

Elecciones Sindicales

 Autor: Andrei Alexandru Miu

Institución: I.E.S Comercio

Desarrollo de Aplicaciones Web

Logroño, La Rioja

2023

Índice

Introducción	3
Objetivos	3
Fases del proyecto.....	4
Diseño.....	4
Implementación y configuración.....	11
Pruebas.....	22
Despliegue	24
Ampliación y posibles mejoras.....	24
Conclusión	24
Bibliografía	25

Introducción

El presente Trabajo de Fin de Grado (TFG) tiene como objetivo examinar, analizar y realizar un recuento de los votos emitidos durante las elecciones sindicales.

Las elecciones sindicales representan un momento crucial en la vida de una organización, ya que son el mecanismo democrático mediante el cual los trabajadores eligen a sus representantes para la defensa de sus derechos e intereses.

Víctor, trabajador del ANPE (el cliente a partir de ahora), ha sido el que me ha comunicado las funciones y objetivos principales que quiere que tenga la aplicación.

Todas las imágenes relacionadas al ANPE fueron consultadas al cliente y ha dado su permiso de que pudiese usarlas.

Objetivos

El objetivo del proyecto es desarrollar una aplicación sencilla y fácil de utilizar donde:

Exista un usuario administrador (ya creado en la base de datos), el cual pueda:

- Gestionar las mesas:
 - Ver una lista de mesas.
 - Añadir mesas.
 - Editar el nombre de las mesas.
 - Eliminar mesas.
- Gestionar los votantes:
 - Ver una lista de votantes.
 - Editar los campos de los votantes.
 - Eliminar los votantes.
- Gestionar los afiliados:
 - Ver una lista de los afiliados.
 - Editar los campos de los afiliados.
 - Eliminar los afiliados.
- Gestionar los interventores:
 - Ver una lista de interventores.
 - Añadir interventores.
 - Editar los campos de los interventores.
 - Eliminar los interventores.
- Un apartado con estadísticas.
- Que se puedan filtrar datos de votantes y afiliados.
- El administrador podrá ver la contraseña de los interventores.

Los usuarios interventores por su parte podrán:

- Ver una lista de los votantes de su mesa, evitando ver los votantes de otras mesas.
- Cambiar el voto del votante de su mesa.

Fases del proyecto

Diseño

Componentes software necesarios:

- Un IDE (Entorno de desarrollo integrado).
- Una herramienta capaz de administrar y desarrollar bases de datos.
- Un entorno de desarrollo para crear y probar la aplicación.

En este caso se ha optado por utilizar Visual Studio Code, SSMS y XAMPP para el desarrollo del proyecto.

Componentes hardware necesarios:

- Un dispositivo capaz de visualizar/developar la aplicación.

Diagrama de la base de datos:

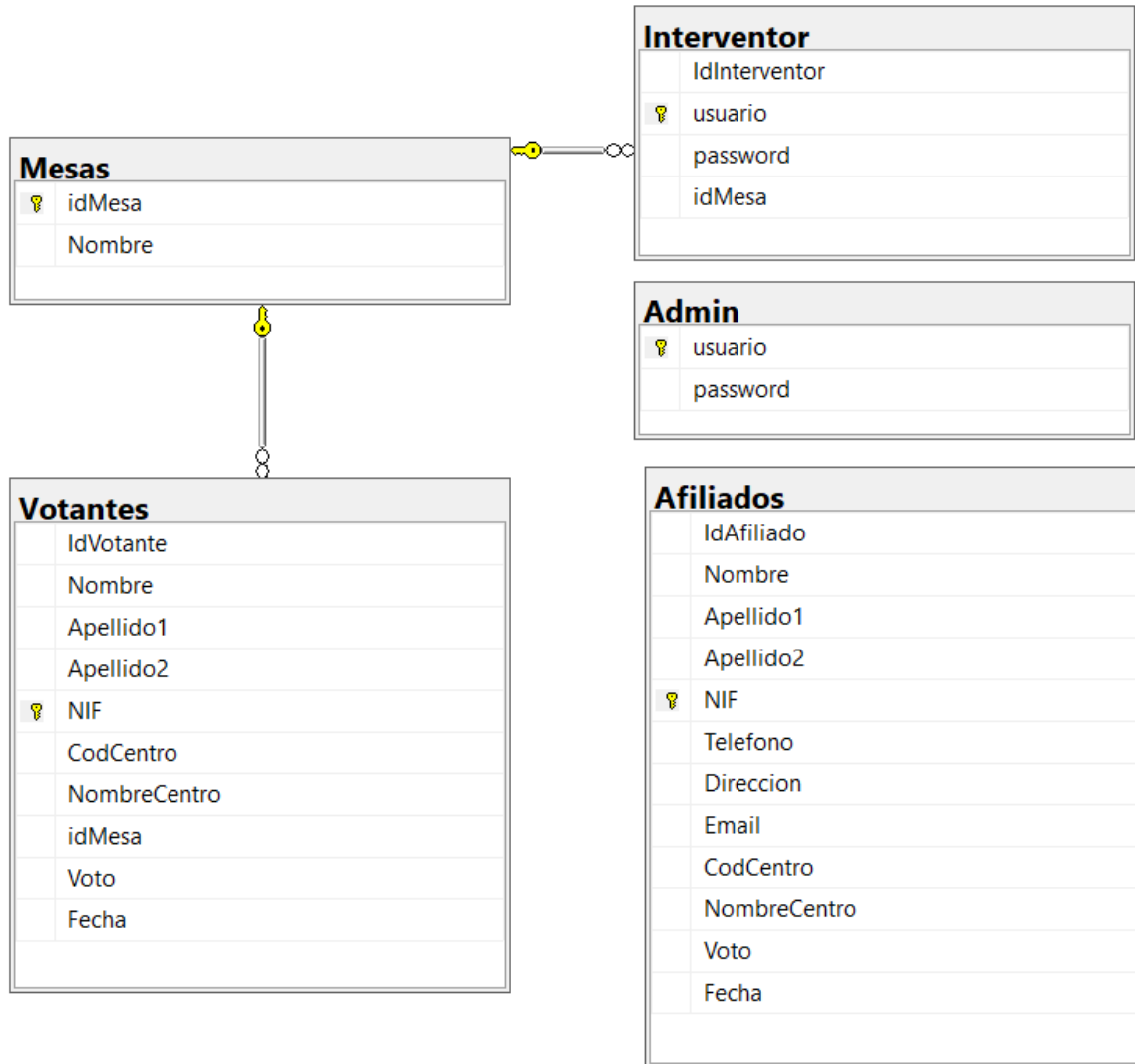


Tabla Admin:

- Usuario: Nombre de usuario del administrador. Clave primaria.
- Password: Contraseña del usuario administrador.

Tabla Mesas:

- idMesa: Numero de mesa. Clave primaria.
- Nombre: Nombre de la mesa.

Tabla Votantes:

- IdVotante: El id del votante.
- Nombre: Nombre del votante.
- Apellido1: Primer apellido del votante.
- Apellido2: Segundo apellido del votante.
- NIF: NIF del votante. Clave primaria.
- CodCentro: Código del centro.
- NombreCentro: Nombre del centro.
- idMesa: Clave foránea de la tabla Mesas, a la que está asignada el votante.
- Voto: Campo para saber si ha votado o no.
- Fecha: La fecha en caso de que haya votado.

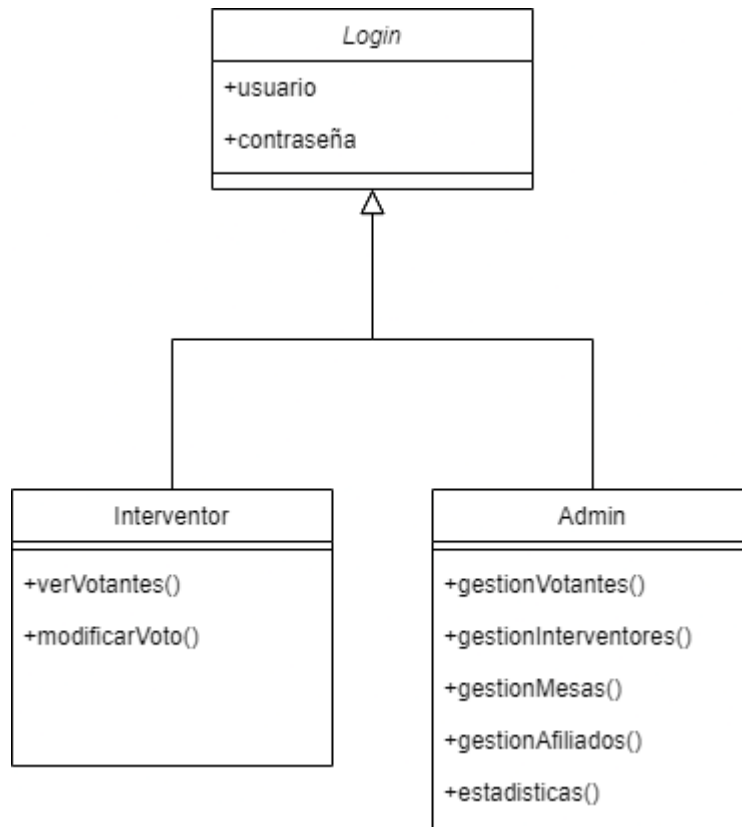
Tabla Afiliados:

- IdAfiliado: El id del afiliado.
- Nombre: Nombre del votante.
- Apellido1: Primer apellido del afiliado.
- Apellido2: Segundo apellido del afiliado.
- NIF: Clave primaria. NIF del afiliado.
- Teléfono: Número de teléfono del afiliado.
- Dirección: Dirección a la que pertenece el afiliado.
- Email: Email de contacto del afiliado.
- CodCentro: Código del centro.
- NombreCentro: Nombre del centro.
- Voto: Campo para saber si ha votado o no.
- Fecha: La fecha del voto.

Tabla Interventor:

- Usuario: Nombre de usuario del interventor. Clave primaria.
- Password: Contraseña generada de forma aleatoria.
- idMesa: Clave foránea de la tabla Mesas, a la que está asignada el interventor.

Diagrama de clases



Diagramas de flujo

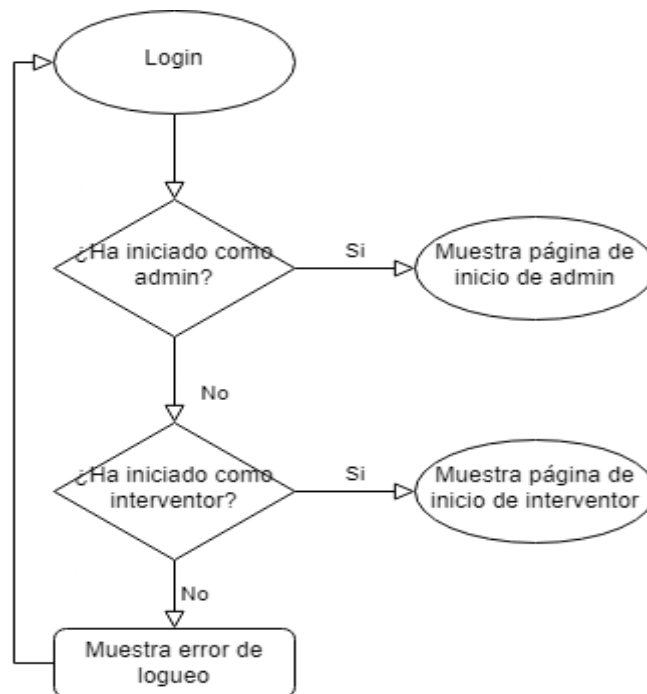


Diagrama de flujo de inicio de sesión.

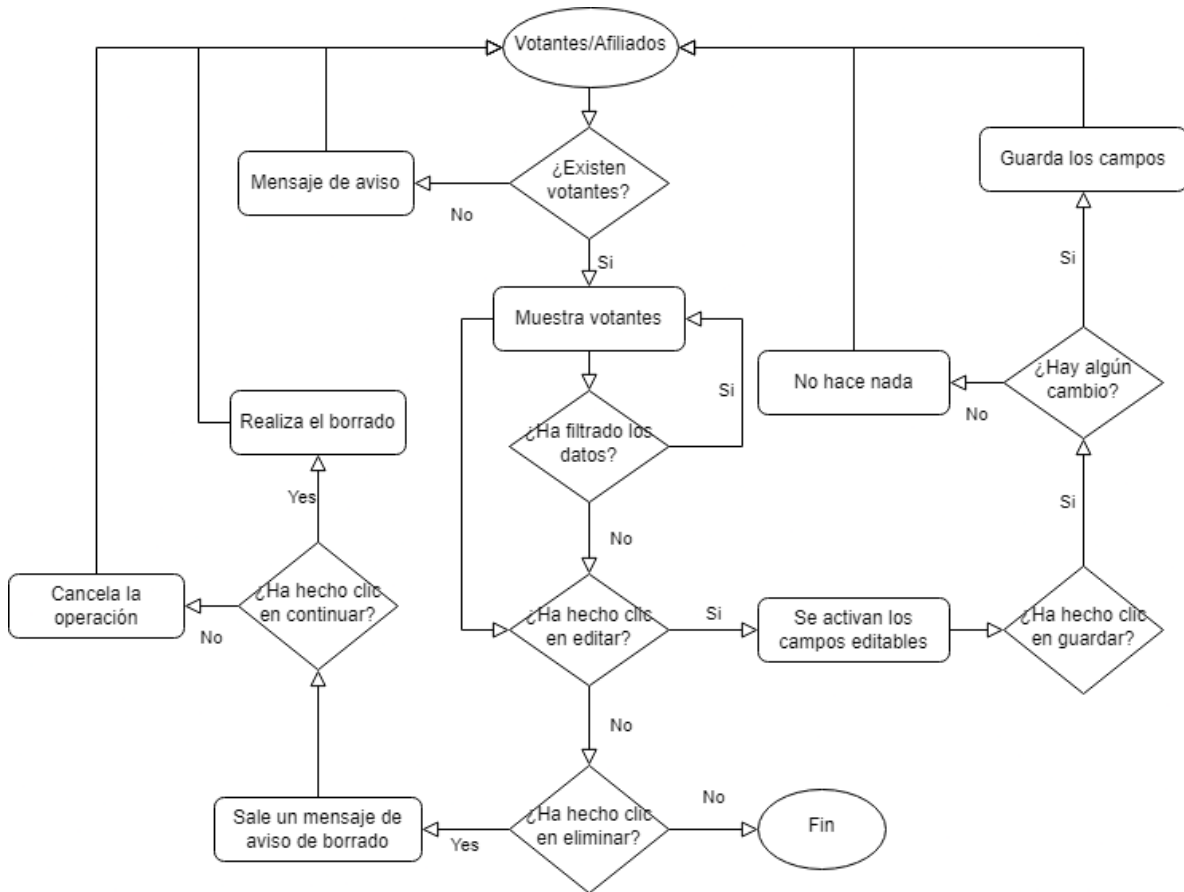


Diagrama de flujo de la página de votantes y afiliados.

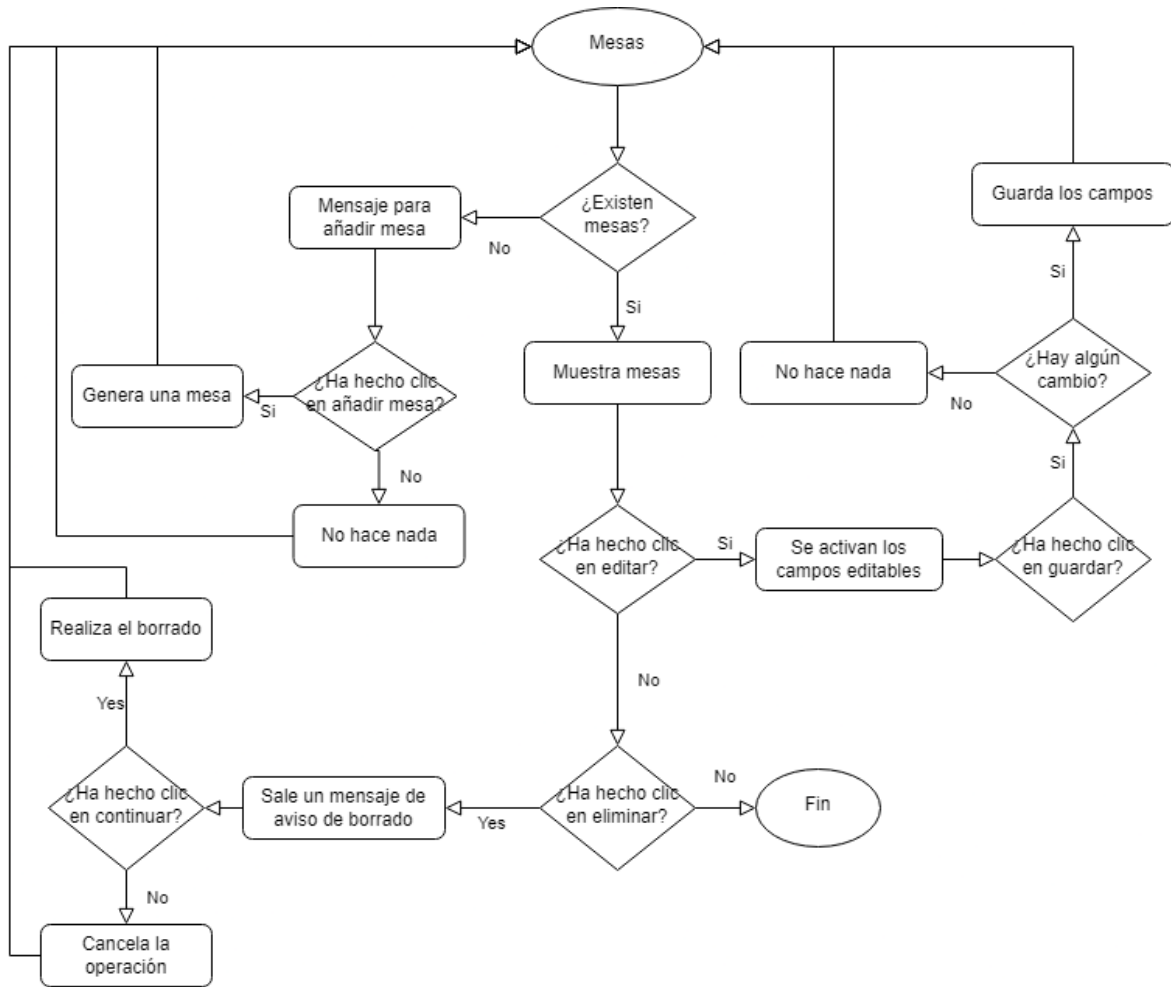
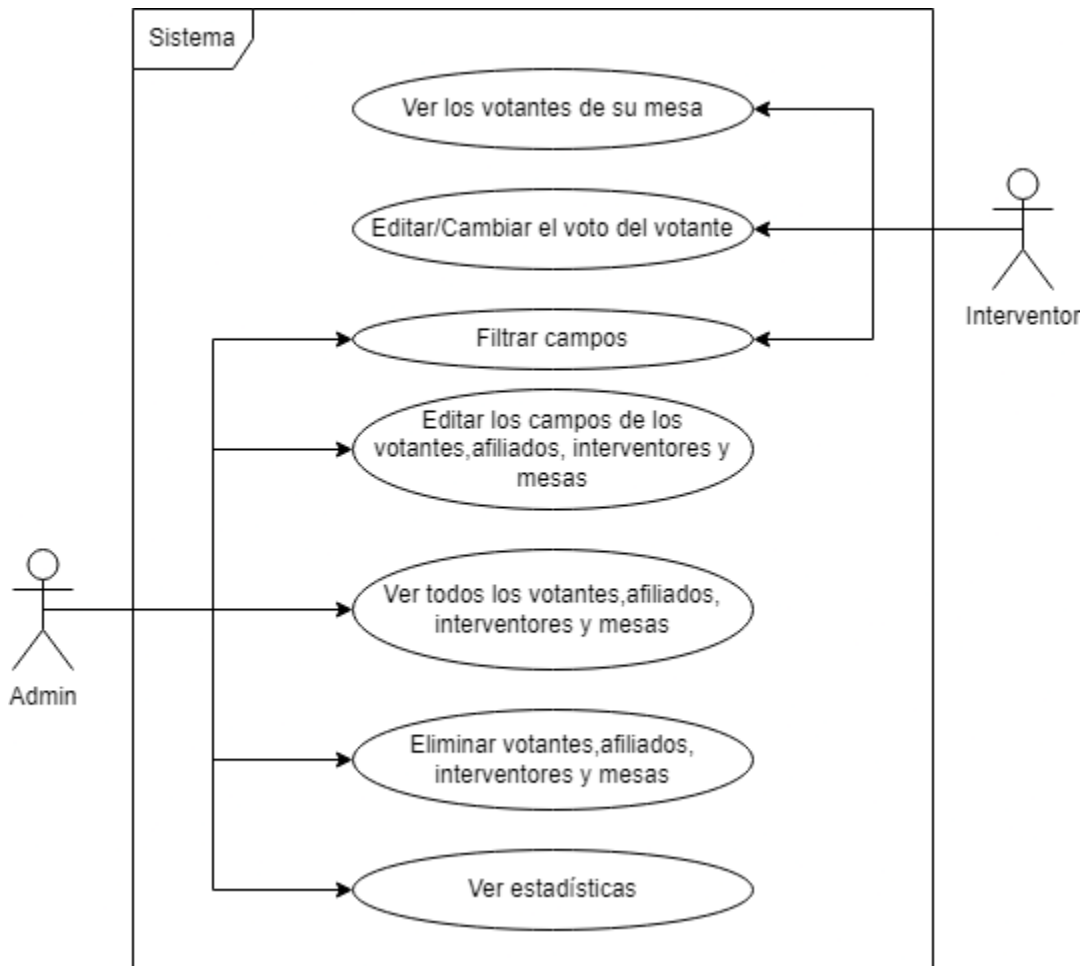


Diagrama de flujo de la página mesas.

Casos de uso



REQUISITOS FUNCIONALES	
RF-1	Fácil y sencilla de utilizar.
RF-2	El usuario podrá iniciar sesión como Administrador o como Interventor.
RF-3	Será lo más responsive posible.
RF-4	El menú del administrador estará dividido en: Mesas, votantes, afiliados, interventores, estadísticas y salir.
RF-5	El menú del interventor estará dividido en: Votos de la mesa y salir.
RF-6	El administrador podrá ver las contraseñas de los interventores.
RF-7	El administrador podrá ver, editar, eliminar y guardar las mesas, votantes, afiliados e interventores.
RF-8	El interventor podrá cambiar el voto de los votantes de su mesa.
RF-9	Se podrán realizar filtro, con el fin de encontrar la información más rápido.

REQUISITOS NO FUNCIONALES	
RF-1	El tiempo de aprendizaje del sistema por un usuario deberá ser menor a 1 hora.
RF-2	La tasa de errores cometidos por el usuario deberá ser menor del 1%
RF-3	El sistema debe proporcionar mensajes de error que sean informativos al usuario final.
RF-4	El sistema debe asegurar que los datos estén protegidos del acceso no autorizado.
RF-5	El sistema incluirá un procedimiento de autorización de usuarios (login), en el cual los usuarios deben identificarse usando un nombre de usuario y contraseña. Sólo los usuarios autorizados de esta forma podrán acceder a los datos del sistema

Implementación y configuración

Una vez instalado SSMS, lo primero de todo será empezar con la base de datos. La crearemos en la base de datos y haremos uso de ella con los siguientes comandos:

```
CREATE DATABASE EleccionesSindicales;  
USE EleccionesSindicales;
```

Posteriormente crearemos las tablas necesarias y sus respectivos atributos.

```
CREATE TABLE Admin (  
    "usuario" VARCHAR(50) NOT NULL,  
    "password" VARCHAR(50) NOT NULL,  
    PRIMARY KEY ("usuario")  
);
```

La tabla Admin.

```
CREATE TABLE Mesas (  
    idMesa INT NOT NULL,  
    Nombre VARCHAR(50) NOT NULL,  
    PRIMARY KEY (idMesa)  
);
```

La tabla Mesas.

```
CREATE TABLE Votantes (  
    IdVotante INT NOT NULL IDENTITY(1,1),  
    Nombre VARCHAR(50) NOT NULL,  
    Apellido1 VARCHAR(50) NOT NULL,  
    Apellido2 VARCHAR(50) NOT NULL,  
    NIF CHAR(9) NOT NULL,  
    CodCentro VARCHAR(10) NOT NULL,  
    NombreCentro VARCHAR(50) NOT NULL,  
    idMesa INT,  
    Voto TINYINT NOT NULL DEFAULT 0,  
    Fecha DATETIME DEFAULT NULL, --AAAA-MM-DD HH:MM:SS  
    PRIMARY KEY (NIF),  
    FOREIGN KEY (idMesa) REFERENCES Mesas(idMesa)  
);
```

La tabla Votantes.

```
CREATE TABLE Afiliados (  
    IdAfiliado INT NOT NULL IDENTITY(1,1),  
    Nombre VARCHAR(50) NOT NULL,  
    Apellido1 VARCHAR(50) NOT NULL,  
    Apellido2 VARCHAR(50) NOT NULL,  
    NIF CHAR(9) NOT NULL,  
    Telefono INT NOT NULL,  
    Direccion VARCHAR(100) NOT NULL,  
    Email VARCHAR(100) NOT NULL,  
    CodCentro VARCHAR(10) NOT NULL,  
    NombreCentro VARCHAR(50) NOT NULL,  
    Voto TINYINT NOT NULL DEFAULT 0,  
    Fecha DATETIME DEFAULT 0, --AAAA-MM-DD HH:MM:SS  
    PRIMARY KEY (NIF),  
);
```

La tabla Afiliados.

```
CREATE TABLE Interventor (  
    "usuario" VARCHAR(50),  
    "password" VARCHAR(50),  
    "idMesa" INT,  
    PRIMARY KEY ("usuario"),  
    FOREIGN KEY ("idMesa") REFERENCES Mesas("idMesa") ON DELETE SET NULL  
);
```

La tabla Interventor.

```
INSERT INTO Admin ("usuario", "password")  
VALUES ('admin', '1234');
```

Insertaremos un usuario administrador en la base de datos.

Y con esto ya tendremos la base de datos lista.

En cuanto al desarrollo del proyecto, he utilizado un código base (con la estructura MVC) hecho en el lenguaje de PHP. Las carpetas y ficheros más importantes son:

La carpeta “config”, que contendrá los ficheros:

- config.inc: Contendrá la configuración y conexión con la base de datos.
- rutas.inc: Contendrá las rutas de la aplicación, haciendo uso de los controladores y funciones de estos.

La carpeta “plantillas”, que como su propio nombre indica, contendrá las plantillas o vistas de la aplicación.

La carpeta “core”, que contendrá un fichero “conexionBd.inc” con la conexión a la base de datos.

La carpeta “fuente”, que contendrá las carpetas:

- Controlador: Contendrá ficheros con los controladores de la aplicación.
- Modelo: Contendrá ficheros con modelos de la aplicación (Utilizable o no).
- Repositorio: Contendrá ficheros con los repositorios de la aplicación.

La carpeta “web”, que contendrá las carpetas:

- CSS: Con los ficheros CSS.
- Imágenes: Con las imágenes de la página.
- JS: Con los ficheros JavaScript.

Finalmente, tendremos el archivo index.php.

Respecto a las configuraciones de los ficheros tendremos:

Configuración del archivo config.inc:

```
public function __construct()
{
    $this->dbParams =
    [
        "driver" => "pdo_sqlsrv", //no utilizada realmente
        "server" => "(local)", //nombre servidor
        "port" => "1433", //puerto por el que escucha las peticiones de la aplicación
        "database" => "EleccionesSindicales", //nombre de la base de datos
        "user" => null, //usuario definido para la aplicación
        "pass" => null, //contraseña del usuario
        "charset" => "utf-8" //conjunto de caracteres
    ];
}
```

En “database” cambiaremos el nombre por el que le hayamos puesto al crear la base de datos en el SSMS. El puerto lo dejaremos por defecto.

Configuración del archivo rutas.inc:

```
$mapeoRutas = [
    // Login
    'login' => ['controller' => 'app\EleccionesSindicales\controlador\SessionController', 'action' => 'login'],
    // Logout
    'logout' => ['controller' => 'app\EleccionesSindicales\controlador\SessionController', 'action' => 'logout'],
    // Pagina inicial - Admin
    'gestionVotantes' => ['controller' => 'app\EleccionesSindicales\controlador\VotantesController', 'action' => 'verVotantes'],
    // Gestion Interventores - Admin
    'gestionInterventores' => ['controller' => 'app\EleccionesSindicales\controlador\InterventorController', 'action' => 'verInterventores'],
    // Gestion Mesas - Admin
    'gestionMesas' => ['controller' => 'app\EleccionesSindicales\controlador\MesasController', 'action' => 'verMesas'],
    // Gestion Afiliados - Admin
    'gestionAfiliados' => ['controller' => 'app\EleccionesSindicales\controlador\AfiliadosController', 'action' => 'verAfiliados'],
    // Estadisticas - Admin
    'estadisticas' => ['controller' => 'app\EleccionesSindicales\controlador\EstadisticasController', 'action' => 'verEstadisticas'],
    // Pagina inicial - Interventores
    'inicioInterventor' => ['controller' => 'app\EleccionesSindicales\controlador\InterventorController', 'action' => 'inicioInterventor'],
];
```

La estructura sería la siguiente:

Nombre al que llamamos en el base.php (En la barra de navegación) => Ubicación del controlador => La función a la que llamamos.

En cuanto a las plantillas, tendremos el base.php:

```
<nav>
  <hr>
  <?php if (isset($_SESSION['admin'])) { ?>
    <a href="index.php?ctl=gestionMesas">Mesas</a>
    <a href="index.php?ctl=gestionVotantes">Votantes</a>
    <a href="index.php?ctl=gestionAfiliados">Afiliados</a>
    <a href="index.php?ctl=gestionInterventores">Interventores</a>
    <a href="index.php?ctl=estadisticas">Estadísticas</a>
    <a href="index.php?ctl=logout" class="salir">Salir</a>
    <hr>
  <?php } ?>

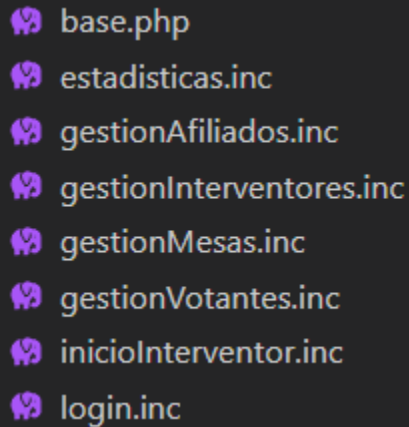
  <?php if (isset($_SESSION['interventor'])) { ?>
    <a href="index.php?ctl=inicioInterventor">Votos de la mesa</a>
    <a href="index.php?ctl=logout" class="salir">Salir</a>
    <hr>
  <?php } ?>
</nav>
<div id="contenido">
  <?= $contenido ?>
</div>
<?php if ($_SESSION) { ?>
  <footer>
    <hr>
    <p id="piePagina">— Pie de página —</p>
  </footer>
<?php } ?>
</body>
```

Aquí se definirá la barra de navegación que verán los usuarios cuando inicien sesión.

El id CSS facilita (si se define) la definición del aspecto visual de su contenido.

La variable \$contenido hará que se muestre la plantilla concreta, el contenido concreto, según la solicitud realizada por el usuario.

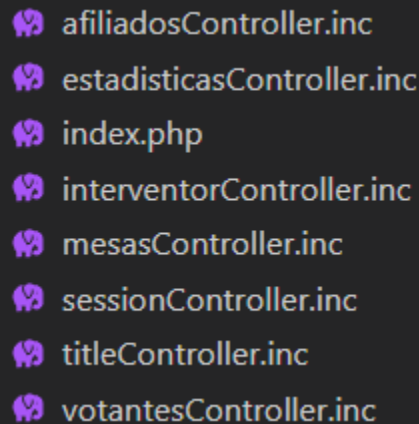
En este fichero también se definirán las rutas de los ficheros CSS, JS o Bootstrap (en caso de utilizarlo).



```
base.php
estadisticas.inc
gestionAfiliados.inc
gestionInterventores.inc
gestionMesas.inc
gestionVotantes.inc
inicioInterventor.inc
login.inc
```

Luego tendremos distintas plantillas que se irán utilizando según la solicitud realizada por el usuario.

En cuanto a los controladores, tendremos que:



```
afiliadosController.inc
estadisticasController.inc
index.php
interventorController.inc
mesasController.inc
sessionController.inc
titleController.inc
votantesController.inc
```

SessionController: Controlador para el login/logout de los usuarios. Contendrá las funciones de:

- login (): Elimina los valores guardados en \$_SESSION. Comprueba que el usuario exista en la base de datos, si existe y es:
 - Administrador: Le redirigirá a una página y pondrá el valor \$_SESSION['admin'] a verdadero.
 - Interventor: Le redirigirá a una página y pondrá el valor \$_SESSION['interventor'] a verdadero.Si no existe, notificará al usuario con un error.
- logout (): Redirigirá a login, ya que en el login se eliminan los valores de \$_SESSION.

InterventorController: Es el controlador de los interventores. Este tendrá distintas funciones como:

- `inicioInterventor ()`: Muestra al interventor su página de inicio, devolviendo así su lista de votantes. El interventor será capaz de modificar el voto de los votantes de su mesa.
- `randomPassword ()`: Genera una contraseña aleatoria de longitud 8 para los interventores.
- `verInterventores ()`: Sobre él se realiza la gestión de los interventores por parte del admin.

MesasController: Es el controlador de las mesas. Este tendrá distintas funciones como:

- `verMesas ()`: Sobre él se realiza la gestión de los interventores por parte del admin.
- `getAllMesas ()`: Función que retorna todas las mesas.

AfiliadosController: Es el controlador de los afiliados. Este tendrá distintas funciones como:

- `verAfiliados ()`: Sobre él se realiza la gestión de los afiliados por parte del admin.
- `descargarCSV ()`: Método que descarga los datos en formato csv.

EstadisticasController: Es el controlador de las estadísticas. Este tendrá distintas funciones como:

- `verEstadísticas ()`: Sobre él se realiza la gestión de las estadísticas por parte del admin.

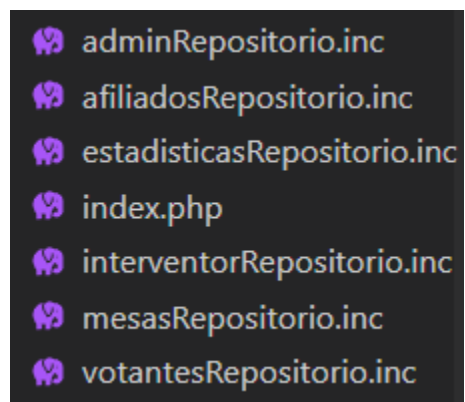
VotantesController: Es el controlador de los votantes. Este tendrá distintas funciones como:

- `verVotantes ()`: Sobre él se realiza la gestión de los votantes por parte del admin.
- `descargarCSV ()`: Método que descarga los datos en formato csv.

TitleController: Controlador para cambiar los títulos de las páginas. Contendrá la función:

- `setTitle ($title)`: Cambia el título de la página.

También tendremos unos repositorios que los utilizarán los controladores.



```
adminRepositorio.inc
afiliadosRepositorio.inc
estadisticasRepositorio.inc
index.php
interventorRepositorio.inc
mesasRepositorio.inc
votantesRepositorio.inc
```

AdminRepositorio:

- Login (): Comprueba que exista el usuario admin en la base de datos, devolviendo true o false.

AfiliadosRepositorio:

- getAfiliados (): Devuelve un array con todos los afiliados.
- borraAfiliado (): Borra un afiliado por ID.
- existeNIF (): Comprueba si existe o no un NIF, devolviendo verdadero o falso.
- guardarCambiosAfiliados (): Guarda los cambios realizados por el admin. En caso de error, devuelve falso y se hace rollback. En caso de éxito, devuelve verdadero.
- resetVotos (): Resetea los votos de los afiliados.

EstadisticasRepositorio:

- estadisticasVotantes (): Devuelve un array con las estadísticas de los votantes.
- totalVotantes (): Devuelve un recuento de los votantes totales, y de los votos totales (tanto si han votado como si no).
- estadisticasAfiliados (): Devuelve un array con las estadísticas de los afiliados.

InterventorRepositorio:

- loginInterventor (): Comprueba que exista el usuario interventor en la base de datos, devolviendo true o false.
- existeInterventor (): Comprueba si existe el nombre de usuario de interventor al modificarlo en la página de Interventores por parte del admin, devolviendo un booleano.
- generalInterventorAdmin (): Genera un interventor.
- borraInterventor (): Borra un interventor por ID.
- getAllInterventores (): Devuelve un array con todos los interventores.
- guardarCambiosInterventores (): Guarda los cambios realizados por el admin. En caso de error, devuelve falso y se hace rollback. En caso de éxito, devuelve verdadero.
- listaInterventor (): Devuelve un array con los votantes de una mesa (Página inicio de interventor).
- guardaVotoSi (): Cambia el voto a Si, por parte del interventor.
- guardaVotoNo (): Cambia el voto a No, por parte del interventor.

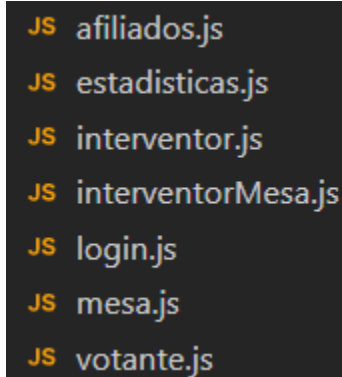
MesasRepositorio:

- generaMesa (): Genera una mesa.
- getAllMesas (): Devuelve un array con todas las mesas.

- `borraMesa ()`: Borra una mesa por ID. En caso de que algún interventor/votante tenga asignada una mesa, se le asigna la mesa 0 (Sin Asignar).
- `guardarCambiosMesa ()`: Guarda los cambios realizados por el admin. En caso de error, devuelve falso y se hace rollback. En caso de éxito, devuelve verdadero.

VotantesRepositorio:

- `getVotantes ()`: Devuelve todos los votantes.
- `existeNIF ()`: Comprueba si existe o no un NIF, devolviendo verdadero o falso.
- `guardarCambiosVotantes ()`: Guarda los cambios realizados por el admin. En caso de error, devuelve falso y se hace rollback. En caso de éxito, devuelve verdadero.
- `borraVotante ()`: Borra un votante por ID.
- `resetVotos ()`: Resetea los votos de los votantes.



```
JS afiliados.js
JS estadisticas.js
JS interventor.js
JS interventorMesa.js
JS login.js
JS mesa.js
JS votante.js
```

Afiliados.js: Fichero JS de la página de afiliados por parte del admin. Contiene funciones como:

- `editaAfiliado ()`: Habilita que el administrador pueda editar los campos de un afiliado.
- `guardadoExitoso ()`: Mensaje al guardar.
- `scrollVertical ()`: Método que implementa (si lo necesita) scroll vertical/horizontal.
- `ordenarTabla ()`: Método que permite ordenar los datos de la tabla desde las cabeceras de la tabla.
- `ordenarFilas ()`: Cambia el orden de las filas de la tabla. Es un método auxiliar de `ordenarTabla`.
- `obtenerColumnaIndex ()`: Obtiene el índice de la columna.
- `obtenerContenidoColumna ()`: Obtiene el contenido de la columna.
- `guardarFilasOriginales ()`: Guarda el orden por default de la tabla.
- `resetearIconos ()`: Resetea los iconos permitiendo ordenar solo por 1 columna.
- `busquedaFiltro ()`: Método que permite filtrar los datos.
- `removeAccents ()`: Elimina los acentos y caracteres especiales, método auxiliar usado en `busquedaFiltro`.

- confirmación (): Ventana de aviso al borrar un afiliado. Se usa el plugin SweetAlert.

Estadisticas.js: Fichero JS de la página de estadísticas por parte del admin. Contiene funciones como:

- refresh(): Método que añade una etiqueta meta para que se refresque la página cada 30 segundos.
- scrollVertical (): Método que implementa (si lo necesita) scroll vertical/horizontal.
- grafico (): Método que inserta un gráfico responsivo. Se utiliza la librería Chart.js.

Interventor.js: Fichero JS de la página de interventor por parte del admin. Contiene funciones como:

- scrollVertical ():Método que implementa (si lo necesita) scroll vertical/horizontal.
- mostrarContraseña (): Habilita al administrador ver la contraseña de un interventor (sin editarla).
- copiarContraseña (): Permite copiar al portapapeles la contraseña del interventor.
- editarInterventor (): Habilita que el administrador pueda editar los campos de un interventor.
- guardadoExitoso (): Mensaje al guardar.
- anadidoExitoso (): Mensaje al añadir.
- confirmacion (): Ventana de aviso al borrar un afiliado. Se usa el plugin SweetAlert.

InterventorMesa.js: Fichero JS de la página del interventor. Contiene funciones como:

- scrollVertical ():Método que implementa (si lo necesita) scroll vertical/horizontal.
- ordenarTabla (): Método que permite ordenar los datos de la tabla desde las cabeceras de la tabla.
- ordenarFilas (): Cambia el orden de las filas de la tabla. Es un método auxiliar de ordenarTabla.
- obtenerColumnaIndex (): Obtiene el índice de la columna.
- obtenerContenidoColumna (): Obtiene el contenido de la columna.
- guardarFilasOriginales (): Guarda el orden por default de la tabla.
- resetearIconos (): Resetea los iconos permitiendo ordenar solo por 1 columna.
- busquedaFiltro (): Método que permite filtrar los datos.
- removeAccents (): Elimina los acentos y caracteres especiales, método auxiliar usado en busquedaFiltro.

Login.js: Fichero JS de la página del login. Al entrar a la página selecciona el cuadro del usuario.

Mesa.js: Fichero JS de la página de mesas por parte del admin. Contiene funciones como:

- scrollVertical (): Método que implementa (si lo necesita) scroll vertical/horizontal.
- copiarNombreMesa: Permite copiar al portapapeles el nombre de la mesa.
- editarNombreMesa: Habilita al administrador editar el nombre de la mesa.
- guardadoExitoso (): Mensaje al guardar.
- anadidoExitoso (): Mensaje al añadir.
- confirmacion (): Ventana de aviso al borrar un afiliado. Se usa el plugin SweetAlert.

Votante.js: Fichero JS de la página de mesas por parte del admin. Contiene funciones como:

- scrollVertical (): Método que implementa (si lo necesita) scroll vertical/horizontal.
- ordenarTabla (): Método que permite ordenar los datos de la tabla desde las cabeceras de la tabla.
- ordenarFilas (): Cambia el orden de las filas de la tabla. Es un método auxiliar de ordenarTabla.
- obtenerColumnaIndex (): Obtiene el índice de la columna.
- obtenerContenidoColumna (): Obtiene el contenido de la columna.
- guardarFilasOriginales (): Guarda el orden por default de la tabla.
- resetearIconos (): Resetea los iconos permitiendo ordenar solo por 1 columna.
- busquedaFiltro (): Método que permite filtrar los datos.
- removeAccents (): Elimina los acentos y caracteres especiales, método auxiliar usado en busquedaFiltro.
- editarVotante (): Habilita que el administrador pueda editar los campos de un votante.
- guardadoExitoso (): Mensaje al guardar.
- confirmacion (): Ventana de aviso al borrar un afiliado. Se usa el plugin SweetAlert.

Configuración del archivo index.php:

```
spl_autoload_register(function (string $clase) {
    if (strpos($clase, 'app\\EleccionesSindicales\\') === 0) {
        $nombre = str_replace('app\\EleccionesSindicales\\', '', $clase);
        $nombre = str_replace('\\', '/', $nombre);
        require_once __DIR__ . '/fuente/' . $nombre . '.inc';
    }
});

require_once __DIR__ . '/app/conf/rutas.inc'; /*Ubicación del archivo de rutas*/

if (isset($_SESSION['admin'])) {
    $_SESSION['admin'] = false;
}
if (isset($_SESSION['interventor'])) {
    $_SESSION['interventor'] = false;
}

// Parseo de la ruta
if (isset($_GET['ctl'])) {
    if (isset($mapeoRutas[$_GET['ctl']])) {
        $ruta = $_GET['ctl'];
    } else {
        header('Status: 404 Not Found');
        echo '<html><body><h1>Error 404: No existe la ruta <i>' .
            $_GET['ctl'] .
            '</p></body></html>';
        exit;
    }
} else {
    // Si no se especifica una ruta, se redirige al login
    $ruta = 'login';
}
```

La función spl_autoload_register: Es el último lugar en el que busca PHP para intentar resolver un tipo /interfaz..etc, antes de generar un error. Con esta función evitamos llenar el fichero de controladores.

Añadimos 2 condiciones if, las cuales las usamos cuando el usuario inicie sesión o se desconecte. Gracias a eso, posteriormente podrá ver o no ciertas cosas de la barra de navegación.

En la función de parseo de rutas, al parserla, cambiamos la ruta de redirección, que será login en vez de index.

Pruebas

Las siguientes pruebas han sido llevadas a cabo con éxito de forma manual:

- Se ha añadido inicio sesión. En caso de fallo, mostrará un error.

- Se ha corregido que el usuario cuando se desconecte, no vea la barra de navegación.
- Se han corregido errores en las rutas y redireccionamientos.
- Se ha corregido un error donde al borrar las mesas, el idMesa no volvía a 1.
- Se han corregido errores varios en la base de datos.
- Se ha corregido un error donde al insertar el interventor, había que poner un idMesa. Ahora si no se pone, su valor será null y:
 - Mostrará un mensaje en la gestión de Interventores como “Sin asignar”.
 - Si inicia sesión como interventor, le mostrar un mensaje diciendo que no tiene mesa asignada.
- Se ha añadido contraseña aleatoria de longitud 8 a los interventores.
- Se ha añadido el detalle de cambio de título al moverte entre las páginas.
- Se han corregido errores de permisos/vistas.
- Se ha añadido CRUD en Mesas/Afiliados/Votantes/Interventores.
- Se han añadido nuevos mensajes al añadir, borrar o de error.
- Se ha cambiado que se pueda editar el campo usuario de Interventores, antes solo permitía la contraseña.
- Se ha añadido como campo obligatorio el NIF (Los demás según cliente no hacen falta).
- Se ha solucionado un error de las fechas al editar y guardar el campo de voto. Ahora se muestra y guarda correctamente.
- Se han añadido filtros tanto general como a las cabeceras en las páginas de Votantes y Afiliados. Además, en la página de Estadísticas se han añadido estadísticas incluyendo así un gráfico.
- Se ha borrado el AdminController, ya que no se realiza ninguna gestión respecto al administrador en sí.
- Se han añadido los botones de resetear votos y guardar la información en ficheros con formato csv.
- Se ha añadido un botón que permite copiar contraseña/nombre de mesa.
- Se ha añadido scroll vertical y horizontal dependiendo del tamaño del dispositivo y de los datos a mostrar.
- Se ha añadido patrón regex al campo NIF.
- Se ha añadido mensaje de aviso al eliminar.

A continuación, repasando con el cliente la página se han realizado las siguientes pruebas:

Tipo	Descripción	¿Ha pasado la prueba?
Interfaz	Se visualizan correctamente las páginas.	Si
Interfaz	Se realizan y muestran correctamente las acciones de los usuarios.	Si
Rendimiento	Se prueba el rendimiento de la página	Si

Código	Se prueba que el código es funcional, y que no existan errores	Si
Base de datos	Se prueba que los usuarios puedan hacer login y logout.	Si
Base de datos	Se comprueba que se añade, elimina o edita correctamente la información.	Si

Despliegue

En cuanto al despliegue de la aplicación del cliente, es importante destacar que será responsabilidad del propio cliente llevar a cabo dicha gestión. Esto implica que el cliente deberá encargarse de la puesta en marcha de la aplicación en su entorno.

Además, es crucial mencionar que el cumplimiento del Reglamento General de Protección de Datos (RGPD) también recae en manos del cliente. El cliente deberá asegurarse de que la aplicación cumple con todos los requisitos legales y normativos establecidos por el RGPD para garantizar la protección y privacidad de los datos personales de los usuarios.

Ampliación y posibles mejoras

Como posibles mejoras y ampliación del proyecto, según mi criterio un par ellos serían:

Ampliaciones:

- Añadido scroll vertical y horizontal dependiendo del tamaño del dispositivo y cantidad de datos en la pantalla, evitando así el bajar la página.
- Añadido botón de resetear votos.
- Añadido botón para exportar la información en fichero .CSV.
- Añadido botón para copiar al portapapeles la contraseña/nombre de mesa.
- Al darle clic en las cabeceras de la tabla, éstas se ordenen de forma ascendente o descendente.
- Mejora en los filtros, filtrando mientras tecleas y no le tengas que dar a buscar.

Posibles mejoras:

- Mejora estética/visual de la aplicación.
- Mejora UX/UI de la aplicación.
- Mejora de rendimiento/fluidez de la aplicación (incluye refactorizar).
- Poder importar archivos Excel, evitando introducirlos desde la base de datos.

Conclusión

Durante la realización del Trabajo de Fin de Grado he investigado, aprendido e implementado funciones y métodos de programación que desconocía por completo.

En la etapa inicial se realizó un análisis de los componentes necesarios para la realización del proyecto, además de las reuniones con el cliente para especificar los aspectos a desarrollar.

Durante la etapa de desarrollo, se creó una arquitectura basada en el patrón MVC, para asegurar la separación de responsabilidades. Se implementaron diversas funcionalidades utilizando el lenguaje de programación PHP, como el CRUD, las estadísticas o los filtrados.

También se realizaron pruebas exhaustivas para garantizar la funcionalidad de la aplicación, corrigiendo errores y realizando mejoras.

El cliente dio el visto bueno a la aplicación, dando por concluido este Trabajo de Fin de Grado.

Bibliografía

Descarga SSMS:

<https://learn.microsoft.com/es-es/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>

Descarga IDE (VSCode):

<https://code.visualstudio.com/>

Descarga XAMPP:

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

Imagen/es ANPE:

<https://anperioja.es/>

Iconos:

<https://fontawesome.com/icons>

Stackoverflow:

<https://stackoverflow.com/>

Manual de PHP (Documentación):

<https://www.php.net/manual/es/index.php>

SweetAlert:

<https://sweetalert.js.org/guides/>

Chart.js:

<https://www.chartjs.org/>