        $value = 'OPINION';

                        break;

                    case 'emision\_sua':

                        $value = 'EMISION';

                        break;

                }

                $fichero = "data/empresas/documentos\_globales/" . $value . "\_000001.pdf";

            }

            array\_push($archivos, $fichero);

        }

        $query = "SELECT sf.id, sf.folio,sf.serie, sf.reporte\_servicio, sf.fecha, sf.orden\_compra, concat(u.nombre, ' ', u.paterno) as responsable, e.razon\_social, concat(e.calle, ' ',e.numero, ' CP ',e.cp) as direccion,e.ciudad, e.estado, e.rfc,e.colonia, ec.nombre as contacto, ec.correo FROM solicitudes\_facturas sf join usuarios u on sf.ejecutivo = u.id JOIN empresas e on sf.cliente = e.id JOIN empresas\_contactos ec on sf.contacto = ec.id WHERE sf.id =$id";

        $factura = $this->Conexion->consultar($query, TRUE);

        // Consulta para obtener los equipos relacionados a la factura

        $query2 = "SELECT rs.descripcion, rs.Equipo\_ID, rs.documento\_id, rs.Fec\_CalibracionMT, s.DescripcionDeServicio, concat(rs.descripcion,if(isnull(rs.Fabricante), '', concat(' ', rs.Fabricante)), if(isnull(rs.Modelo), '', concat(' ', rs.Modelo)), if(isnull(rs.Serie), '', concat(' Serie: ', rs.Serie)) ) as CadenaDescripcion from rsitems rs JOIN catalogo\_servicios s on s.servicio\_id = rs.item\_servicio\_id WHERE rs.Solicitud\_ID =".$id." and rs.Factura = '".$factura->folio."'";

         $rs = $this->MLConexion->consultar($query2);

         $total=0;

        foreach($rs as $row){

            $total++;

        }

        // Se recopilan IDs de documentos para certificados

        foreach ($rs as $key) {

        array\_push($certificados, $key->documento\_id);

    }

        ini\_set('display\_errors', 0);

        $this->load->library('pdfview');

        // Generación del archivo PDF tipo POD

        $pdf = new TCPDF(PDF\_PAGE\_ORIENTATION, PDF\_UNIT, PDF\_PAGE\_FORMAT, true, 'UTF-8', false);

        $f=$factura->serie . "-".$factura->folio;

        $pdf->SetCreator(PDF\_CREATOR);

        $pdf->SetAuthor('AleksOrtiz');

        $pdf->SetTitle('Masmetrologia');

        $pdf->SetSubject('Formato Cotización');

        $spc = "           ";

        $head = "$spc           Prueba de Entrega      $spc Folio: $id / Factura: $f";

        $txt = "                             Proof of delivery                       Responsable: " . $factura->responsable;

        $txt .= "\n                                                                                Fec. de elaboración:: " . $factura->fecha;

        $pdf->SetHeaderData(PDF\_HEADER\_LOGO\_ORIGINAL, '40', $head, $txt);

        $pdf->setHeaderFont(Array(PDF\_FONT\_NAME\_MAIN, '', 10));

        $pdf->setFooterFont(Array(PDF\_FONT\_NAME\_DATA, '', PDF\_FONT\_SIZE\_DATA));

        $pdf->SetDefaultMonospacedFont(PDF\_FONT\_MONOSPACED);

        $pdf->SetMargins(8, PDF\_MARGIN\_TOP, 8);

        $pdf->SetHeaderMargin(PDF\_MARGIN\_HEADER);

        $pdf->SetFooterMargin(PDF\_MARGIN\_FOOTER);

        $pdf->SetAutoPageBreak(TRUE, PDF\_MARGIN\_BOTTOM);

        $pdf->setImageScale(PDF\_IMAGE\_SCALE\_RATIO);

        if (@file\_exists(dirname(\_\_FILE\_\_).'/lang/eng.php')) {

            require\_once(dirname(\_\_FILE\_\_).'/lang/eng.php');

            $pdf->setLanguageArray($l);

        }

        $pdf->SetFont('helvetica', '', 8);

        $pdf->AddPage();

        $pdf->SetFillColor(255, 255, 255);

$pdf->SetFont('helvetica', '', 10);

        // Tabla con los datos del cliente

        $tbl = <<<EOD

        <br>

            <table border="0">

                <tr>

                    <td>

                        <b>Cliente/Client:</b><br>

                        $factura->razon\_social<br>

                        $factura->direccion<br>

                        $factura->colonia<br>

                        $factura->ciudad, $factura->estado<br>

                        $factura->rfc<br>

                    </td>

                    <td>

EOD;

        $tbl .= <<<EOD

        <b>Orden de Compra: </b>

        $factura->orden\_compra<br>

        <b>Contacto / Contact: </b><br>

        $factura->contacto<br>

        $factura->correo<br>

        <br>

        Total de Equipo(s):  $total<br>

        </td>

        </tr>

    </table>

EOD;

        $pdf->writeHTML($tbl, false, false, false, false, '');

        $w = array(8, 125, 12, 24, 24);

        $pdf->SetFont('helvetica', 'B', 9);

        $pdf->SetTextColor(255);

        $pdf->Ln();$pdf->Ln();

        $w = array(20, 20, 125, 20);

        $pdf->SetTextColor(0);

        $pdf->SetFont('helvetica', '', 10);

            $pdf->Ln();

        // Tabla de servicios/equipos

        $tabla\_items='';

            $tabla\_items .= '<table style=" ">

                            <thead>

                                <tr>

                                    <th style="border-bottom: 1px solid #000; text-align: center; font-weight: bold; width: 15%;">Servicio</th>

                                    <th style="border-bottom: 1px solid #000; text-align: center; font-weight: bold; width: 15%;">Equipo ID</th>

                                    <th style="border-bottom: 1px solid #000; text-align: center; font-weight: bold; width: 55%;">Descripcion</th>

                                    <th style="border-bottom: 1px solid #000; text-align: center; font-weight: bold; width: 15%;">Realizado</th>

                                </tr>

                            </thead>

                            <tbody>';

         foreach($rs as $row){

            $date=date\_create($row->Fec\_CalibracionMT);

            $tabla\_items .='

                         <tr>

                            <td style="border-bottom: 1px solid #000; text-align: center; width: 15%; tr:nth-child:background: #F8F8F8;">'.$row->DescripcionDeServicio .'</td>

                            <td style="border-bottom: 1px solid #000; text-align: center; width: 15%; tr:nth-child:background: #F8F8F8;">'.$row->Equipo\_ID.'</td>

                            <td style="border-bottom: 1px solid #000; text-align:center ; width: 55%; tr:nth-child:background: #F8F8F8;">'.$row->CadenaDescripcion.'</td>

                            <td style="border-bottom: 1px solid #000; text-align: center;width: 15%; tr:nth-child:background: #F8F8F8;">'.date\_format($date,'d/M/Y').'</td>

                        </tr>';

             }

             $tabla\_items .= '</tbody>

                </table>

                <br>

                <br>

                <br>

                <br>

                <br>

                <br> ';

        $pdf->writeHTML($tabla\_items, true, false, false, false, '');

        $pdf->Ln();

        $pdf->Ln();

        $pdf->SetFont('helvetica', '', 10);

        $pdf->MultiCell(150, 8,"Recibí el servicio a los equipos arriba listados", 0, 'L', 1, 0, '', '', true, 0, false, true, 0);

        $pdf->SetFont('helvetica', 'B', 10);

        $pdf->writeHTML("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", true, false, false, false, '');

        $pdf->MultiCell(150, 8,"Firma de recibido", 0, 'L', 1, 0, '', '', true, 0, false, true, 0);

        $pdf->Ln();

        $name = "POD ".$id." - PO ".$factura->orden\_compra.'.pdf';

        $pdf->Ln();

$pdf->Output(sys\_get\_temp\_dir().'/'.$name, 'F');

$datos['pod']=sys\_get\_temp\_dir().'/'.$name;

        $datos['id'] = $id;

        $datos['para'] = $para;

        $datos['cc'] = $cc;

        $datos['subject'] = "POD ".$id." - PO ".$factura->orden\_compra;

        $datos['body'] = $body;

        $datos['campos'] = $campos;

        $datos['archivos'] = $archivos;

        $datos['certificados'] = $certificados;

        $datos['pdfname']=$pdfname;

        $datos['xmlname']=$xmlname;

        $datos['po']=$factura->orden\_compra;

        $this->correos\_facturacion->archivos\_facturacion($datos);

    }

    public function validar\_archivos(){

        $id=$this->input->post('ID');

        $factura = $this->Conexion->consultar("SELECT id, folio  from solicitudes\_facturas where id= ".$id, true);

        $query = " select item\_id, documento\_id,Fec\_CalibracionMT from rsitems where factura = ".$factura->folio." and Solicitud\_ID";

        $rs = $this->MLConexion->consultar($query);

        if($rs)

        {

            echo json\_encode($rs);

        }

        else{

            echo "";

        }

    }

}

## Inicio.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Inicio extends CI\_Controller {

    public function index() {

        if(isset($this->session->id)) {

            $id = $this->session->id;

            // Carga de modelos necesarios

            $this->load->model('tickets\_model');

            $this->load->model('agenda\_model');

            $this->load->model('tool\_model');

            $this->load->model('compras\_model');

            $this->load->model('privilegios\_model');

            // Se obtienen tickets pendientes del usuario

            $datos['tickets'] = $this->tickets\_model->getMis\_tickets\_pendientes($this->session->id);

            // Se obtienen juntas agendadas

            $datos['agenda'] = $this->agenda\_model->getjuntas();

            // Pedidos pendientes del módulo "tool"

            $datos['tool'] = $this->tool\_model->pedidosPendientes();

            // Requisiciones QR personales del usuario

            $datos['compras'] = $this->compras\_model->misQRS();

            // Notificaciones del sistema para el usuario

            $datos['noti'] = $this->privilegios\_model->getNotificaciones();

            // Consulta de requisiciones rechazadas del usuario

            $queryQR = "SELECT QR.id, QR.fecha, QR.subtipo, QR.cantidad, QR.descripcion, QR.estatus from requisiciones\_cotizacion QR where QR.usuario = $id";

            $queryQR .= " and QR.estatus = 'RECHAZADO'";

            // Consulta de PRs por recibir del usuario

            $queryPR = "SELECT PR.id, PR.fecha, PR.usuario, PR.prioridad, PR.tipo, PR.subtipo, PR.cantidad, PR.unidad, PR.clave\_unidad, PR.descripcion, PR.atributos, PR.critico, PR.destino, PR.lugar\_entrega, PR.comentarios, PR.estatus, concat(U.nombre, ' ', U.paterno) as User";

            $queryPR .= " from prs PR left join usuarios U on PR.usuario = U.id where 1 = 1 and PR.usuario = $id and PR.estatus = 'POR RECIBIR' order by PR.fecha desc";

            // Consulta de facturas con ciertos estatus (comentada/deshabilitada)

            $queryFact = "SELECT F.id, F.fecha, F.usuario, F.cliente, F.contacto, F.reporte\_servicio, F.orden\_compra, F.forma\_pago, F.pagada, F.conceptos, F.notas, F.estatus, F.estatus\_factura, F.documentos\_requeridos, F.serie, F.folio, F.codigo\_impresion, (SELECT count(id) from recorrido\_facturas where factura = F.id) as Recorridos, E.nombre as Cliente, concat(U.nombre, ' ', U.paterno) as User from solicitudes\_facturas F inner join empresas E on E.id = F.cliente inner join usuarios U on U.id = F.usuario";

            $queryFact .= " where (F.folio > 0 and F.estatus = 'ACEPTADO' and F.estatus\_factura != 'RETORNADA AUTORIZADA') or (F.estatus = 'RECHAZADO')";

            // Resultados de consultas QR y PR

            $datos['qrs'] = $this->Conexion->consultar($queryQR, FALSE, FALSE);

            $datos['prs'] = $this->Conexion->consultar($queryPR, FALSE, FALSE);

            // Desactivado: resultados de facturas

            $datos['facturas'] = "";

        }

        // Carga de vistas principales

        $this->load->view('header');

        $this->load->view('inicio', $datos);

    }

    function primera\_sesion() {

        // Muestra vista para capturar datos en primera sesión

        $this->load->view('inicio/capturar\_datos');

    }

    function confirmar\_datos() {

        // Modelo personalizado para actualizar datos de usuario

        $this->load->model('inicio\_model','Modelo');

        $ACIERTOS = array();

        $ERRORES = array();

        // Datos recibidos del formulario

        $data['correo'] = trim($this->input->post('correo'));

        $data['password'] = sha1($this->input->post('password'));

        $data['activo'] = "1";

        // Validación y actualización de datos

        if($this->Modelo->confirmarDatos($this->session->id, $data)) {

            $this->session->correo = $data['correo'];

            $this->session->password = $data['password'];

            $acierto = array(

                'titulo' => 'Datos Actualizados',

                'detalle' => 'Recuerda que puedes iniciar sesión con tu Numero de Empleado o Correo'

            );

            array\_push($ACIERTOS, $acierto);

        } else {

            $error = array(

                'titulo' => 'ERROR',

                'detalle' => 'Error al actualizar Datos'

            );

            array\_push($ERRORES, $error);

        }

        // Almacenar mensajes de éxito o error en sesión

        $this->session->aciertos = $ACIERTOS;

        $this->session->errores = $ERRORES;

        // Redirección al inicio

        redirect(base\_url('inicio'));

    }

}

## Login.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Login extends CI\_Controller {

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('usuarios\_model');

        $this->load->library('correos\_tickets');

    }

    function index() {

        date\_default\_timezone\_set('America/Chihuahua');

        $data['url\_actual'] = base\_url('inicio');

        // Si existe una URL previa en sesión, se conserva para redireccionar luego

        if(isset($this->session->url\_actual)) {

            $data['url\_actual'] = $this->session->url\_actual;

        }

        // Destruye la sesión existente

        $this->session->sess\_destroy();

        // Carga vista de login

        $this->load->view('login', $data);

    }

    function autenticar() {

        $url\_actual = $this->input->post('url\_actual');

        $usuario = $this->input->post('user');

        $pass = $this->input->post('pass');

        // Intenta autenticar al usuario con los datos proporcionados

        $res = $this->usuarios\_model->autenticar($usuario, $pass);

        if ($res) {

            // Datos del usuario autenticado

            $row = $res->row();

            // Guardar datos importantes en la sesión

            $this->session->nombre = $row->User;

            $this->session->id = $row->id;

            $this->session->no\_empleado = $row->no\_empleado;

            $this->session->password = $row->password;

            $this->session->vencimiento\_password = $row->vencimiento\_password;

            $this->session->password\_correo = $row->password\_correo;

            $this->session->correo = $row->correo;

            $this->session->puesto = $row->puesto;

            $this->session->activo = $row->activo;

            $this->session->foto = $row->foto;

            $this->session->chats = '{ "chatbox0" : "CHAT"}';

            $this->session->departamento = $row->departamento;

            // Se asignan privilegios del usuario

            $this->session->privilegios = $this->usuarios\_model->getPrivilegios($this->session->id);

            // Se actualiza última sesión

            $this->usuarios\_model->ultimaSesion($row->id);

            // Se valida si hay tickets abiertos o en curso del usuario

            $us = $this->Conexion->consultar("SELECT max(ultima\_sesion) as us from usuarios", true);

            $fecha = date("y-m-d", strtotime($us->us));

            $today = date("y-m-d", strtotime("today"));

            $query = "SELECT 'IT' as tipo, T.id, T.usuario, concat(U.nombre,' ',U.paterno) as User, U.correo, T.estatus from tickets\_sistemas T inner join usuarios U on U.id = T.usuario where T.estatus = 'ABIERTO' and usuario = $row->id"

                . " union "

                . "SELECT 'IT' as tipo, T.id, T.usuario, concat(U.nombre,' ',U.paterno) as User, U.correo, T.estatus from tickets\_sistemas T inner join usuarios U on U.id = T.usuario where T.estatus = 'EN CURSO' and usuario = $row->id"

                . " union "

                . "SELECT 'AT', T.id, T.usuario, concat(U.nombre,' ',U.paterno) as User, U.correo, T.estatus from tickets\_autos T inner join usuarios U on U.id = T.usuario where T.estatus = 'ABIERTO' and usuario = $row->id"

                . " union "

                . "SELECT 'AT', T.id, T.usuario, concat(U.nombre,' ',U.paterno) as User, U.correo, T.estatus from tickets\_autos T inner join usuarios U on U.id = T.usuario where T.estatus = 'EN CURSO' and usuario = $row->id"

                . " union "

                . "SELECT 'ED', T.id, T.usuario, concat(U.nombre,' ',U.paterno) as User, U.correo, T.estatus from tickets\_edificio T inner join usuarios U on U.id = T.usuario where T.estatus = 'ABIERTO' and usuario = $row->id"

                . " union "

                . "SELECT 'ED', T.id, T.usuario, concat(U.nombre,' ',U.paterno) as User, U.correo, T.estatus from tickets\_edificio T inner join usuarios U on U.id = T.usuario where T.estatus = 'EN CURSO' and usuario = $row->id";

            $arr = array();

            $res = $this->Conexion->consultar($query);

            if($res) {

                foreach ($res as $key => $value) {

                    if(!isset($arr[$value->usuario])) {

                        $arr[$value->usuario]['nombre'] = $value->User;

                        $arr[$value->usuario]['correo'] = $value->correo;

                        $arr[$value->usuario]['tickets'] = array();

                    }

                    array\_push($arr[$value->usuario]['tickets'], $value->tipo . str\_pad($value->id, 6, "0", STR\_PAD\_LEFT));

                }

            }

            // Posible envío de correo (actualmente desactivado)

            // $this->correos\_tickets->ticketsPendientes($arr);

            // Redirige a la URL original

            redirect($url\_actual);

        } else {

            // Autenticación fallida, se guarda mensaje de error en sesión

            $ERRORES = array();

            $error = array('titulo' => 'ERROR', 'detalle' => 'Usuario y/o Contraseña incorrectas');

            array\_push($ERRORES, $error);

            $this->session->errores = $ERRORES;

            // Se conserva URL actual para reintentar

            $this->session->url\_actual = $url\_actual;

            // Redirige de nuevo a login

            redirect(base\_url('login'));

        }

    }

    public function cerrar\_sesion() {

        // Destruye la sesión y redirige a login

        $this->session->sess\_destroy();

        redirect(base\_url('login'));

    }

}

## Logistica.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Logistica extends CI\_Controller {

    function \_\_construct() {

    parent::\_\_construct();

    // Carga la librería personalizada para envío de correos de logística

    $this->load->library('correos\_logistica');

    // Carga el modelo de conexión a base de datos para logística

    $this->load->model('MLConexion\_model', 'MLConexion');

}

function index() {

    // Vista principal de opciones de logística

    $this->load->view('header');

    $this->load->view('logistica/logistica\_opciones');

}

function documentacion() {

    // Vista para el módulo de documentación logística

    $this->load->view('header');

    $this->load->view('logistica/documentacion');

}

function equipos() {

    // Vista del módulo de equipos logísticos

    $this->load->view('header');

    $this->load->view('logistica/equipos');

}

function programacion\_recorridos() {

    // Vista para programar recorridos logísticos

    $this->load->view('header');

    $this->load->view('logistica/programacion\_recorridos');

}

function recorridos() {

    // Vista para ver y gestionar recorridos

    $this->load->view('header');

    $this->load->view('logistica/recorridos');

}

function dashboard() {

    // Vista del panel de control logístico

    $this->load->view('header');

    $this->load->view('logistica/dashboard');

}

function firmar() {

    // Muestra formulario o vista para firmar documentos/logística

    $data['data'] = $this->input->post();

    $this->load->view('logistica/firmar', $data);

}

function formato\_requisitos() {

    // Decodifica lista de IDs de empresas enviada vía POST

    $empresas = json\_decode($this->input->post('empresas'));

    $data = [];

    // Consulta datos de cada empresa y los agrega a un arreglo

    foreach ($empresas as $key => $value) {

        $res = $this->Conexion->consultar("SELECT \* from empresas where id = $value", TRUE);

        $empresa = new stdClass;

        $empresa->nombre = $res->nombre;

        $empresa->requisitos\_logisticos = $res->requisitos\_logisticos;

        $empresa->requisitos\_documento = $res->requisitos\_documento;

        array\_push($data, $empresa);

    }

    // Se ocultan errores PHP

    ini\_set('display\_errors', 0);

    // Carga librería para generar PDF

    $this->load->library('pdfview');

    $pdf = new TCPDF(PDF\_PAGE\_ORIENTATION, PDF\_UNIT, PDF\_PAGE\_FORMAT, true, 'UTF-8', false);

    // Configuración del documento PDF

    $pdf->SetCreator(PDF\_CREATOR);

    $pdf->SetAuthor('AleksOrtiz');

    $pdf->SetTitle('Masmetrologia');

    $pdf->SetSubject('Formato Requisitos');

    $pdf->SetHeaderData(PDF\_HEADER\_LOGO\_ORIGINAL, '40', '                         Requisitos de Empresa');

    $pdf->setHeaderFont(Array(PDF\_FONT\_NAME\_MAIN, '', 10));

    $pdf->setFooterFont(Array(PDF\_FONT\_NAME\_DATA, '', PDF\_FONT\_SIZE\_DATA));

    $pdf->SetDefaultMonospacedFont(PDF\_FONT\_MONOSPACED);

    $pdf->SetMargins(PDF\_MARGIN\_LEFT, PDF\_MARGIN\_TOP, PDF\_MARGIN\_RIGHT);

    $pdf->SetHeaderMargin(PDF\_MARGIN\_HEADER);

    $pdf->SetFooterMargin(PDF\_MARGIN\_FOOTER);

    $pdf->SetAutoPageBreak(TRUE, PDF\_MARGIN\_BOTTOM);

    $pdf->setImageScale(PDF\_IMAGE\_SCALE\_RATIO);

    $pdf->SetFont('times', 'B', 15);

    $pdf->AddPage();

    $pdf->SetFillColor(255, 255, 255);

    // Escribe la información de cada empresa en el PDF

    foreach ($data as $key => $empresa) {

        $pdf->SetFont('times', 'B', 16);

        $pdf->Write(0, $empresa->nombre, '', 0, 'L', true, 0, false, false, 0);

        $pdf->SetFont('times', 'B', 10);

        $pdf->Write(0, 'Requisitos Logísticos:', '', 0, 'L', true, 0, false, false, 0);

        $pdf->SetFont('times', '', 10);

        $pdf->Write(0, $empresa->requisitos\_logisticos, '', 0, 'L', true, 0, false, false, 0);

        $pdf->ln();

        $pdf->SetFont('times', 'B', 10);

        $pdf->Write(0, 'Requisitos de Documento:', '', 0, 'L', true, 0, false, false, 0);

        $pdf->SetFont('times', '', 10);

        $pdf->Write(0, $empresa->requisitos\_documento, '', 0, 'L', true, 0, false, false, 0);

        $pdf->ln();

        $pdf->ln();

    }

    // Muestra el PDF en pantalla

    $pdf->Output('Requisitos.pdf', 'I');

}

    ////////////////////////////////// A J A X //////////////////////////////////

    function ajax\_getSolicitudes(){

    // Consulta base: solicitudes de factura con empresa, usuario y conteo de recorridos

    $query = "SELECT F.id, F.fecha, F.fecha\_retorno, F.usuario, F.cliente, F.contacto, F.reporte\_servicio, F.orden\_compra, F.forma\_pago, F.pagada, F.conceptos, F.notas, F.estatus\_factura, F.documentos\_requeridos, F.serie, F.folio, F.codigo\_impresion, E.nombre as Cliente, concat(U.nombre, ' ', U.paterno) as User, ";

    $query .= "(SELECT count(id) from recorrido\_conceptos where id\_concepto = F.id and tipo = 'FACTURA') as Recorridos, (SELECT count(id) from recorrido\_conceptos where id\_concepto = F.id and tipo = 'FACTURA' and cerrado = 1) as RecorridosCerrados from solicitudes\_facturas F inner join empresas E on E.id = F.cliente inner join usuarios U on U.id = F.usuario where 1 = 1";

    if(isset($\_POST['estatus\_factura']))

    {

        $estatus = $this->input->post('estatus\_factura');

        // Filtra solicitudes por estatus de entrega si corresponde

        if($estatus == "ENTREGA")

        {

            $query .= " and (F.estatus\_factura = 'RECIBIDO EN LOGISTICA' or F.estatus\_factura = 'NO ENTREGADA' or F.estatus\_factura = 'ENTREGA RECHAZADA') having Recorridos = RecorridosCerrados";

        }

        // Filtra solicitudes por estatus de recolección si corresponde

        else if($estatus == "RECOLECTA")

        {

            $query .= " and (F.estatus\_factura = 'DEJADA CON CLIENTE' or F.estatus\_factura = 'NO RECOLECTADA' or F.estatus\_factura = 'RECOLECTA RECHAZADA') having Recorridos = RecorridosCerrados";

            $query .= " order by F.fecha\_retorno asc";

        }

        // Filtra por otro estatus específico

        else

        {

            $query .= " and F.estatus\_factura = '$estatus'";

        }

    }

    // Ejecuta la consulta y devuelve resultados en JSON

    $res = $this->Conexion->consultar($query);

    if($res)

    {

        echo json\_encode($res);

    }

}

    ////////////////////////////////// L O G I S T I C A //////////////////////////////////

   function ajax\_getMLEquipos() {

    // Consulta equipos en proceso de entrega desde la vista v\_items\_entregando

    $res = $this->MLConexion->consultar("SELECT \*, ifnull(Item,'') as Equipo\_ID, ifnull(fabricante,'') as Fabricante, ifnull(modelo,'') as Modelo, ifnull(serie,'') as Serie from v\_items\_entregando");

    if($res) {

        echo json\_encode($res);

    }

}

function ajax\_getMensajeros() {

    // Consulta usuarios activos, potencialmente filtrando solo mensajeros

    $query = "SELECT U.id, U.nombre, U.paterno, U.materno, U.no\_empleado, U.puesto, U.correo, U.ultima\_sesion, U.departamento, U.activo, U.jefe\_directo, U.autorizador\_compras, U.autorizador\_compras\_venta, concat(U.nombre, ' ', U.paterno) as User, concat(U.nombre, ' ', U.paterno, ' ', U.materno) as CompleteName from usuarios U inner join privilegios P on P.usuario = U.id where U.activo = '1'";

    if(isset($\_POST['mensajeros']) && $\_POST['mensajeros'] == "1") {

        $query .= " and P.mensajero = '1'";

    }

    $query .= " order by User";

    $res = $this->Conexion->consultar($query);

    if($res) {

        echo json\_encode($res);

    }

}

function ajax\_setRecorrido() {

    // Obtiene datos del formulario

    $mensajero = $this->input->post('mensajero');

    $fecha = $this->input->post('fecha');

    $recorrido = json\_decode($this->input->post('recorrido'));

    // Crea nuevo recorrido principal

    $data['mensajero'] = $mensajero;

    $data['fecha\_recorrido'] = $fecha;

    $data['estatus'] = 'ASIGNADO A MENSAJERO';

    $recorrido\_id = $this->Conexion->insertar('recorridos', $data);

    // Inserta cada concepto dentro del recorrido

    foreach ($recorrido as $value) {

        $data2['recorrido'] = $recorrido\_id;

        $data2['tipo'] = $value[0];

        $data2['id\_concepto'] = $value[1];

        $data2['rs'] = $value[2];

        $data2['cliente'] = $value[3];

        $data2['descripcion'] = $value[4];

        $data2['accion'] = $value[5];

        $data2['estatus'] = "ASIGNADO A MENSAJERO";

        $data2['discrepancia'] = "0";

        $data2['cerrado'] = "0";

        $data2['reporte'] = "0";

        $this->Conexion->insertar('recorrido\_conceptos', $data2);

        // Actualiza estatus en la tabla solicitudes\_facturas (si corresponde)

        $this->Conexion->modificar('solicitudes\_facturas', array('estatus\_factura' => $data2['estatus']), null, array('id' => $data2['factura']));

    }

}

    function ajax\_getRecorridos() {

    $pendientes = $this->input->post('pendientes');

    $factura = $this->input->post('factura');

    // Consulta de recorridos asociados a facturas con datos de usuario, cliente y estado

    $query = "SELECT RF.\*, SF.folio, SF.cliente, R.mensajero, R.fecha\_recorrido, R.estatus as Estatus\_recorrido, ifnull(RR.id,'N/A') as Reporte, E.nombre as Cliente,";

    $query .= "(SELECT count(RF2.id) from recorrido\_facturas RF2 where RF2.recorrido = RF.recorrido and (RF2.estatus = 'ASIGNADO A MENSAJERO' or RF2.estatus = 'EN RECORRIDO')) as Pendientes,";

    $query .= "ifnull(concat(M.nombre, ' ', M.paterno), 'N/A') as Mensajero";

    $query .= " from recorrido\_facturas RF";

    $query .= " inner join recorridos R on R.id = RF.recorrido";

    $query .= " left join recorrido\_reporte RR on RR.id = RF.reporte";

    $query .= " inner join solicitudes\_facturas SF on SF.id = RF.factura";

    $query .= " inner join empresas E on E.id = SF.cliente";

    $query .= " left join usuarios M on M.id = R.mensajero where 1 = 1";

    // Filtro por factura específica

    if(isset($\_POST['factura'])) {

        $query .= " and RF.factura = $factura";

    }

    // Filtro por recorridos pendientes

    if($pendientes == "1") {

        $query .= " and (R.estatus = 'ACEPTADO' or R.estatus = 'ASIGNADO A MENSAJERO')";

    }

    $query .= " order by R.fecha\_recorrido, R.id, SF.cliente, RF.accion, SF.folio, RF.id asc";

    $res = $this->Conexion->consultar($query);

    // Devuelve resultados como JSON

    echo json\_encode($res);

}

function ajax\_getEnvios() {

    $factura = $this->input->post('factura');

    // Consulta de envíos asociados a una factura específica

    $query = "SELECT EF.\*, concat(U.nombre, ' ', U.paterno) as User from envios\_factura EF inner join usuarios U on U.id = EF.usuario where EF.factura = $factura";

    $res = $this->Conexion->consultar($query);

    if($res) {

        echo json\_encode($res);

    }

}

    function ajax\_setEstatusEnvioCorreo() {

    // Actualiza el estatus de un envío en la tabla envios\_factura

    $where['id'] = $this->input->post('id\_envio');

    $data['estatus'] = $this->input->post('estatus');

    $this->Conexion->modificar('envios\_factura', $data, null, $where);

}

function ajax\_setEnviosComentarios() {

    // Inserta un comentario asociado a un envío

    $data['envio'] = $this->input->post('id\_envio');

    $data['comentario'] = $this->input->post('comentario');

    $data['usuario'] = $this->session->id;

    $func['fecha'] = 'CURRENT\_TIMESTAMP()';

    $this->Conexion->insertar('envios\_factura\_comentarios', $data, $func);

}

function ajax\_getEnviosComentarios() {

    // Consulta comentarios relacionados a un envío

    $id\_envio = $this->input->post('id\_envio');

    $query = "SELECT EFC.\*, concat(U.nombre, ' ', U.paterno) as User from envios\_factura\_comentarios EFC inner join usuarios U on U.id = EFC.usuario where EFC.envio = $id\_envio";

    $res = $this->Conexion->consultar($query);

    if($res) {

        echo json\_encode($res);

    }

}

function ajax\_getFacturasRecorrido() {

    // Consulta facturas que forman parte de un recorrido específico

    $id = $this->input->post('id');

    $query = "SELECT RF.\*, SF.folio from recorrido\_facturas RF inner join solicitudes\_facturas SF on SF.id = RF.factura where RF.recorrido = $id";

    $res = $this->Conexion->consultar($query);

    echo json\_encode($res);

}

    function ajax\_aceptarRecorrido() {

    // Cambia estatus del recorrido a "ACEPTADO"

    $id = $this->input->post('recorrido');

    $data["estatus"] = "ACEPTADO";

    $res = $this->Conexion->modificar('recorridos', $data, null, array('id' => $id));

    // Actualiza estatus de recorrido\_facturas y solicitudes\_facturas a "EN RECORRIDO"

    $this->Conexion->comando("UPDATE solicitudes\_facturas SF, recorrido\_facturas RF set SF.estatus\_factura = 'EN RECORRIDO', RF.estatus = 'EN RECORRIDO' where SF.id = RF.factura and RF.recorrido = $id");

    if($res > 0) {

        echo "1";

    }

}

function ajax\_rechazarRecorrido() {

    // Cambia estatus del recorrido a "RECHAZADO POR MENSAJERO"

    $id = $this->input->post('recorrido');

    $acc = $this->input->post('accion');

    $data["estatus"] = "RECHAZADO POR MENSAJERO";

    $res = $this->Conexion->modificar('recorridos', $data, null, array('id' => $id));

    // Actualiza estatus de recorrido\_facturas y solicitudes\_facturas a "{ACCION} RECHAZADA"

    $this->Conexion->comando("UPDATE solicitudes\_facturas SF, recorrido\_facturas RF set SF.estatus\_factura = '$acc RECHAZADA', RF.estatus = 'RECHAZADO POR MENSAJERO' where SF.id = RF.factura and RF.recorrido = $id");

    if($res > 0) {

        echo "1";

    }

}

function ajax\_pendienteCierreRecorrido() {

    // Marca recorrido como "PENDIENTE CIERRE"

    $id = $this->input->post('recorrido');

    $data["estatus"] = "PENDIENTE CIERRE";

    $res = $this->Conexion->modificar('recorridos', $data, null, array('id' => $id));

    if($res > 0) {

        echo "1";

    }

}

function ajax\_getRecorridosPendienteCierre() {

    // Obtiene recorridos que están pendientes por cerrar

    $res = $this->Conexion->consultar('SELECT R.\*, CONCAT(M.nombre, " ", M.paterno) as Mensajero from recorridos R inner join usuarios M on M.id = R.mensajero where R.estatus = "PENDIENTE CIERRE"');

    if($res) {

        echo json\_encode($res);

    }

}

    function guardarFirma() {

    ///////////////// F I R M A  J P G /////////////////

    // Se obtiene la firma en base64 desde POST

    $file = $this->input->post('firma');

    // Decodifica la imagen base64 a binario

    $image = base64\_decode(str\_replace('data:image/png;base64,', '', $file));

    // Define el nombre del archivo y la ruta de guardado

    $image\_name = 'HOLA';

    $filename = $image\_name . '.' . 'jpg';

    $path = "data/logistica/firmas/" . $filename;

    // Guarda la imagen en disco

    file\_put\_contents($path, $image);

    // Retorna el contenido recibido

    echo $file;

    return;

        if($file != "undefined")

        {

            $config['upload\_path'] = 'data/logistica/firmas/';

            $config['allowed\_types'] = 'gif|jpg|png';

            $config['encrypt\_name'] = TRUE;

            $this->load->library('upload', $config);

            if ($this->upload->do\_upload('iptFoto'))

            {

                $data['id'] = $id;

                $data['foto'] = $this->upload->data('file\_name');

                if($this->Modelo->update($data)){

                    if($foto != "default.png"){

                        unlink('data/empresas/fotos/' . $foto );

                    }

                    echo $data['foto'];

                }

            } else {

                echo "";

            }

        } else {

            $data['id'] = $id;

            $data['foto'] = 'default.png';

            if($this->Modelo->update($data)){

                if($foto != "default.png"){

                    unlink('data/empresas/fotos/' . $foto );

                }

                echo $data['foto'];

            }

        }

        ///////////////////////////////////////////////////////////////

    }

    function ajax\_updateRecorrido() {

    $recorrido = json\_decode($this->input->post('recorrido'));

    $facturas = json\_decode($this->input->post('facturas'));

    $recolecta = $this->input->post('recolecta');

    $comentario = $this->input->post('comentario');

    $fecha = $this->input->post('fecha');

    // Si la acción fue completada y se requiere recolección, cambia estatus y fecha

    if($recorrido->estatus == "ENTREGADA") {

        if($recolecta == "1" && $recorrido->accion == "ENTREGA") {

            $recorrido->estatus = "DEJADA CON CLIENTE";

            $data["fecha\_retorno"] = $fecha;

        } else {

            $recorrido->estatus = "RETORNADA AUTORIZADA";

        }

    }

    // Se genera reporte del recorrido

    $recorrido\_reporte['recorrido'] = $recorrido->id;

    $recorrido\_reporte['cliente'] = $recorrido->cliente;

    $recorrido\_reporte['contacto'] = $recorrido->contacto;

    $recorrido\_reporte['accion'] = $recorrido->accion;

    $recorrido\_reporte['resultado'] = $recorrido->estatus;

    $recorrido\_reporte['firma'] = isset($\_POST['firma']) ? 1 : 0;

    $RR = $this->Conexion->insertar('recorrido\_reporte', $recorrido\_reporte, array('fecha' => 'CURRENT\_TIMESTAMP()'));

    // Se registran facturas del recorrido y se actualiza su estatus

    $folios = "";

    foreach ($facturas as $value) {

        $folios .= $value->folio . ", ";

        $recorrido\_facturas['recorrido'] = $recorrido->id;

        $recorrido\_facturas['factura'] = $value->factura;

        $recorrido\_facturas['accion'] = $recorrido->accion;

        $recorrido\_facturas['estatus'] = $recorrido->estatus;

        $recorrido\_facturas['reporte'] = $RR;

        if($value->nueva == "1") {

            $this->Conexion->insertar('recorrido\_facturas', $recorrido\_facturas);

        } else {

            $this->Conexion->modificar('recorrido\_facturas', $recorrido\_facturas, null, array('id' => $value->id));

        }

        $data['estatus\_factura'] = $recorrido->estatus;

        $this->Conexion->modificar('solicitudes\_facturas', $data, null, array('id' => $value->factura));

    }

    // Si hay comentario, se registra con color según el estatus

    if($comentario) {

        $color = substr($recorrido->estatus, 0, 2) == "NO" ? "red" : "green";

        $comentario = '<font color=' . $color . '><b>' . $recorrido->estatus . ':</b></font> ' . $comentario;

        $data\_com['reporte'] = $RR;

        $data\_com['usuario'] = $this->session->id;

        $data\_com['comentario'] = $comentario;

        $func\_com['fecha'] = "CURRENT\_TIMESTAMP()";

        $this->Conexion->insertar('recorrido\_comentarios', $data\_com, $func\_com);

    }

    // Si se capturó firma, se guarda como imagen y se envía correo si aplica

    if(isset($\_POST['firma'])) {

        $file = $this->input->post('firma');

        $image = base64\_decode(str\_replace('data:image/png;base64,', '', $file));

        $image\_name = $RR . '.jpg';

        $path = "data/logistica/firmas/" . $image\_name;

        file\_put\_contents($path, $image);

        // Enviar correo de acuse si se proporcionó correo de contacto

        if($recorrido->CorreoContacto) {

            $data = new stdClass;

            $data->facturas = rtrim($folios, ', ');

            $data->contacto = $recorrido->NombreContacto;

            $data->correo = $recorrido->CorreoContacto;

            $data->comentarios = $comentario;

            $data->firma = $image\_name;

            $data->fecha\_retorno = $fecha;

            $this->correos\_logistica->acuse($data);

        }

    }

}

    function ajax\_getComentariosRecorrido() {

    $id = $this->input->post('id');

    // Consulta comentarios del recorrido con nombre del usuario

    $query = "SELECT C.\*, concat(U.nombre, ' ', U.paterno) as User from recorrido\_comentarios C inner join usuarios U on U.id = C.usuario where 1 = 1";

    if($id) {

        $query .= " and C.reporte = '$id'";

    }

    $query .= " order by C.fecha";

    $res = $this->Conexion->consultar($query);

    if($res) {

        echo json\_encode($res);

    }

}

function ajax\_getReporte() {

    $idReporte = $this->input->post('id');

    // Consulta detalles del reporte de recorrido con información de factura, cliente, contacto y requisitor

    $query = "SELECT RF.id, RR.fecha, RF.discrepancia, RR.accion, RR.resultado, SF.folio, RF.estatus, E.nombre as Cliente, ifnull(EC.nombre, 'N/A') as Contacto, RR.firma, concat(U.nombre, ' ', U.paterno) as Requisitor FROM recorrido\_facturas RF";

    $query .= " inner join solicitudes\_facturas SF on SF.id = RF.factura";

    $query .= " inner join usuarios U on U.id = SF.usuario";

    $query .= " inner join recorrido\_reporte RR on RR.id = RF.reporte";

    $query .= " inner join empresas E on E.id = RR.cliente";

    $query .= " left join empresas\_contactos EC on EC.id = RR.contacto";

    $query .= " where RF.reporte = $idReporte";

    $res = $this->Conexion->Consultar($query);

    if($res) {

        echo json\_encode($res);

    }

}

function ajax\_aceptarCierreRecorrido() {

    // Marca recorrido como cerrado y actualiza sus conceptos

    $recorrido = json\_decode($this->input->post('recorrido'));

    $this->Conexion->modificar('recorridos', $recorrido, null, array('id' => $recorrido->id));

    $this->Conexion->modificar('recorrido\_facturas', array('cerrado' => 1), null, array('recorrido' => $recorrido->id));

}

function ajax\_rechazarCierreRecorrido() {

    // Marca recorrido como cerrado pero identifica discrepancias en facturas específicas

    $recorrido = json\_decode($this->input->post('recorrido'));

    $facturas = json\_decode($this->input->post('facturas'));

    $this->Conexion->modificar('recorridos', $recorrido, null, array('id' => $recorrido->id));

    $this->Conexion->modificar('recorrido\_facturas', array('cerrado' => 1), null, array('recorrido' => $recorrido->id));

    foreach ($facturas as $id\_factura) {

        $this->Conexion->comando("UPDATE recorrido\_facturas RF, solicitudes\_facturas SF set RF.discrepancia = 1, RF.estatus = concat('NO ', RF.accion, 'DA'), SF.estatus\_factura = concat('NO ', RF.accion, 'DA') where SF.id = RF.factura and RF.id = $id\_factura");

    }

}

function ajax\_getEmpresasRecorrido() {

    // Consulta empresas involucradas en un recorrido

    $recorrido = $this->input->post('recorrido');

    $query = "SELECT E.id, E.nombre from empresas E inner join solicitudes\_facturas SF on SF.cliente = E.id inner join recorrido\_facturas RF on RF.factura = SF.id where RF.recorrido = $recorrido order by E.nombre";

    $res = $this->Conexion->consultar($query);

    echo json\_encode($res);

}

    function ajax\_getDashboard() {

    // Consulta resumen de estatus y fechas mínimas por categoría para el dashboard logístico

    $query = 'SELECT count(\*) as Total, (SELECT count(\*) FROM solicitudes\_facturas where estatus\_factura = "ENVIADA LOGISTICA") as D1, (SELECT min(fecha) FROM solicitudes\_facturas where estatus\_factura = "ENVIADA LOGISTICA") as ult1,';

    $query .= ' (SELECT count(\*) FROM solicitudes\_facturas where estatus\_factura = "RECIBIDO EN LOGISTICA") as D2, (SELECT min(fecha) FROM solicitudes\_facturas where estatus\_factura = "RECIBIDO EN LOGISTICA") as ult2,';

    $query .= ' (SELECT count(\*) FROM solicitudes\_facturas where estatus\_factura = "RECHAZADO EN LOGISTICA") as D3, (SELECT min(fecha) FROM solicitudes\_facturas where estatus\_factura = "RECHAZADO EN LOGISTICA") as ult3,';

    $query .= ' (SELECT count(\*) FROM solicitudes\_facturas where estatus\_factura = "ASIGNADO A MENSAJERO") as D4, (SELECT min(fecha) FROM solicitudes\_facturas where estatus\_factura = "ASIGNADO A MENSAJERO") as ult4,';

    $query .= ' (SELECT count(\*) FROM solicitudes\_facturas where estatus\_factura = "ENTREGA RECHAZADA") as D5, (SELECT min(fecha) FROM solicitudes\_facturas where estatus\_factura = "ENTREGA RECHAZADA") as ult5,';

    $query .= ' (SELECT count(\*) FROM solicitudes\_facturas where estatus\_factura = "RECOLECTA RECHAZADA") as D6, (SELECT min(fecha) FROM solicitudes\_facturas where estatus\_factura = "RECOLECTA RECHAZADA") as ult6,';

    $query .= ' (SELECT count(\*) FROM solicitudes\_facturas where estatus\_factura = "EN RECORRIDO") as D7, (SELECT min(fecha) FROM solicitudes\_facturas where estatus\_factura = "EN RECORRIDO") as ult7,';

    $query .= ' (SELECT count(\*) FROM solicitudes\_facturas where estatus\_factura = "DEJADA CON CLIENTE") as D8, (SELECT min(fecha) FROM solicitudes\_facturas where estatus\_factura = "DEJADA CON CLIENTE") as ult8,';

    $query .= ' (SELECT count(\*) FROM solicitudes\_facturas where estatus\_factura = "NO ENTREGADA") as D9, (SELECT min(fecha) FROM solicitudes\_facturas where estatus\_factura = "NO ENTREGADA") as ult9,';

    $query .= ' (SELECT count(\*) FROM solicitudes\_facturas where estatus\_factura = "NO RECOLECTADA") as D10, (SELECT min(fecha) FROM solicitudes\_facturas where estatus\_factura = "NO RECOLECTADA") as ult10';

    $query .= ' FROM solicitudes\_facturas;';

    $res = $this->Conexion->consultar($query, TRUE);

    echo json\_encode($res);

}

function ajax\_setSolicitud() {

    $solicitud = json\_decode($this->input->post('solicitud'));

    $other = json\_decode($this->input->post('other'));

    // Adjunta archivos binarios si existen

    if(isset($\_FILES['f\_O'])) {

        $solicitud->f\_orden\_compra = file\_get\_contents($\_FILES['f\_O']['tmp\_name']);

    }

    if(isset($\_FILES['f\_R'])) {

        $solicitud->f\_remision = file\_get\_contents($\_FILES['f\_R']['tmp\_name']);

    }

    $funciones = array('fecha' => 'CURRENT\_TIMESTAMP()');

    $res = false;

    // Inserta nueva solicitud o actualiza una existente

    if($solicitud->id == 0) {

        $res = $this->Conexion->insertar('solicitudes\_facturas', $solicitud, $funciones);

        $solicitud->id = $res;

    } else {

        $res = $this->Conexion->modificar('solicitudes\_facturas', $solicitud, null, array('id' => $solicitud->id)) >= 0;

    }

    // Si se cargó archivo "other", se notifica vía correo

    if(isset($\_FILES['other'])) {

        $solicitud->User = $other->User;

        $solicitud->Client = $other->Client;

        $solicitud->Contact = $other->Contact;

        $correos = [];

        $correos\_a = $this->Conexion->consultar("SELECT U.correo from privilegios P inner join usuarios U on P.usuario = U.id where P.responder\_facturas = 1");

        foreach ($correos\_a as $key => $value) {

            array\_push($correos, $value->correo);

        }

        $solicitud->correos = array\_merge(array($this->session->correo), $correos);

        // Envía correo de notificación

        $this->correos\_facturacion->solicitud($solicitud);

    }

    if($res) {

        echo "1";

    }

}

    function ajax\_editSolicitud() {

    $solicitud = json\_decode($this->input->post('solicitud'));

    $other = json\_decode($this->input->post('other'));

    // Se cargan archivos si se enviaron en el formulario

    if(isset($\_FILES['f\_A'])) {

        $solicitud->f\_acuse = file\_get\_contents($\_FILES['f\_A']['tmp\_name']);

    }

    if(isset($\_FILES['f\_F'])) {

        $solicitud->f\_factura = file\_get\_contents($\_FILES['f\_F']['tmp\_name']);

    }

    if(isset($\_FILES['f\_X'])) {

        $solicitud->f\_xml = file\_get\_contents($\_FILES['f\_X']['tmp\_name']);

    }

    $comentario = $this->input->post('comentario');

    // Actualiza la solicitud

    $res = $this->Conexion->modificar('solicitudes\_facturas', $solicitud, null, array('id' => $solicitud->id));

    if($res > 0) {

        // Si se envió comentario, lo guarda

        if(isset($\_POST['comentario']) && !empty($comentario)) {

            $this->Conexion->insertar('solicitudes\_facturas\_comentarios', array('solicitud' => $solicitud->id, 'usuario' => $this->session->id, 'comentario' => $comentario), array('fecha' => 'CURRENT\_TIMESTAMP()'));

            $solicitud->comentario = $comentario;

        }

        // Prepara correos de notificación

        $correos = [];

        $correos\_a = $this->Conexion->consultar("SELECT U.correo from privilegios P inner join usuarios U on P.usuario = U.id where P.responder\_facturas = 1");

        foreach ($correos\_a as $key => $value) {

            array\_push($correos, $value->correo);

        }

        $solicitud->correos = array\_merge(array($this->session->correo), $correos);

        $solicitud->User = $other->User;

        $solicitud->Client = $other->Client;

        $solicitud->Contact = $other->Contact;

        // Envía correo de modificación de solicitud

        $this->correos\_facturacion->editar\_solicitud($solicitud);

        echo "1";

    } else {

        echo "";

    }

}

function ajax\_setComentarios() {

    $comentario = json\_decode($this->input->post('comentario'));

    // Asigna el usuario de sesión al comentario y registra la fecha actual

    $comentario->usuario = $this->session->id;

    $funciones = array('fecha' => 'CURRENT\_TIMESTAMP()');

    // Inserta el comentario en la base de datos

    $res = $this->Conexion->insertar('solicitudes\_facturas\_comentarios', $comentario, $funciones);

    if($res > 0) {

        echo "1";

    }

}

    function ajax\_getComentarios() {

    $id = $this->input->post('id');

    // Consulta comentarios asociados a una solicitud específica

    $query = "SELECT C.\*, concat(U.nombre, ' ', U.paterno) as User from solicitudes\_facturas\_comentarios C inner join usuarios U on U.id = C.usuario where 1 = 1";

    if($id) {

        $query .= " and C.solicitud = '$id'";

    }

    $res = $this->Conexion->consultar($query);

    if($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

function ajax\_getRequisitores() {

    $id = $this->input->post('id');

    // Consulta de usuarios con privilegio de solicitar facturas

    $query = "SELECT U.id, concat(U.nombre, ' ', U.paterno) as Nombre, P.puesto as Puesto from usuarios U inner join puestos P on U.puesto = P.id inner join privilegios PR on PR.usuario = U.id where U.activo = 1 and PR.solicitar\_facturas = 1";

    if($id) {

        $query .= " and U.id = '$id'";

    }

    $res = $this->Conexion->consultar($query, $id);

    if($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

function ajax\_getVFPData() {

    // Ejecuta lector de datos VFP externo pasando el modelo como argumento

    $modelo = $this->input->post('modelo');

    $res = shell\_exec("C:/xampp/htdocs/MASMetrologia/vfp\_reader/vfp\_reader.exe \"$modelo\"");

    echo $res;

}

    function archivo\_impresion() {

    // Desactiva errores en pantalla y carga librería de combinación de PDFs

    ini\_set('display\_errors', 0);

    $this->load->library('pdfmerge');

    $id = $this->input->post('id');

    $codigo = $this->input->post('codigo');

    $pdf = new PDFMerger();

    // Consulta solicitud de factura por ID

    $res = $this->Conexion->consultar("SELECT SF.\* from solicitudes\_facturas SF where SF.id = $id", TRUE);

    // Itera cada letra del código recibido para determinar qué documentos incluir

    for ($i = 0; $i < strlen($codigo); $i++) {

        switch (strtoupper($codigo[$i])) {

            case 'F':

                $campo = 'f\_factura';

                break;

            case 'R':

                $campo = 'f\_remision';

                break;

            case 'O':

                $campo = 'f\_orden\_compra';

                break;

            case 'A':

                $campo = 'f\_acuse';

                break;

            case 'P':

                $campo = 'OPINION';

                break;

            case 'S':

                $campo = 'EMISION';

                break;

            default:

                $campo = null;

                break;

        }

        if($campo != null) {

            if(substr($campo, 0, 2 ) == "f\_") {

                // Documentos almacenados como BLOB en BD (factura, remisión, etc.)

                $file = $res->$campo;

                $fichero = sys\_get\_temp\_dir() . '/' . $campo . '.pdf';

                file\_put\_contents($fichero, $file);

                $pdf->addPDF($fichero, 'all');

            } else {

                // Documentos externos por código fijo (OPINION, EMISION)

                $fichero = "data/empresas/documentos\_globales/" . $campo . "\_000001.pdf";

                $pdf->addPDF($fichero, 'all');

            }

        }

    }

    // Combina y muestra los PDF en el navegador

    $pdf->merge('browser');

}

    function ajax\_getClientes() {

    // Consulta clientes registrados como empresas con rol de cliente

    $query = "SELECT C.id, C.nombre, C.razon\_social, C.foto, C.opinion\_positiva, C.emision\_sua from empresas C where C.cliente = 1";

    $res = $this->Conexion->consultar($query);

    if($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

function ajax\_getDocumentosGlobales() {

    // Consulta los documentos globales configurados para la empresa

    $query = "SELECT id, opinion\_positiva, emision\_sua from documentos\_globales where id = 1";

    $res = $this->Conexion->consultar($query);

    if($res) {

        echo json\_encode($res);

    } else {

        echo "";

    }

}

    function ajax\_filesExists($id) {

    $this->load->helper('file');

    // Rellenar el ID con ceros a la izquierda (ej. 1 => 000001)

    $id = str\_pad($id, 6, "0", STR\_PAD\_LEFT);

    // Verifica si existen los archivos PDF de ACUSE y EMISIÓN para la empresa

    $acuse = read\_file(base\_url("data/empresas/documentos\_facturacion/ACUSE\_" . $id . ".pdf")) ? "1" : "0";

    $emision = read\_file(base\_url("data/empresas/documentos\_facturacion/EMISION\_" . $id . ".pdf")) ? "1" : "0";

    echo json\_encode(array($acuse, $emision));

}

function ajax\_readXML() {

    $dom = new DomDocument;

    $dom->preserveWhiteSpace = FALSE;

    // Cargar el XML desde archivo temporal subido

    $dom->loadXML(file\_get\_contents($\_FILES['f\_X']['tmp\_name']));

    $comp = $dom->getElementsByTagName('Comprobante');

    $data = array();

    // Obtener atributos Serie y Folio del nodo <Comprobante>

    foreach ($comp[0]->attributes as $elem) {

        if($elem->name == "Serie" | $elem->name == "Folio") {

            $e = array($elem->name => $elem->value);

            array\_push($data, $e);

        }

    }

    echo json\_encode($data);

}

function ajax\_setDocumentoFacturacion() {

    $file = $this->input->post('file');

    $documento = $this->input->post('documento');

    $id = $this->input->post('empresa');

    // Formatear ID a 6 dígitos

    $id = str\_pad($id, 6, "0", STR\_PAD\_LEFT);

    // Verificar que se haya subido un archivo

    if($file != "undefined") {

        $config['upload\_path'] = 'data/empresas/documentos\_facturacion/';

        $config['allowed\_types'] = 'pdf';

        $config['overwrite'] = TRUE;

        $config['file\_name'] = $documento . '\_' . $id;

        $this->load->library('upload', $config);

        if ($this->upload->do\_upload('file')) {

            $where['id'] = $id;

            // Determinar campo a actualizar según tipo de documento

            switch($documento) {

                case "EMISION":

                    $campo = "emision\_sua";

                    break;

                case "OPINION":

                    $campo = "opinion\_positiva";

                    break;

            }

            $data[$campo] = $this->upload->data('file\_name');

            $this->Conexion->modificar('empresas', $data, null, $where);

            echo "1";

        }

    }

}

    function ajax\_deleteDocumentoFacturacion() {

    $documento = $this->input->post('documento');

    $id = $this->input->post('empresa');

    // Formatear ID a 6 dígitos (rellenando con ceros a la izquierda)

    $id = str\_pad($id, 6, "0", STR\_PAD\_LEFT);

    // Eliminar archivo físico del documento

    unlink('data/empresas/documentos\_facturacion/' . $documento . '\_' . $id . '.pdf');

    // Actualizar campo correspondiente en base de datos

    $where['id'] = $id;

    switch($documento) {

        case "EMISION":

            $campo = "emision\_sua";

            break;

        case "OPINION":

            $campo = "opinion\_positiva";

            break;

    }

    $data[$campo] = "";

    $this->Conexion->modificar('empresas', $data, null, $where);

}

function ajax\_setDocumentoGlobal() {

    $file = $this->input->post('file');

    $documento = $this->input->post('documento');

    $id = $this->input->post('empresa');

    // Formatear ID a 6 dígitos (rellenando con ceros a la izquierda)

    $id = str\_pad($id, 6, "0", STR\_PAD\_LEFT);

    // Validar que se haya recibido un archivo

    if($file != "undefined") {

        $config['upload\_path'] = 'data/empresas/documentos\_globales/';

        $config['allowed\_types'] = 'pdf';

        $config['overwrite'] = TRUE;

        $config['file\_name'] = $documento . '\_' . $id;

        $this->load->library('upload', $config);

        if ($this->upload->do\_upload('file')) {

            $where['id'] = $id;

            // Determinar campo correspondiente a actualizar

            switch($documento) {

                case "EMISION":

                    $campo = "emision\_sua";

                    break;

                case "OPINION":

                    $campo = "opinion\_positiva";

                    break;

            }

            $data[$campo] = $this->upload->data('file\_name');

            $this->Conexion->modificar('documentos\_globales', $data, null, $where);

            echo "1";

        }

    }

}

    function ajax\_deleteDocumentoGlobal() {

    $documento = $this->input->post('documento');

    $id = $this->input->post('empresa');

    // Formatear ID a 6 dígitos con ceros a la izquierda

    $id = str\_pad($id, 6, "0", STR\_PAD\_LEFT);

    // Eliminar archivo físico

    unlink('data/empresas/documentos\_globales/' . $documento . '\_' . $id . '.pdf');

    // Determinar campo a limpiar

    $where['id'] = $id;

    switch($documento) {

        case "EMISION":

            $campo = "emision\_sua";

            break;

        case "OPINION":

            $campo = "opinion\_positiva";

            break;

    }

    $data[$campo] = "";

    $this->Conexion->modificar('documentos\_globales', $data, null, $where);

}

function ajax\_enviarCorreo() {

    $id = $this->input->post('id');

    $body = $this->input->post('body');

    $para = $this->input->post('para');

    $cc = $this->input->post('cc');

    // Archivos a enviar

    $campos = ['f\_xml', 'f\_factura'];

    $archivos = [];

    $res = $this->Conexion->consultar("SELECT SF.\* from solicitudes\_facturas SF where SF.id = $id", TRUE);

    foreach ($campos as $value) {

        if(substr($value, 0, 2 ) == "f\_") {

            $file = $res->$value;

            $fichero = sys\_get\_temp\_dir(). '/' . $value . ($value == "f\_xml" ? '.xml' : '.pdf');

            file\_put\_contents($fichero, $file);

        } else {

            switch ($value) {

                case 'opinion\_positiva':

                    $value = 'OPINION';

                    break;

                case 'emision\_sua':

                    $value = 'EMISION';

                    break;

            }

            $fichero = "data/empresas/documentos\_globales/" . $value . "\_000001.pdf";

        }

        array\_push($archivos, $fichero);

    }

    // Armar datos y enviar

    $datos['id'] = $id;

    $datos['para'] = $para;

    $datos['cc'] = $cc;

    $datos['body'] = $body;

    $datos['campos'] = $campos;

    $datos['archivos'] = $archivos;

    $this->correos\_logistica->enviarCorreoFactura($datos);

}

    function ajax\_enviarCorreoLogistica() {

    $id = $this->input->post('id');

    $destinatario = $this->input->post('destinatario');

    $mensaje = $this->input->post('mensaje');

    $data['factura'] = $id;

    $data['destino'] = $destinatario;

    $data['usuario'] = $this->session->id;

    $data['estatus'] = "ENVIADA";

    $data['mensaje'] = $mensaje;

    $func['fecha'] = 'CURRENT\_TIMESTAMP()';

    $recorrido\_id = $this->Conexion->insertar('envios\_factura', $data, $func);

}

////////////////////////////////// F A C T U R A S //////////////////////////////////

function ajax\_getFacturas() {

    $id = $this->input->post('id');

    $query = "SELECT F.id, F.fecha, F.usuario, F.cliente, F.contacto, F.reporte\_servicio, F.orden\_compra, F.forma\_pago, F.pagada, F.conceptos, F.notas, F.estatus\_factura, F.documentos\_requeridos, F.serie, F.folio, F.codigo\_impresion,";

    $query .= " (SELECT count(id) from recorrido\_conceptos where factura = F.id) as Recorridos,";

    $query .= " E.nombre as Cliente, concat(U.nombre, ' ', U.paterno) as User";

    $query .= " from solicitudes\_facturas F";

    $query .= " inner join empresas E on E.id = F.cliente";

    $query .= " inner join usuarios U on U.id = F.usuario";

    $query .= " where F.folio > 0 and F.estatus = 'ACEPTADO'";

    if ($id) {

        $query .= " and F.id = '$id'";

    }

    if (isset($\_POST['estatus'])) {

        $estatus = $this->input->post('estatus');

        $query .= " and F.estatus = '$estatus'";

    }

    if (isset($\_POST['estatus\_factura'])) {

        $estatus\_factura = $this->input->post('estatus\_factura');

        $query .= " and F.estatus\_factura = '$estatus\_factura'";

    }

    if (isset($\_POST['cliente'])) {

        $cliente = $this->input->post('cliente');

        $query .= " and F.cliente = '$cliente'";

    }

    $res = $this->Conexion->consultar($query, $id);

    if ($res) {

        echo json\_encode($res);

    }

}

}