UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Trabajo de Fin de Grado

Grado en Ingeniería Informática

ALMA

Sistema de gestión integral para asociaciones sin ánimo de lucro

Autor: Alejandro Torres Pastor

Director: Manuel Llorca Alcón

Índice

Índice de imágenes

0. PREFACIO

0.1. Copyright

1. INTRODUCCIÓN

1.1. Motivación

Existen en el mercado un número elevado de aplicaciones informáticas diseñadas para la gestión contable de diferentes estructuras empresariales.

Prácticamente la totalidad de las mismas están desarrolladas desde el punto de vista contable y, tienen en común que el usuario final debe poseer unos amplios conocimientos contables para poder sacar el máximo provecho de las mismas o un gran equipo de personas, algo que las asociaciones sin ánimo de lucro no pueden permitirse.

La estructura empresarial para la que ha sido pensada mi aplicación es aquella compuesta de una 'Sede Central' con su propia gestión (administrativa y contable) de la que dependen 'Subsedes' con sus propias gestiones (administrativas y contables) cada una de ellas, pero que permita agregar todos los datos de forma que cada 'subsede' tenga su independencia sin menoscabo de conseguir una agregación de datos para poder así ofrecer una información global desde la sede central. Con las obvias salvedades contables, y las particularidades específicas, esta solución informática puede ser utilizada por empresas con estructura piramidal (Un banco con su oficina central y sus oficinas urbanas, un Ayuntamiento con sus dependencias municipales, cualquiera de las marcas comerciales que tengan tiendas diseminadas, cualquiera de las federaciones deportivas con una sede central y otras sedes provinciales/locales y como puede verse en esta simple enumeración, un sinfín de posibles destinatarios).

Pensando en este tipo de Asociaciones (sin ánimo de lucro) es por lo que he desarrollado esta solución informática que conjugue tanto la posibilidad de ser utilizada por personas sin amplios conocimientos contables pero que sus resultados sean óptimos, como que permita otras funciones no contables, agrupándolas en una única solución informática de fácil acceso y utilización.

Como indico más concretamente en los siguientes puntos, mi aplicación maneja todos los aspectos básicos de las asociaciones sin ánimo de lucro como son la contabilidad (con integración/agregación de datos a nivel piramidal de una sede central y sus delegaciones, pero sin perder la individualidad con sus respectivos presupuestos y ejecuciones presupuestarias), admisión y gestión de asociados (incluyendo lista de espera), secretaría (con envío de correos, impresión de cartas, saludas, etc.), asistencia a actividades, etc.

1.2. Objetivos

Por los motivos expuestos anteriormente, voy a focalizar el desarrollo de este trabajo en la cobertura de estos objetivos principales:

1. Permitir que cualquier persona sin profundos conocimientos sobre la contabilidad de una entidad, sea capaz de afrontar la gestión económica de la misma.

2. Incorporar las peculiaridades específicas de la contabilidad de las asociaciones sin ánimo de lucro que habitualmente suelen no estar contempladas en las aplicaciones informáticas existentes en el mercado y que estas asociaciones son capaces de permitirse.

3. Añadir la gestión habitual de cualquier entidad/asociación en cuanto a sus asociados y actividades, incluyendo labores de secretaría.

4. Crear una aplicación eficiente y usable capaz de ejecutarse en cualquier dispositivo sin requerir ningún tipo de prestación concreta.

1.3. Retos pragmáticos

El principal reto al que me he enfrentado con el desarrollo de esta solución informática es simplificarla al máximo en aras de maximizar su usabilidad.

Además de éste, también ha sido necesario lograr que el usuario final se sienta cómodo con la utilización de la solución informática, evitando así que vuelva a utilizar otros métodos alternativos y más rudimentarios (hojas de cálculo, bloc de notas, etc.).

De hecho, en el desarrollo de los interfaces ha primado más la facilidad para usar la aplicación desde el punto de vista del posible usuario final, que el uso estricto de la nomenclatura contable estándar al presentar los diferentes formularios.

1.4. Aplicaciones similares

Como he indicado anteriormente, en el mercado existen una amplia gama de aplicaciones informáticas diseñadas para la gestión contable de diferentes estructuras empresariales, pero no de asociaciones sin ánimo de lucro. De hecho, tras una exhaustiva búsqueda por la red, solo he encontrado unas pocas aplicaciones similares a la que yo estoy creando, pero todas ellas con grandes restricciones.

A continuación paso a mencionar las principales aplicaciones existentes en el mercado con un carácter muy similar al que esta memoria recoge. Incluyendo sus pros y contras:

- **Microsoft Excel:**

  Pro: Este es el software más extendido en el mercado y el más utilizado por todo aquel que necesita de una hoja de cálculo debido a su simplicidad.

  Contra: No permite ningún tipo de escalabilidad de datos y la complejidad de los resultados obtenidos dependen directamente de los conocimientos del usuario. Además es imprescindible la adquisición de una licencia de Microsoft Office.

- **OpenOffice Calc:**

  Pro: Esta es la versión de código libre de Microsoft Excel y se puede ejecutar tanto en Windows como en Linux.

  Contra: El número de herramientas es más restringido que el de Microsoft Excel y es menos utilizado por los usuarios.

- **Fundesplai (Fundación Catalana de Esplai):**

  Pro: Ofrece soporte técnico integrado con sus soluciones de software y existe una versión de demostración de todas ellas.

  Contra: Esta fundación ofrece 8 programas diferentes para la gestión administrativa y contable de las asociaciones sin ánimo de lucro, pero independientes entre sí. Para poder hacer uso de todas sus funcionalidades es necesario ser socio de su club.

- **GestiONG**

  Pro: Es una aplicación informática para la gestión de asociaciones sin ánimo de lucro y ONG's de código libre y se puede programar las funcionalidades que se deseen.

  Contra: Está de desarrollado en Linux y se ejecuta de forma local. También incluye una contabilidad y gestión de socios muy básica. Según su página web, la última publicación tiene fecha de 8 de diciembre de 2008, por lo que parece que el proyecto no se ha continuado. Además, no contempla la escalabilidad de la asociación, quedándose tan solo en una única sede.

- **Contasol**

Pro: es un programa de contabilidad profesional gratuito desarrollado para cumplir con los requerimientos fiscales de cualquier empresa. Se adapta fácilmente a las necesidades de los usuarios y ofrece la información de un modo visual y atractivo

Contra: No incluye ningún aspecto relacionado con la gestión administrativa de las asociaciones. Incluye la posibilidad de trabajar de forma colaborativa sobre los mismos datos, pero forma parte de un paquete de pago mensual.

- **Fundly CRM**

Pro: Totalmente ideado para la gestión de asociaciones sin ánimo de lucro.

Contra: Solo admite inglés y está exclusivamente ideado para el modelo contable americano, no pudiéndose adaptar o modificar. También obliga al pago de cuotas mensuales.

2. TECNOLOGÍAS UTILIZADAS

2.1. Hardware

Debido a que el principal objetivo es que se pueda utilizar en cualquier máquina sin tener en cuenta sus prestaciones y a que el foco siempre ha sido crear una aplicación simple y ligera, todo el hardware utilizado tanto para el desarrollo como el alojamiento del sitio web ha sido un ordenador portátil de características ordinarias:

- Procesador Intel Pentium 2.1GHz.
- 4GB de memoria RAM.
- Monitor de 15.6 pulgadas (1366x768 píxeles).

2.2. Software

Este proyecto de final de grado ha sido ideado para su ejecución en un entorno cliente-servidor con tal de evitar la dependencia por parte del cliente de un software pesado y, favorecer así la fácil implementación del mismo en cualquier entorno.

Para su desarrollo han sido utilizados los siguientes componentes de software:

***MySQL Server** (Oracle),* como motor de base de datos.

***MySQL Workbench** (Oracle),* para la ejecución de código SQL durante las fases tempranas de desarrollo y el análisis y diseño de las funcionalidades.

PHP 5, como lenguaje de programación para la creación de páginas dinámicas desde el lado del servidor.

HTML 5, para la creación del contenido web.

CSS 3, para el diseño visual de los diferentes componentes de las páginas web.

***Bootstrap 3** (Twitter),* como *framework* HTML, CSS y JS, para un diseño web adaptable a cualquier resolución de pantalla.

***Codeigniter 3** (EllisLab),* como framework PHP basado en el concepto modelo-vista-controlador.

JavaScript, como lenguaje de programación para la creación de contenido dinámico y llamativo.

***Sublime Text Editor** (Jon Skinner),* como editor de texto.

***Apache** (Apache Software Foundation),* como motor web.

***Microsoft Word** (Microsoft),* para la elaboración de esta memoria y el diseño técnico de la base de datos.

3. DESARROLLO DEL PROYECTO

3.1. Análisis de requerimientos

A continuación detallo los principales requisitos que considero que cualquier asociación sin ánimo de lucro podría esperar de la aplicación.

3.1.1. Definición de requerimientos

3.1.1.1. Poder ejecutar la aplicación en cualquier equipo.

3.1.1.2. Cualquier usuario debe poder utilizarlo.

3.1.1.3. Gestionar altas, bajas y modificaciones de los asociados.

3.1.1.4. Gestionar los tipos de asociados.

3.1.1.5. Gestionar las actividades a desarrollar en la Asociación y su participación.

3.1.1.6. Gestionar plenamente la contabilidad.

3.1.1.7. Implementar las siguientes funciones contables:

Recepción de facturas.
Pago de facturas.
Emisión de cobros.
Pago de cuotas de préstamos.
Pago de nóminas.
Distribuir gastos entre distintas sedes.
Remesa bancaria de recibos.
Gestionar altas y bajas de devoluciones de recibos.
Gestionar y consultar la amortización del inmovilizado.
Consultar los pagos/cobros realizados por meses.
Consultar la situación económica resumida del año en curso.

3.1.1.8. Poder elegir la fecha de cierre contable

3.1.1.9. Gestionar las actividades más usuales de la secretaría como el envío de saludas.

3.1.2. Especificación de requerimientos

3.1.2.1. La aplicación será desarrollada en un entorno cliente-servidor para liberar la carga de procesamiento del equipo del usuario y poder así ser utilizada en cualquier dispositivo.

3.1.2.2. Se aplicarán los filtros necesarios por tal de verificar que resulta accesible a cualquier usuario.

3.1.2.3. Para la gestión de los asociados, crear un formulario para las altas, donde se recogerán todos los datos susceptibles de ser utilizados a futuro y que se consideran básicos para el normal funcionamiento de la Asociación.

Así mismo crear otro formulario, para el mantenimiento o modificación de los datos alfabéticos de la base de datos.

Crear también un formulario para las 'Altas en Espera', creando una lista de espera de futuros asociados, en caso de ser necesario.

3.1.2.4. Para la gestión de los distintos tipos de asociados, diseñar un formulario donde el usuario pueda crear y modificar los distintos tipos de asociados en función de la edad de los mismos.

3.1.2.5. Gestionar el alta y modificación de las diversas actividades que la entidad llevará a cabo durante el año así como su asistencia, creando así un histórico que deberá poder ser consultado bajo distintos criterios.

3.1.2.6. Crear formularios para dar de alta y modificar las distintas cuentas contables ajustadas al plan general contable en vigor, evitando duplicidades y recogiendo todos los datos necesarios para las obligaciones fiscales/tributarias.

Crear un formulario para poder realizar los distintos apuntes contables (con y sin contrapartida), pudiendo asignar los importes a las distintas partidas presupuestarias existentes.

Crear un formulario para realizar/consultar/modificar los presupuestos anuales y hacer un seguimiento de los mismos.

3.1.2.7. Crear los siguientes formularios en términos sencillos, evitando en la medida de lo posible la terminología contable, para facilitar su utilización:

Recepción de facturas.
Pago de facturas.
Emisión de cobros.
Pago de cuotas de préstamos.
Pago de nóminas.
Distribuir gastos entre distintas sedes. (Ej.: Amortización de inmovilizado).
Remesas bancarias de recibos. (Incluir consultas individuales y agrupadas).
Gestionar altas y bajas de devoluciones de recibos.
Gestionar y consultar la amortización del inmovilizado.
Consultar los pagos/cobros realizados por meses.
Consultar la situación económica resumida del año en curso.

3.1.2.8. Permitir al usuario asignar la fecha que desee como cierre anual contable sin menoscabo de realizar las oportunas consultas del año natural para las distintas obligaciones fiscales/tributarias.

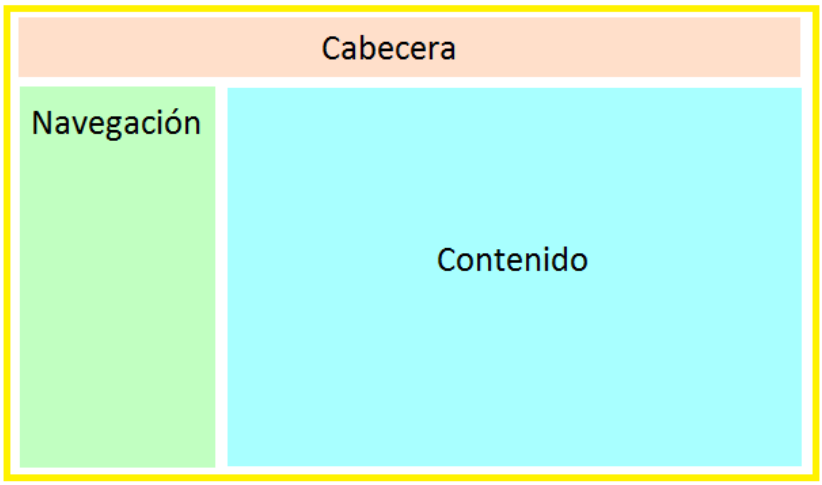
3.1.2.9. Crear una interfaz de envío de saludas, cartas y correos.

3.2. Diseño del sistema

Siguiendo con la estructura marcada por CodeIgniter por ser el framework escogido, diferenciamos tres aspectos principales, cuyos diseños son:

3.2.1.Vista o Interfaz

La visualización de cada una de las páginas se guiará por 3 capas, correspondientes a la cabecera, la navegación y contenido, tal como se muestra en la imagen:



Diseño de la estructura de las vistas.

En la esquina izquierda de la cabecera, se incluirá el nombre o el logo de la asociación, mientras que en la derecha aparecerá la información relativa al usuario actual.

La navegación se realizará con una estructura de árbol agrupándolo en bloques lógicos para facilitar la visualización del contenido, tales como:

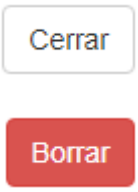
- Asociados
- Actos
- Secretaría
- Contabilidad
- Administración

Los colores elegidos serán determinados por la entidad según su guía de estilos, pero con alguna excepción que se detalla a continuación:

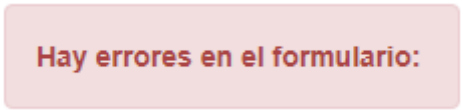
- Formularios generales:
  - Botón de cancelar: tamaño mediano con fondo naranja y letras blancas.
  - Botón de añadir: tamaño mediano con fondo verde y letras blancas.
  - Botón de "nuevo": tamaño mediano con un dibujo del signo '+' con fondo azul claro y letras blancas.

- Tablas:
  - Botón de modificar: tamaño pequeño y color azul.
  - Botón de borrar: tamaño pequeño y color rojo.

- Ventanas emergentes:
  - Botón de cerrar: tamaño mediano y color transparente con borde y letras negras.
  - Boton de borrar: tamaño mediano con color rojo y letras blancas

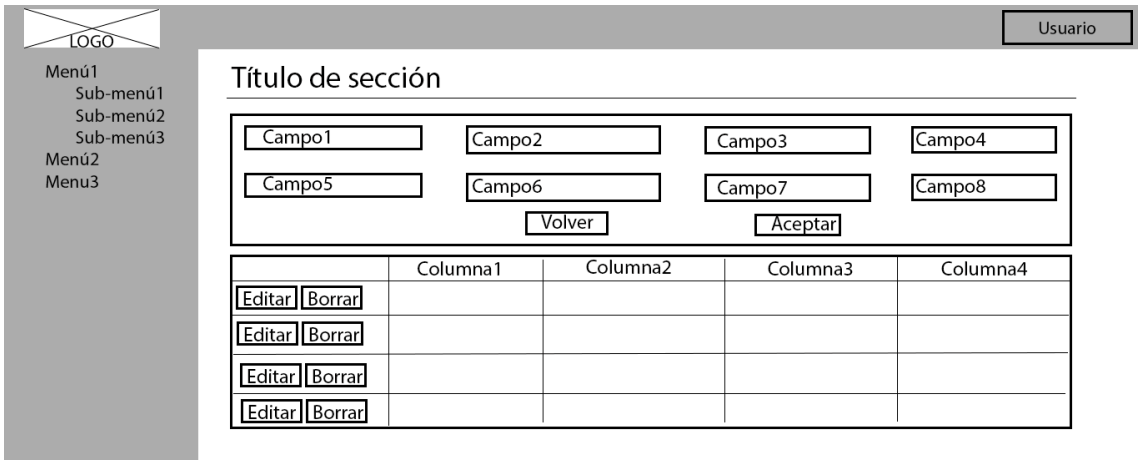


- Alertas:
  - Panel de alerta: Rectángulo rojo claro con letras y borde rojo oscuro.



Una primera aproximación al estilo final de la aplicación son los siguientes bocetos. En ellos se puede observar como quedarían dispuestos los diferentes elementos que podrían existir en una vista cualquiera, tales como el acceso de usuarios, la navegación, un formulario de entrada de datos y una tabla para la visualización de la información ya existente.

Boceto del formulario de acceso



Boceto del diseño de una vista

Tras la primera aproximación, lo siguiente es la realización de un posible diseño final de la aplicación. En el siguiente mockup se incluye algo de contenido como referencia a cómo podría quedar la visualización de la ventana en el diseño final:

Diseño de la página de acceso

Diseño de una vista

3.2.2.Controlador o Lógica de Negocio

Para la realización de este apartado he tenido en cuenta los requerimientos marcados en la fase de análisis por tal de crear toda la lógica necesaria para el buen funcionamiento de la aplicación. Además de esto, para mantener una coherencia en los datos acumulados y de cara a posibles necesidades contables, no se borrará ningún registro de la aplicación, tan solo se marcarán como cancelados. A excepción, claro está, de aquellos usuarios que lo expresen así según la LOPD.

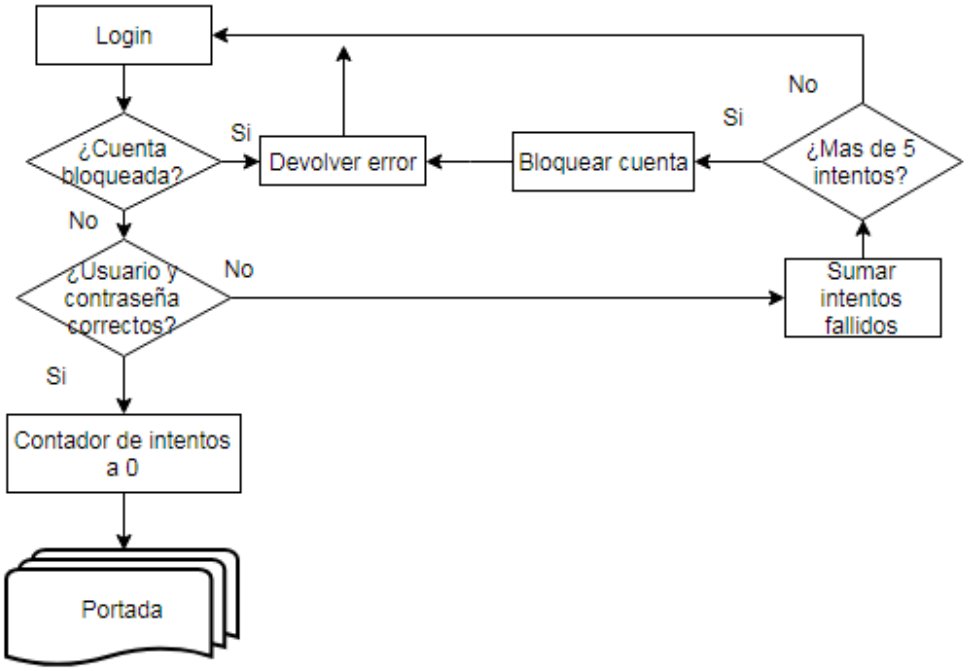


Diagrama de flujo de acceso a la aplicación

En el caso de la modificación y creación de registros, el flujo de la aplicación es prácticamente idéntico para todos los casos. Además, como he indicado antes, no se realiza ningún tipo de borrado de la base de datos, por lo que se puede replicar el siguiente diagrama para todos los casos, modificando los nombres de las distintas fases por el que corresponda:

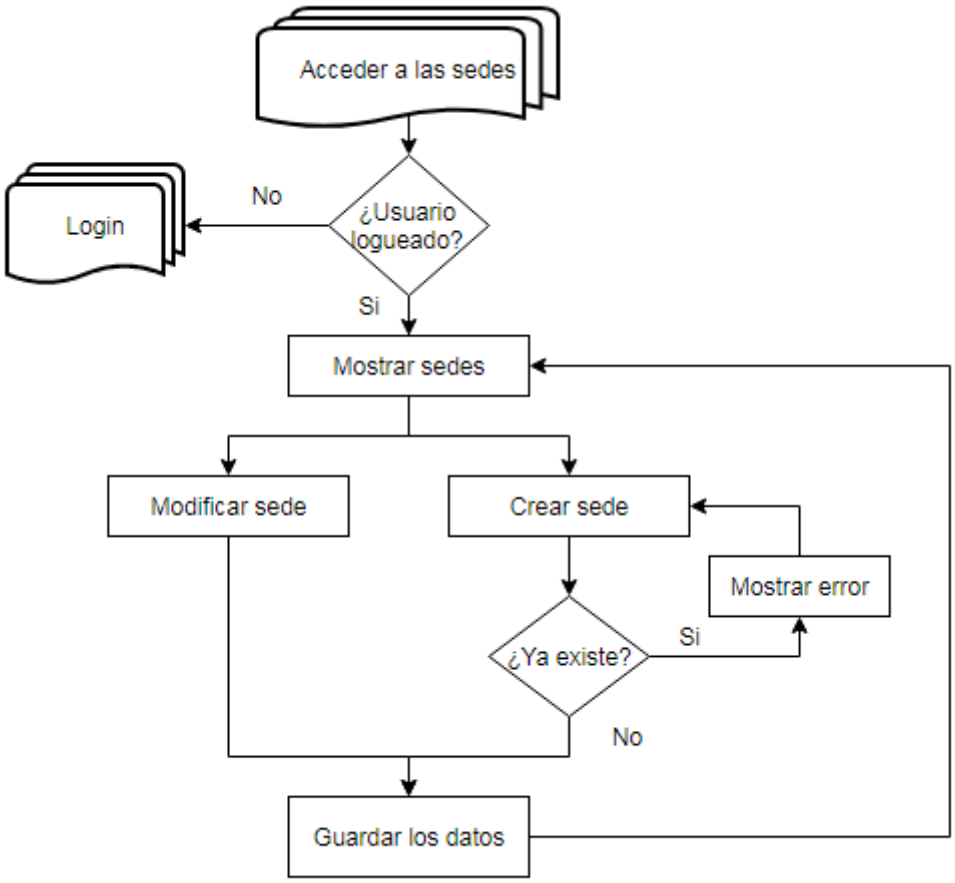
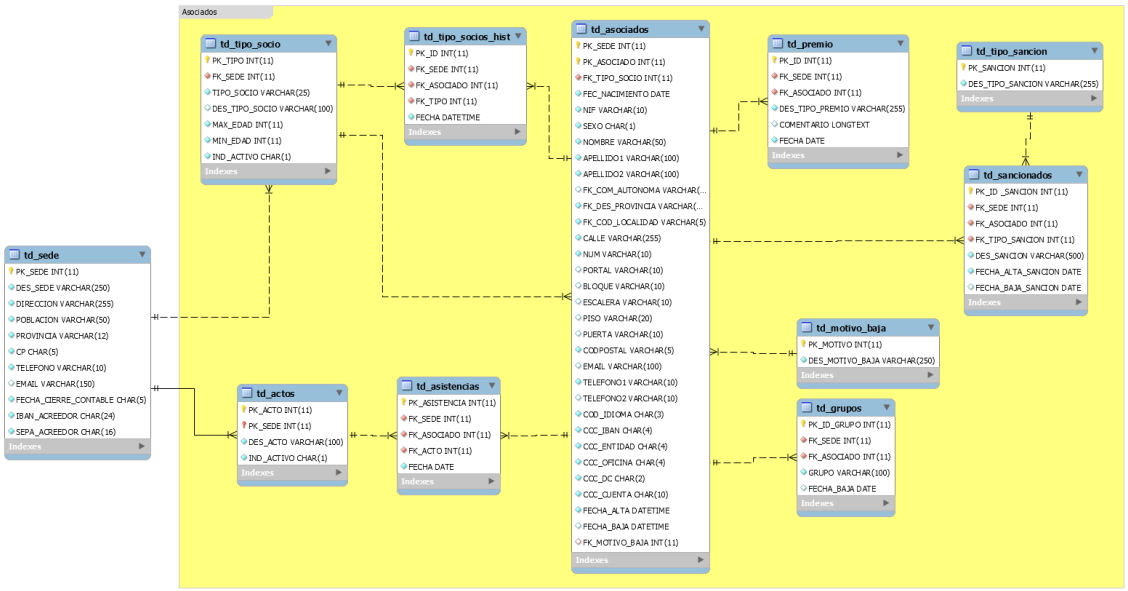


Diagrama de flujo para cualquier entrada y modificación de datos

3.2.3.Modelo o Acceso a Datos

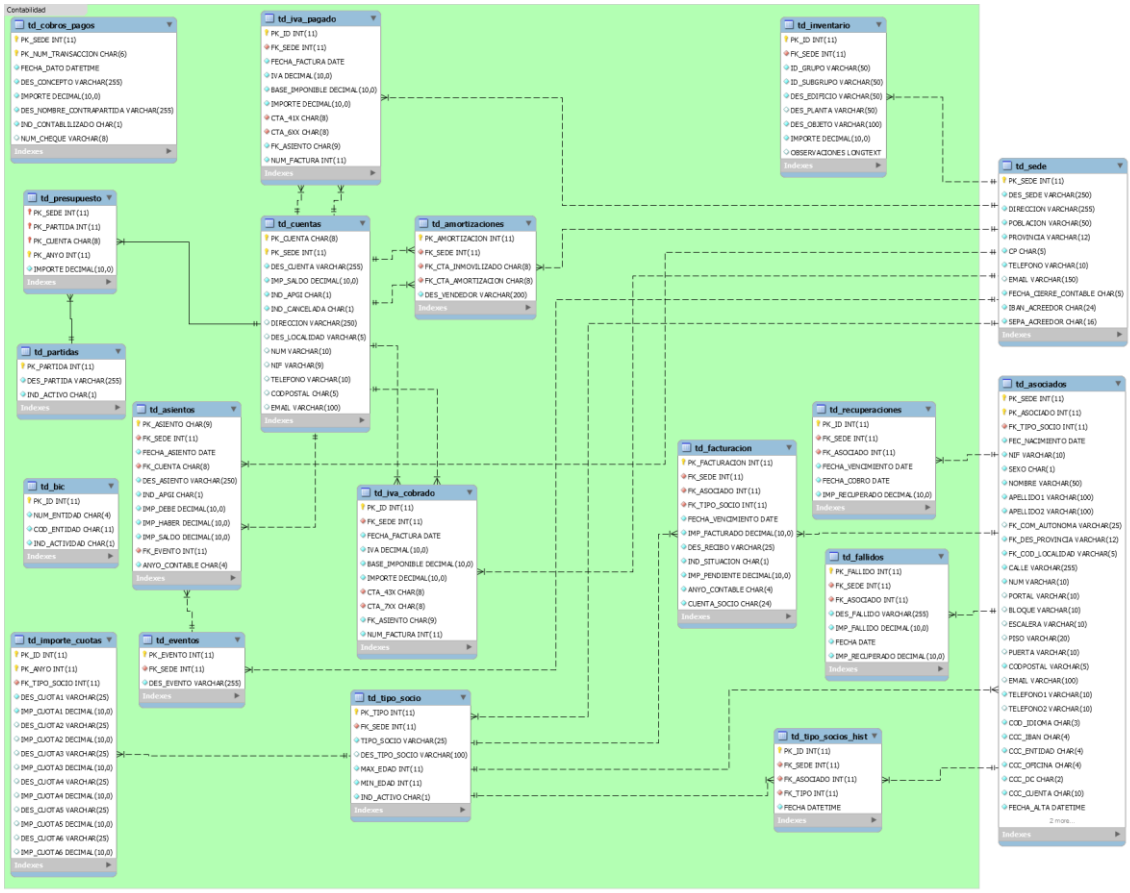
Por tal de obtener todos los datos relevantes de la aplicación de una forma cómoda y fácil de encontrar para agilizar el desarrollo y depuración de datos, el esquema de la base de datos se compone de 32 tablas, separadas en 3 modelos para una mayor organización:

- Asociados: Incluye todas las tablas referentes a la gestión de los socios de la entidad, relacionados por la sede a la que pertenecen.



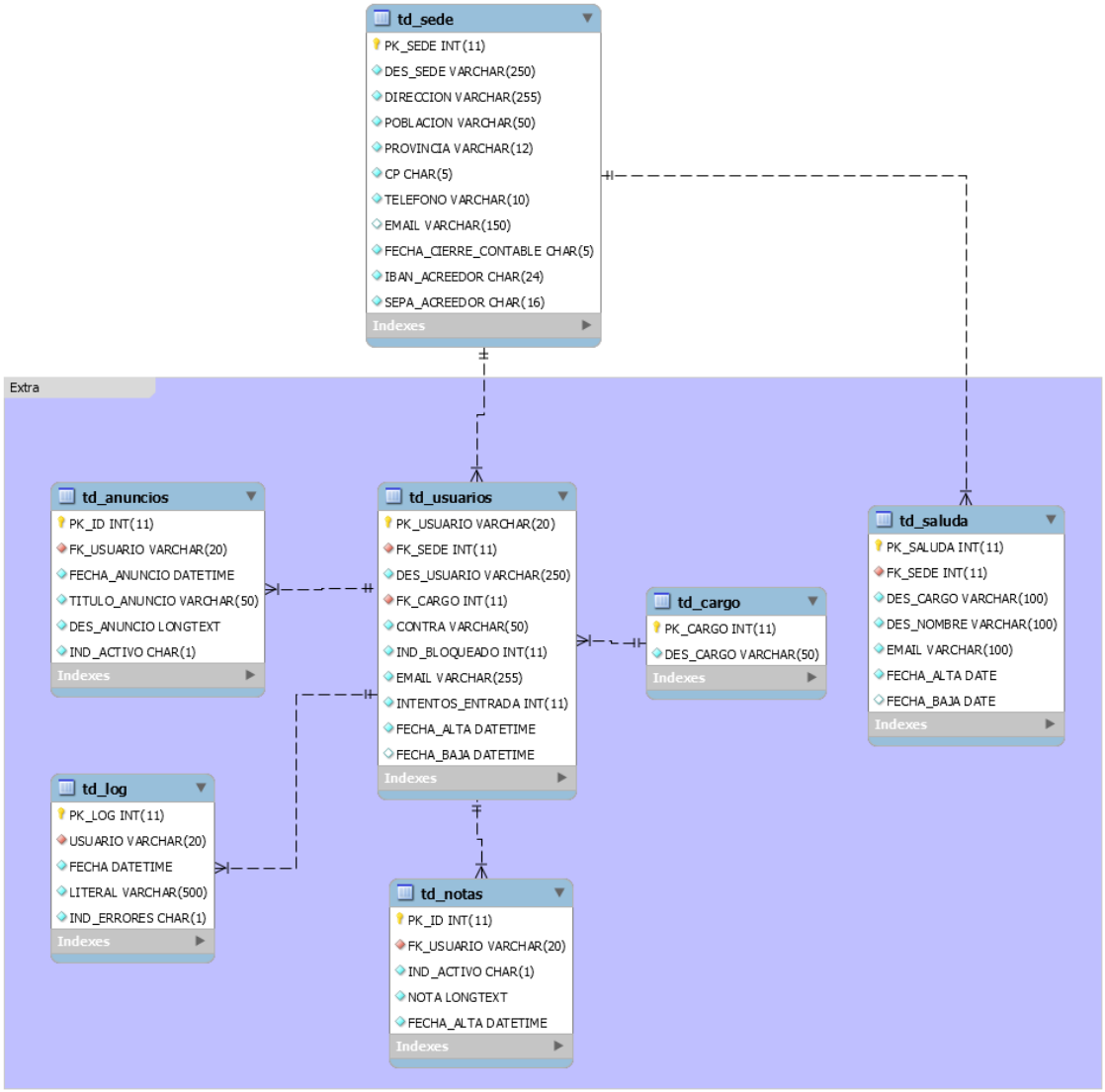
Modelo del esquema de asociados

- Contabilidad: Está formada por todas las tablas que hacen referencia a algún aspecto de la contabilidad de las asociaciones. Las tablas se vinculan a cada asociado y sede.



Modelo del esquema de contabilidad

- Funcionalidades: incluye todas las tablas que añaden funcionalidad a la aplicación, tales como la gestión de usuarios, anuncios, etc.



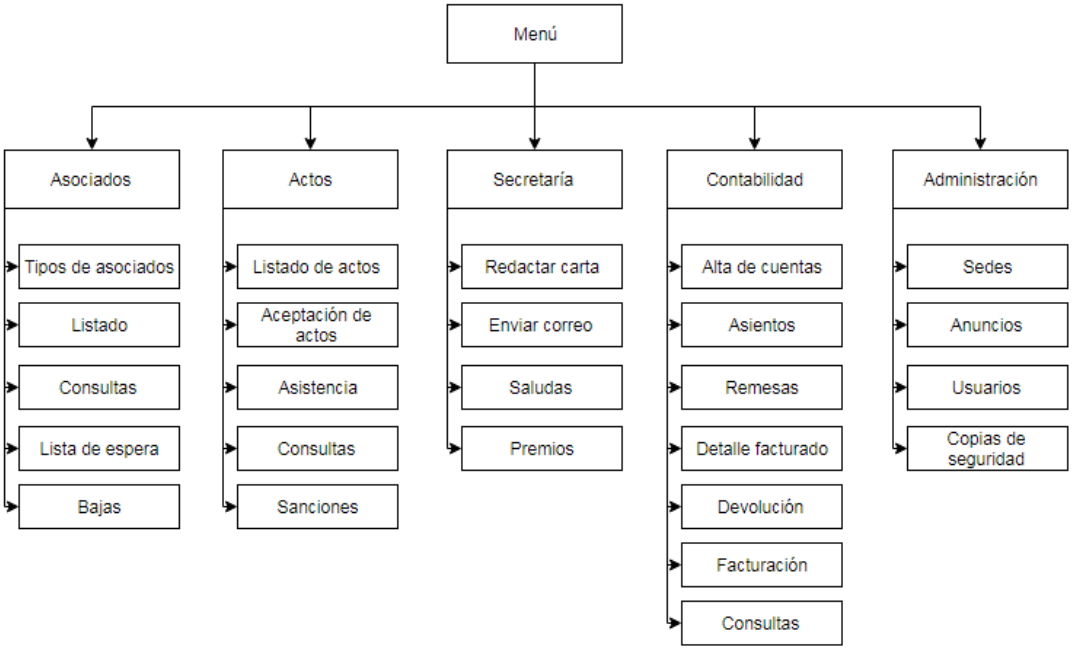
Modelo del esquema de funcionalidades

3.3. Implementación

Tras toda la fase de diseño, paso a describir las decisiones tomadas y a mostrar la implementación final de toda la aplicación separada en los mismos apartados que el punto anterior:

3.3.1.Vista o Interfaz

Tras un estudio de card-sorting, que consiste en que los usuarios elijen como agrupar las diferentes secciones de la aplicación para definir un modelo de fácil entendimiento, el menú de navegación queda dividido en 5 principales grupos de la siguiente forma:



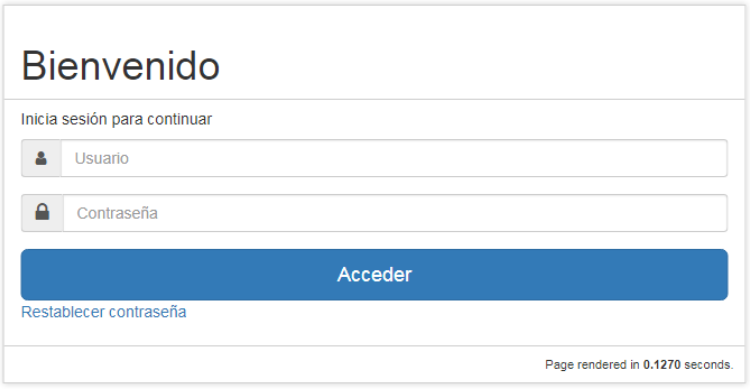
Organización definitiva del menú de navegación

El diseño definitivo para la aplicación la definirá el usuario en sus guías de estilo, pero el diseño elegido para la presentación al mismo ha sido el siguiente:

Tipo	Valor	Ejemplo
Fuente general	Helvetica (Arial o sans-serif para navegadores que no lo soporten) tamaño 14px en color negro (#000) con fondo blanco (#fff)	Texto de ejemplo
Fuente de la cabecera	Fuente general en color blanco (#fff) con fondo azul claro (#337ab7)	Texto de prueba
Borde inferior de la cabecera	Negro (#080808)	
Botones de la cabecera al pasar por encima	Fuente general con fondo negro (#000) con letras blancas (#fff)	Texto de prueba
Elementos del menú	Fuente general de color gris oscuro (#555) sobre fondo gris claro (#d0d0d0)	Texto de prueba
Elementos del menú al pasar por encima	Fuente general con fondo negro (#000) con letras blancas (#fff)	Texto de prueba
Fondo del contenido	Blanco (#fff)	
Títulos	Fuente general de tamaño 36px con una línea inferior gris clara (#d0d0d0)	Texto de ejemplo

Al usar Bootstrap como framework de diseño, tengo una amplia gama de posibilidades de fácil implementación que facilitan toda la tarea de diseño y agilizan el proceso de toma de decisiones y desarrollo del contenido.

El siguiente es un ejemplo del login de usuarios y una vista cualquiera con los formatos arriba mencionados haciendo uso de Bootstrap:



Diseño final de la ventana de login

Diseño final de una vista

A continuación detallo una parte del código desarrollado para la visualización de los datos obtenidos a partir de una consulta de una tabla. Aquí se puede apreciar como creo una tabla de 7 columnas con 2 botones en la primera de ellas para la modificación de registros y el borrado de los mismos.

```
<div class= "table-responsive">
      <table class="table table-bordered table-hover table-striped">
            <thead>
                  <tr>
                        <th>Acción</th>
                        <th>Tipo</th>
                        <th>Sede</th>
                        <th>Tipo de socio</th>
                        <th>Descripción</th>
                        <th>Edad máxima</th>
                        <th>Edad mínima</th>
                  </tr>
            </thead>
            <tbody>
                  <?php
                  foreach ($infoTiposVista->result_array() as $row) {
                        ?><tr>
                              <td align="center"><a href="<?php echo
base_url(); ?>tipo_socios/muestra/<?php echo $row['pk_tipo'];
?>"><button type="button" class="btn btn-xs btn-
primary">Editar</button></a>
                                    button type="button" id="modal1"
class="btn btn-xs btn-danger" data-toggle="modal" data-
target="#modalBorrar" onClick="<?php echo
'borrarTipo(\''.$row['pk_tipo'].'\',\''.$row['fk_sede'].'\');';?>">Bor
rar</button>
                              </td>
                              <td><?php echo $row['pk_tipo']; ?></td>
                              <td><?php echo $row['des_sede']; ?></td>
```

```
                                <td><?php echo $row['tipo_socio'];
?></td>
                                <td><?php echo $row['des_tipo_socio'];
?></td>
                                <td><?php echo $row['max_edad']; ?></td>
                                <td><?php echo $row['min_edad']; ?></td>
                            </tr><?php
                    }
                    ?>
                </tbody>
            </table>
</div>
```

Acción	Tipo	Sede	Tipo de socio	Descripción	Edad máxima	Edad mínima
Editar Borrar	1	SEDE CENTRAL	Socio completo	Socio de pleno derecho	99	10
Editar Borrar	3	SEDE CENTRAL	fgn	sdfg	544	55555

Diseño de una tabla con posibilidad de edición y borrado de registros

Para el borrado de los datos se muestra una ventana modal en la que se debe confirmar la decisión de borrado.

```
<div class="modal-dialog">
        <div class="modal-content">
                <div class="modal-header">
                        <button type="button" class="close" data-
dismiss="modal">×</button>
                        <h4 class="modal-title">¿Estas seguro?</h4>
                </div>
                <div class="modal-body" id="modalTexto">
                        <input id="ruta" type="hidden" readonly="" placeholder="Ruta"
value="http://localhost:81/" name="ruta">
                        <input id="idTipo" type="hidden" readonly="" placeholder="Id"
name="idTipo" value="1">
                        <input id="idSede" type="hidden" readonly="" placeholder="Sede"
name="idSede" value="0">
                        <p style="color:red;">Vas a borrar el tipo de socio del listado. Los
socios con este tipo seguirán existiendo hasta que les cambies el tipo. ¿Estás seguro?</p>
                </div>
```

```
                <div class="modal-footer" id="modalBotones">
                        <a id="EnlaceBorrar" <button="" type="button" class="btn btn-danger"
href="http://localhost:81/tipo_socios/borrar/1/0">Borrar</a>
                        <button type="button" class="btn btn-default" id="btnCerrar" data-
dismiss="modal">Cerrar</button>
                </div>
        </div>
</div>
```

Diseño de una ventana modal de confirmación de borrado

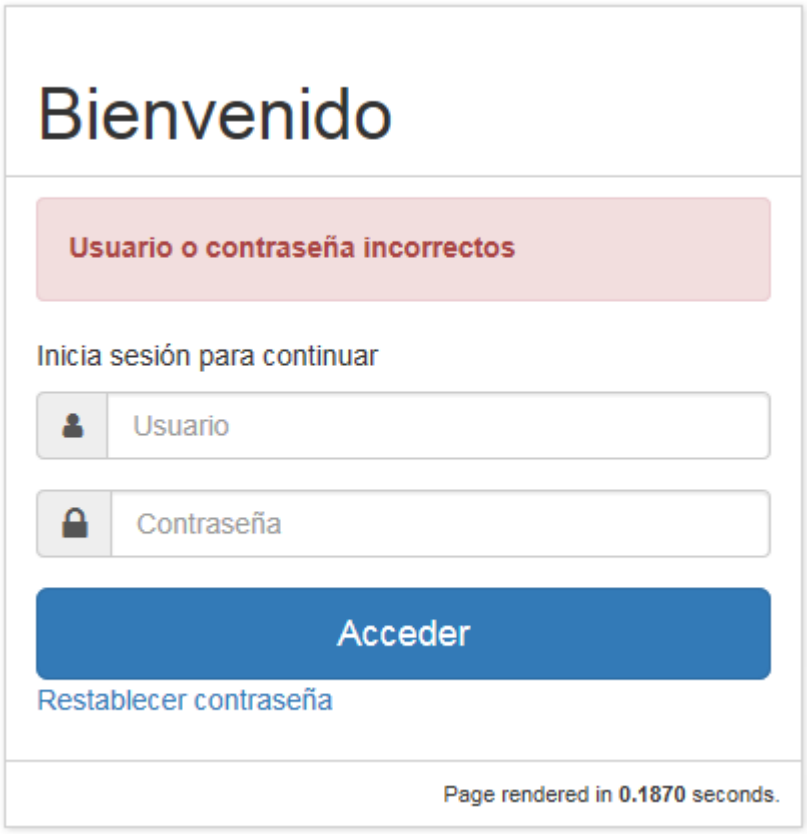
3.3.2.Controlador o Lógica de Negocio

Al estar utilizando Codeigniter como framework de desarrollo, tengo la posibilidad de usar todos los aspectos de los que el framework dispone, como el acceso a las diferentes funciones mediante la URL con el formato:

www.sitio.domino/controlador/funcion/parametro1/parametro2...

Según este formato, solo necesito indicar el controlador de php al que se hace referencia y la función y parámetros que necesite. Si no indico ninguna función, se carga la función index() por defecto. Luego solo es necesario indicar que vista quiero que se cargue.

En todas las rutas existentes en la aplicación, lo primero que se hace es comprobar que el usuario está registrado correctamente. En caso contrario se le devuelve a la ventana de inicio con un mensaje de error:



Mensaje de error obtenido al intentar acceder a una URL sin estar registrado

Además de esto, existe mucha más lógica implementada en la aplicación, de la cual paso a mostrar algunos de los fragmentos más relevantes:

```
public function index()
	{
	//cargo el modelo de tipos de socios
	$this->load->model('tipo_socios_model');

	//sede del usuario conectado
	$sede = $this->session->sede;
	$datos_vista['sede'] = $sede;

	//pido los tipos de usuario para la sede al modelo
	$tipos = $this->tipo_socios_model->muestra_tipos($sede);

	//creo el array con datos de configuración para la vista
	$datos_vista = array('infoTiposVista' => $tipos);
```

```
        $datos_vista['user'] = $this->session->nombre;

        //cargo la vista pasando los datos de configuacion
        $this->load->view('cabecera', $datos_vista);
           $this->load->view('menu', $datos_vista);
        $this->load->view('tipo_socios', $datos_vista);
           $this->load->view('footer', $datos_vista);
    }
```

En este primer fragmento, se carga toda la información referente a la vista 'tipo_socios' de la función index. Es un ejemplo muy simple de cómo funciona este framework, pero que ilustra bastante bien lo simple que es.

El siguiente fragmento es algo más complejo, ya que forma parte de un formulario de modificación de registros, en el que se comprueba la integridad de datos para evitar una posible inyección en la base de datos y proteger así la integridad de la misma:

```
        function muestra($id){

                //cargo el modelo de tipos de socios
                $this->load->model('tipo_socios_model');

                //pido el tipo que se desea ver
                $tipo = $this->tipo_socios_model->info_tipo($id);

                //creo el array con los datos para la vista
                $datos_vista['TipoVista'] = $tipo;
                $datos_vista['user'] = $this->session->nombre;

                //Variable de control de errores
                $errorModificar = 0;

                //genero el campo para mostrar errores y cargo sobre el
los strings
                $datos_vista['err_mensajes'] = "
                <div class='alert alert-danger'>
                                <strong>Hay errores en el
formulario:</strong>
                ";

                //compruebo si viene algo por el POST
                if(isset($_POST['control'])){//Si viene es porque ya se ha
modificado algo y vamos a actualizar la bd
```

```
            if(empty($_POST['nomTipo'])){//compruebo si viene
informado
                $errorModificar ++;
                $datos_vista['err_mensajes'] =
$datos_vista['err_mensajes']."<p>".$errorModificar.": Debes indicar un
título</p>";
            }else if (!preg_match('/^[a-zA-Z
]*$/',$_POST['nomTipo'])){
                $errorModificar ++;
                $datos_vista['err_mensajes'] =
$datos_vista['err_mensajes']."<p>".$errorModificar.": El título solo
admite letras</p>";
            }

            if(!empty($_POST['descripcion'])){//compruebo si viene
informado
                if (!preg_match('/^[a-zA-Z
]*$/',$_POST['descripcion'])){
                    $errorModificar ++;
                    $datos_vista['err_mensajes'] =
$datos_vista['err_mensajes']."<p>".$errorModificar.": La descripción
solo admite letras</p>";
                }
            }

            if(empty($_POST['maxEdad'])){//compruebo si viene
informado
                $errorModificar ++;
                $datos_vista['err_mensajes'] =
$datos_vista['err_mensajes']."<p>".$errorModificar.": Debes indicar
una edad máxima</p>";
            }else if (!preg_match('/^[0-9]*$/',$_POST['maxEdad'])){
                $errorModificar ++;
                $datos_vista['err_mensajes'] =
$datos_vista['err_mensajes']."<p>".$errorModificar.": La edad máxima
solo admite números</p>";
            }

            if(empty($_POST['minEdad'])){//compruebo si viene
informado
                $errorModificar ++;
                $datos_vista['err_mensajes'] =
$datos_vista['err_mensajes']."<p>".$errorModificar.": Debes indicar
una edad mínima</p>";
            }else if (!preg_match('/^[0-9]*$/',$_POST['minEdad'])){
                $errorModificar ++;
                $datos_vista['err_mensajes'] =
$datos_vista['err_mensajes']."<p>".$errorModificar.": La edad mínima
solo admite números</p>";
```

```
                }

                $datos_vista['err_mensajes'] =
$datos_vista['err_mensajes']."</div>";

                //compruebo si ha habido algun fallo al verificar
                if(!$errorModificar>0){
                   //acabo el update volviendo al listado de sedes
                   $this->tipo_socios_model->modifico_tipo($id, $this-
>session->sede, $_POST['nomTipo'], $_POST['descripcion'],
$_POST['maxEdad'], $_POST['minEdad']);
                   redirect("tipo_socios");
                }
                //else vuelvo a cargar el formulario con los datos ya
guardados y un mensaje de error
            }

            //compruebo si he recibido un numero
            if (!$id){
               //no he recibido ningún valor
               //voy a lanzar un error 404 de página no encontrada
               show_404();
            }else{

               //conteo de errores
               $datos_vista['errores'] = $errorModificar;

               //he encontrado la sede
               //muestro la vista de la página de mostrar la sede
pasando los datos del array de la misma
               $this->load->view('cabecera', $datos_vista);
                  $this->load->view('menu', $datos_vista);
               $this->load->view('muestra_tipo_socios', $datos_vista);
               $this->load->view('footer', $datos_vista);
            }
         }
```

Otro de los aspectos importantes en la aplicación es el 'borrado' de datos. Como he indicado anteriormente, no se elimina ningún registro de la base de datos, ya que aunque a la vista del usuario el registro no se vuelve a mostrar, este sigue existiendo en la base de datos pero con un indicador de 'cancelado' activo y una fecha de baja del registro, como se muestra en el código siguiente:

```
function borrar($id,$sede){
    //cargo el modelo
    $this->load->model('tipo_socios_model');

    //marcas de tiempo
    $datestring = '%Y-%m-%d %H:%i:%s';
    $time = time();
    $fecha = mdate($datestring, $time);

    $this->tipo_socios_model->baja($id, $sede, $fecha);
    redirect("tipo_socios");
  }
```

A parte de esto, también es de interés algún fragmento como la asignación del tipo de cuenta contable a partir de los cuatro primeros dígitos de la numeración que el usuario asigna a cada cuenta, de esta forma se distinguen automáticamente las cuentas de Activos, Pasivos, Gastos e Ingresos:

```
$valor = $_POST['numCuenta'];//numCuenta son los 4 primeros digitos
          switch (true){
          case (($valor>=2000 && $valor<=2099)||
                    ($valor>=2100 && $valor<=2799)||
                    ($valor>=3000 && $valor<=3899)||
                    ($valor>=4070 && $valor<=4099)||
                    ($valor>=4300 && $valor<=4379)||
                    ($valor>=4400 && $valor<=4499)||
                    ($valor>=4600 && $valor<=4649)||
                    ($valor>=4700 && $valor<=4749)||
                    ($valor>=4800 && $valor<=4899)||
                    ($valor>=5300 && $valor<=5399)||
                    ($valor>=5400 && $valor<=5499)||
                    ($valor>=5500 && $valor<=5549)||
                    ($valor>=5560 && $valor<=5599)||
                    ($valor>=5650 && $valor<=5679)||
                    ($valor>=5700 && $valor<=5799)||
                    ($valor>=5800 && $valor<=5849)
                    ):
                $datos_vista['apgi'] = "Activo";
                $apgi = "A";
                break;
```

```
            case (($valor>=1000 && $valor<=1999)||
                  ($valor>=2800 && $valor<=2999)||
                  ($valor>=3900 && $valor<=3999)||
                  ($valor>=4000 && $valor<=4069)||
                  ($valor>=4100 && $valor<=4199)||
                  ($valor>=4380 && $valor<=4399)||
                  ($valor>=4650 && $valor<=4699)||
                  ($valor>=4750 && $valor<=4799)||
                  ($valor>=4900 && $valor<=4999)||
                  ($valor>=5000 && $valor<=5099)||
                  ($valor>=5100 && $valor<=5199)||
                  ($valor>=5200 && $valor<=5299)||
                  ($valor>=5550 && $valor<=5559)||
                  ($valor>=5600 && $valor<=5619)||
                  ($valor>=5680 && $valor<=5699)||
                  ($valor>=5850 && $valor<=5899)||
                  ($valor>=5900 && $valor<=5999)
                  ):
               $datos_vista['apgi'] = "Pasivo";
               $apgi = "P";
               break;
            case ($valor>=6000 && $valor<=6999):
               $datos_vista['apgi'] = "Gasto";
               $apgi = "G";
               break;
            case ($valor>=7000 && $valor<=7999):
               $datos_vista['apgi'] = "Ingreso";
               $apgi = "I";
               break;
            default:
               $datos_vista['apgi'] = "Cuenta no válida";
               $errorModificar ++;
               $err_mensajes = $err_mensajes."<p>".$errorModificar.":
La cuenta no es correcta</p>";
               break;
         }
```

3.3.3.Modelo o Acceso a Datos

Otro de los beneficios del uso de Codeigniter es la fácil configuración del acceso a la base de datos y el trabajo con la misma. Basta con indicar los datos de acceso al motor de base de datos y éste se encarga de abrir y cerrar las conexiones necesarias.

A continuación muestro unos fragmentos de código del modelo de socios en el que se incluyen ejemplos de las funciones más utilizadas en toda la aplicación, como son la inserción de datos, modificaciones o extracciones de los mismos.

Es de destacar que aunque en el controlador ya realice las comprobaciones necesarias para evitar la inyección de código SQL en la base de datos mediante las variables excluyendo los signos potencialmente más peligrosos, en los campos más susceptibles a las modificaciones del usuario como las URL, esta comprobación se realiza de nuevo aquí para asegurar que el usuario no haya interferido en las variables de la transacción SQL.

```
   function muestra_socios($sede){
      return $ssql = $this->db->query('SELECT se.des_sede as nom_sede,
aso.pk_asociado, ts.tipo_socio as nom_tipo_socio, aso.nombre,
aso.apellido1, aso.apellido2, aso.fecha_alta
         FROM dw_sgaso.td_asociados aso
         left join dw_sgaso.td_sede se
         on aso.pk_sede = se.pk_sede
         left join dw_sgaso.td_tipo_socio ts
         on aso.pk_sede = ts.fk_sede
         and aso.fk_tipo_socio = ts.pk_tipo
         where aso.fecha_baja is null and  aso.pk_sede like "%' .
      $this->db->escape_like_str($sede).'%" ESCAPE "!"');
   }


   function crea_socio($sede,
      $numSocio,
      $tipoSocio,
      $fecNacimiento,
      $nif,
      $sexo,
      $nombre,
      $apellido1,
      $apellido2,
      $comAutonoma,
      $provincia,
      $localidad,
      $calle,
      $num,
```

```
$portal,
$bloque,
$escalera,
$piso,
$puerta,
$cp,
$email,
$tel1,
$tel2,
$idioma,
$iban,
$entidad,
$oficina,
$dc,
$cuenta,
$fecha_alta){
$ssql = "INSERT INTO dw_sgaso.td_asociados
   (PK_SEDE,
   PK_ASOCIADO,
   FK_TIPO_SOCIO,
   FEC_NACIMIENTO,
   NIF,
   SEXO,
   NOMBRE,
   APELLIDO1,
   APELLIDO2,
   FK_COM_AUTONOMA,
   FK_DES_PROVINCIA,
   FK_COD_LOCALIDAD,
   CALLE,
   NUM,
   PORTAL,
   BLOQUE,
   ESCALERA,
   PISO,
   PUERTA,
   CODPOSTAL,
   EMAIL,
   TELEFONO1,
   TELEFONO2,
   COD_IDIOMA,
   CCC_IBAN,
   CCC_ENTIDAD,
   CCC_OFICINA,
   CCC_DC,
   CCC_CUENTA,
   FECHA_ALTA)
   VALUES
   ('".$sede."',
```

```
            '".$numSocio."',
            '".$tipoSocio."',
            '".$fecNacimiento."',
            '".$nif."',
            '".$sexo."',
            '".$nombre."',
            '".$apellido1."',
            '".$apellido2."',
            '".$comAutonoma."',
            '".$provincia."',
            '".$localidad."',
            '".$calle."',
            '".$num."',
            '".$portal."',
            '".$bloque."',
            '".$escalera."',
            '".$piso."',
            '".$puerta."',
            '".$cp."',
            '".$email."',
            '".$tel1."',
            '".$tel2."',
            '".$idioma."',
            '".$iban."',
            '".$entidad."',
            '".$oficina."',
            '".$dc."',
            '".$cuenta."',
            '".$fecha_alta."');";
         $this->db->query($ssql);
      }


      function nuevo_socio($sede){
         $ssql = 'select COALESCE(MAX(aso.PK_ASOCIADO)+1,0) as
PK_ASOCIADO, se.des_sede as DES_SEDE FROM dw_sgaso.td_asociados aso
left join dw_sgaso.td_sede se on aso.pk_sede = se.pk_sede where
aso.PK_SEDE like '.$sede;
         $num = $this->db->query($ssql);
         return $num->row();
      }


      function modifico_socio($id,
         $sede,
         $tipoSocio,
         $fecNacimiento,
         $nif,
         $sexo,
```

```
$nombre,
$apellido1,
$apellido2,
$comAutonoma,
$provincia,
$localidad,
$calle,
$num,
$portal,
$bloque,
$escalera,
$piso,
$puerta,
$cp,
$email,
$tel1,
$tel2,
$idioma,
$iban,
$entidad,
$oficina,
$dc,
$cuenta){
$ssql = "UPDATE dw_sgaso.td_asociados
   SET
   FK_TIPO_SOCIO = '".$tipoSocio."',
   FEC_NACIMIENTO = '".$fecNacimiento."',
   NIF = '".$nif."',
   SEXO = '".$sexo."',
   NOMBRE = '".$nombre."',
   APELLIDO1 = '".$apellido1."',
   APELLIDO2 = '".$apellido2."',
   FK_COM_AUTONOMA = '".$comAutonoma."',
   FK_DES_PROVINCIA = '".$provincia."',
   FK_COD_LOCALIDAD = '".$localidad."',
   CALLE = '".$calle."',
   NUM = '".$num."',
   PORTAL = '".$portal."',
   BLOQUE = '".$bloque."',
   ESCALERA = '".$escalera."',
   PISO = '".$piso."',
   PUERTA = '".$puerta."',
   CODPOSTAL = '".$cp."',
   EMAIL = '".$email."',
   TELEFONO1 = '".$tel1."',
   TELEFONO2 = '".$tel2."',
   COD_IDIOMA = '".$idioma."',
   CCC_IBAN = '".$iban."',
   CCC_ENTIDAD = '".$entidad."',
```

```
         CCC_OFICINA = '".$oficina."',
         CCC_DC = '".$dc."',
         CCC_CUENTA = '".$cuenta."'
         WHERE PK_SEDE = '".$sede."' AND PK_ASOCIADO = '".$id."'";
      $this->db->query($ssql);
   }


   function existe($id,$sede){
      $ssql = "select pk_asociado from dw_sgaso.td_asociados where
pk_sede LIKE '" .
      $this->db->escape_like_str($sede)."' ESCAPE '!' and pk_asociado
like '".
      $this->db->escape_like_str($id)."' ESCAPE '!'";
      $rs = $this->db->query($ssql);
      if($rs->num_rows()>0){
         return true;
      }else return false;

   }


   function info_socio($id,$sede){
      $ssql = 'select ba.des_motivo_baja, se.des_sede,
ts.des_tipo_socio, aso.* from dw_sgaso.td_asociados aso
         left join dw_sgaso.td_sede se
         on se.pk_sede = aso.pk_sede
         left join dw_sgaso.td_tipo_socio ts
         on ts.pk_tipo = aso.fk_tipo_socio
         left join dw_sgaso.td_motivo_baja ba
         on ba.pk_motivo = aso.fk_motivo_baja
         where aso.pk_sede = ' .$sede. ' and aso.pk_asociado = '.$id;
      $rs = $this->db->query($ssql);
      if ($rs->num_rows()==0){
         return false;
      }else{
         $rs = $this->db->query($ssql);
         return $rs->row_array();
      }
   }


   function tipo_socios($sede){
      $ssql = 'select * from dw_sgaso.td_tipo_socio where ind_activo =
1 and fk_sede = '.$sede;
      return $this->db->query($ssql);
   }
```

```
   function motivos_baja(){
      $ssql = 'SELECT * FROM dw_sgaso.td_motivo_baja';
      return $this->db->query($ssql);
   }


   function baja($id,$sede, $motivo, $fecha){
      $ssql = "UPDATE dw_sgaso.td_asociados
         SET
         FECHA_BAJA = '".$fecha."',
         FK_MOTIVO_BAJA = '".$this->db->escape_like_str($motivo)."'
         WHERE PK_SEDE = '".$this->db->escape_like_str($sede)."' AND
PK_ASOCIADO = '".$this->db->escape_like_str($id)."';";
      $this->db->query($ssql);
   }
```

4. APLICACIONES REALES

- Asociación de San Jorge (Alcoy)

  La Asociación abarca las distintas 'Filaes' (según las distintas poblaciones, toman los nombres de 'Comparsas', 'Kábilas', 'Zocos', etc.)

- Juntas Centrales Falleras

  La Junta abarca todas y cada una de las 'Fallas' de su ciudad

- Federación de Hogueras

  La Federación abarca todas y cada una de las 'Hogueras' de su ciudad

- Federaciones deportivas (de futbol, baloncesto, atletismo, etc.)

  Cada Federación tiene una sede central y distintas sedes provinciales/locales, etc.

Y en definitiva cualquier Asociación/Entidad sin ánimo de lucro que tenga sedes dependientes, que mantengan su autonomía (en todos los sentidos), pero con la posibilidad de agregar todos los datos para obtener toda la información.

5. CONCLUSIONES

La realización de este trabajo me ha aportado una gran experiencia en el entorno del desarrollo de aplicaciones web, ya que es algo que nunca antes había hecho y tampoco había aprendido hasta ahora.

El desarrollo web es un área que siempre está avanzando y con él las tecnologías existentes. De no ser por haber descubierto algún que otro framework, todavía estaría indeciso sobre cómo llevar a cabo ciertos aspectos del diseño y el desarrollo.

Además, la creación de este proyecto me ha enseñado la importancia del uso de buenas prácticas tales como añadir comentarios a cada línea de código para agilizar la depuración de errores, la creación de un calendario de trabajo detallado para evitar demorarme demasiado en ciertos puntos del proyecto, y la organización del tiempo para poder compaginar el proyecto con las prácticas en empresa.

Aunque las clases impartidas han ayudado mucho en cada aspecto de este proyecto, a la hora de la realidad ha sido necesario ampliar mis conocimientos específicos sobre la programación web, ya que el PHP es un lenguaje con el que no había tratado todavía.

Así mismo, las prácticas en empresa que estoy realizando actualmente junto con los conocimientos en SQL ya adquiridos, han facilitado y mucho todo el trabajo relativo a la capa de datos.

En definitiva, este proyecto ha supuesto todo un reto para mí, no solo a nivel académico, sino también personal y sin duda una gran oportunidad para poner a prueba todo lo aprendido durante estos últimos 4 años.

6. BIBLIOGRAFÍA Y DOCUMENTACIÓN CONSULTADA

6.1. Aplicaciones similares:

http://gestiong.sourceforge.net/

http://www.fundlycrm.com/

https://www.sdelsol.com/programa-contabilidad-contasol/

http://www.suport.org/

https://products.office.com/es-es/excel

https://es.libreoffice.org/descubre/calc/

6.2. Documentación:

http://php.net/manual/es/intro-whatis.php

https://www.w3schools.com/php/default.asp

https://desarrolloweb.com/php/

https://codeigniter.com/docs

http://getbootstrap.com/

https://www.w3schools.com/css/css3_intro.asp

https://startbootstrap.com/template-overviews/sb-admin/

https://www.w3schools.com/js/

https://www.w3schools.com/html/html5_intro.asp

http://www.forosdelweb.com/

6.3. Software empleado

https://www.sublimetext.com/

https://www.mysql.com/products/workbench/

https://www.apachefriends.org/es/index.html

7. AGRADECIMIENTOS

Quiero dar las gracias a todas aquellas personas que han hecho posible que haya logrado llegar hasta aquí, porque sin su ayuda no habría conseguido finalizar este proyecto:

A todos los profesores del Grado en Ingeniería Informática del Campus de Alcoy de la Universidad Politécnica de Valencia por transmitirme sus inestimables y valiosos conocimientos y por estos inolvidables 4 años de aprendizaje.

A mis compañeros de empresa, por ayudarme a ampliar mis conocimientos sobre bases de datos y resolver aquellas cuestiones que les planteaba de forma totalmente altruista.

En especial a Manuel Llorca Alcón, director de este trabajo de fin de grado, por confiar en mí idea y estar dispuesto a llevarla a cabo conmigo y por todos sus consejos y revisiones tanto sobre el proyecto en sí, como sobre esta misma memoria.

Y en definitiva, a todos aquellos que me han ayudado de una forma u otra, ya sea valorando mis ideas en este trabajo o respondiendo con paciencia todas las dudas que les he ido planteando.